Distributed Biconnectivity Testing in Wireless Multi-hop Networks

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät II
Humboldt-Universität zu Berlin

von
Dipl.-Ing. Bratislav Milic
24.07.1978 in Belgrad

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Christoph Markschies

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II:
Prof. Dr. Peter Frensch

Gutachter:
1. Prof. Dr. Edgar Nett
2. Prof. Dr. Alexander Reinefeld
3. Prof. Dr. Miroslaw Malek

eingereicht am: 01.09.2009
Tag der mündlichen Prüfung: 04.12.2009

First and foremost I would like to thank my supervisor, Professor Miroslaw Malek, for his continued support throughout my work at the Humboldt University. My thanks also go to all the colleagues for useful comments and fruitful discussions.
This work would not be possible without the understanding, patience, support and love of my family.

Abstract

Wireless multi-hop network (WMN) is a distributed communication system composed of autonomous processing nodes that is known for its ability to automatically adjust to rapidly changing conditions in the surrounding environment. Connectivity is one of the basic properties of a network. Removal of a bridge or an articulation point partitions a network. Biconnectivity testing identifies bridges and articulation points in a network, and once they are known corrective actions can be performed in order to improve network's reliability. Numerous biconnectivity testing algorithms are successfully applied in graphs, wired networks and multiprocessor systems. However, they are inadequate for application in wireless networks since the frequent packet losses introduce uncertainty in the system which these algorithms cannot handle. The stochastic analysis shows that errors in decision-making in WMNs are considerable even for seemingly simple tasks such as the detection of links.

The main contribution of this work is to provide means for accurate binary decision-making under uncertainty within the context of biconnectivity testing in WMNs. A distributed algorithm is developed that successfully handles the faults caused by message losses and simultaneously utilizes benefits of wireless communication to reduce message complexity from $O(e)$ to $O(n)$. Based on stochastic analysis of WMN topologies and a comprehensive analysis of impact of communication faults on algorithm's behavior, the algorithm is extended by voting theory to reduce probability of erroneous decisions.

The WMNs in Berlin and Leipzig are used as the case study. Topological data collected in them demonstrates that real networks are composed of dense and sparse parts, and confirms existence of numerous bridge and articulation points. The known node-placement algorithms are unable to recreate these properties so a new node placement algorithm for realistic topologies in WMN simulation was developed.

The algorithm and the voting rules are evaluated in experiments in Motelab testbed and in the event-based simulator Jist/SWANS. The algorithm is accurate under various conditions which demonstrates its applicability in reality and capability of successful operation in presence of packet losses.

Zusammenfassung

Ein drahtloses Multihop-Netzwerk (DMN) ist ein verteiltes Kommunikationssystem aus autonomen Verarbeitungsknoten, welches vor allem die Fähigkeit zur automatischen Anpassung an sich ständig änderne Umgebungsbedingungen hat. Eine zentrale Fragestellung in DMNen ist, ob das Netzwerk partitioniert ist, ob also nicht mehr jeder Knoten mit jedem anderen Knoten kommunizieren kann. Um festzustellen, ob eine Partitionierung droht werden mit Hilfe von 2-Zusammenhangstests Brücken und Artikulationspunkte im Kommunikationsgraphen gesucht. Daraufhin können anschließend korrektive Aktionen eingeleitet werden um die Partitionierung zu verhindern und somit die Netzwerkverfügbarkeit zu erhöhen. Eine Vielzahl von 2-Zusammenhangstestverfahren wurde bereits erfolgreich bei drahtgebundenen Netzen eingesetzt. Allerdings sind diese Verfahren ungeeignet für drahtlose Netze, da die Ungenauigkeiten durch den häufigen Paketverlust in solchen Systemen bisher nicht berücksichtigt wurden. Mit Hilfe von stochastischen Modellen wird gezeigt, dass Fehler in der Entscheidungsfindung für DMNen bereits bei sehr einfachen Problemen wie der Link-Erkennung signifikant sein können.

In dieser Arbeit werden daher verschiedene Verfahren präsentiert, die auch auf Grundlage unsicherer Informationen noch eine verlässliche Entscheidungsfindung ermöglichen. Die Arbeit präsentiert einen neuen verteilten Algorithmus zum Test auf 2-Zusammenhang, welcher Fehler durch Nachrichtenverlust berücksichtigt und gleichzeitig die Anzahl an Nachrichten reduziert. Basierend auf einer umfassenden Analyse der Einflüsse von Kommunikationsfehlern auf den Algorithmus, wurden Abstimmungsprozeduren entwickelt, die die Wahrscheinlichkeit von Fehlentscheidungen nochmals reduzieren.

Eine Fallstudie mit öffentlichen DMNen in Berlin und Leipzig hat die Existenz zahlreicher Brücken und Artikulationspunkte in echten Netzwerken bestätigt. Da keiner der bekannten Algorithmen zur Erzeugung von Topologien in simulierten DMNen zu annähernd realistischen Ergebnissen führt, wird ein neuer Algorithmus vorgestellt.

Zur weiteren Analyse werden die Algorithmen erstens in der Motelab-Umgebung und zweitens mit Hilfe von Simulationen untersucht. Die präsentierten Algorithmen zeigen überzeugende Ergebnisse unter variierenden Bedingungen, was ihre Anwendbarkeit in realen Szenarien unterstreicht.

Contents

1. Introduction

1.1. Wireless Multi-hop Networks

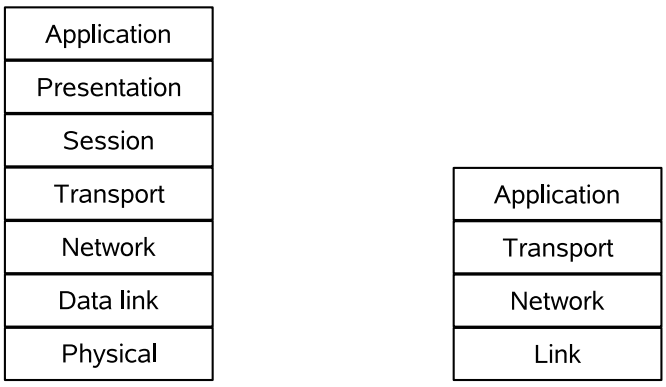
Wireless multi-hop networks (WMN) are composed of autonomous processing nodes that use wireless network adapters for communication and share common set of communication protocols. In case that two nodes are unable to communicate directly, a subset of network nodes is responsible for message relaying, hence they are called multi-hop. They form a distributed system that is primarily used for communication, although other distributed or centralized applications and services may be deployed in it. It is usually implied that WMNs act in accordance with general guidelines that are defined for best-effort networks [45].

WMNs are praised for their ability to automatically adjust to rapidly changing conditions in the surrounding environment, as well as for reduction of deployment duration, network setup and operational costs. In ideal case, effort invested in node configuration before their deployment is minimal, nodes may move, join and leave network freely, message routing is performed in a distributed manner with negligible overhead, and no central administrative body is required. Unfortunately, that does not come for free. When compared to widespread wired (LAN/WAN) or base station oriented wireless networks (WiMAX, UMTS) WMNs may have deficiencies in throughput, security, QoS guarantees, scalability, and billing mechanisms.

At the time of their introduction, they were envisioned as general purpose networks, applicable in wide class of scenarios. However, some of the proposed application scenarios, such as the communication in a general purpose mobile ad hoc network, did not match actual user needs [61], and have not been used in practice despite numerous developed protocols. Research community has realized the need for application-oriented protocols. The current trend is to focus on concrete scenarios and application areas, reusing the knowledge obtained in research of general purpose WMNs. Exemplary application scenarios include:

- Communication infrastructure is not available, deployment and operational costs of traditional (wired or wireless with base stations) infrastructure are not acceptable. Typical examples are general purpose community networks [7] [8] [10] [164] [2] [151], industry and logistics networks [20][89][92].

- Considerable node mobility imposes immense costs for fixed infrastructure since it must cover huge area. Examples are sensor networks for animal behavior tracking in wilderness [97], or vehicular wireless networks (VANETs) [1] [17] [18]. VANETs also possess inherent locality of data and services that are required by users. This locality is important argument in favor of WMNs and enables creation of interesting and useful applications.



(a) OSI basic reference model.

(b) TCP/IP stack.

Figure 1.1.: OSI and TCP/IP protocol stacks. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text.

- Pre-deployment of network infrastructure is not possible because it is unknown where it will be required (equivalently: extremely fast setup is needed) or environment is extremely hostile to deployed infrastructure like in military applications [139] [115].

- Existing infrastructure may be damaged or destroyed by external factors so self-reconfigurable WMNs are used as backup (or even primary) communication option. Disaster monitoring and management are prominent application fields [9] [86].

In all scenarios, the primary functional requirement of a WMN is clear – it should enable communication between pairs of nodes in the network. This complex task is divided into subtasks and consigned to a set of network protocols.

Traditionally, the network protocols are placed in layers of communication stack that isolate sets of common functionalities, provide clear interfaces between them and improve understanding of the whole system. Two most common protocol stacks are OSI (Open Systems Interconnection) reference model [182] and TCP/IP stack (Figure 1.1).

The division to layers, with a crisp-clear division of responsibilities among them, has been criticized for a while so they are often taken more as guidelines than as mandatory standards. In order to provide better utilization of scarce resources, considerable number of WMN protocols are developed as cross-layer: they combine functionality and improve quality of more than one layer simultaneously. Even if the cross-layer technique is not fully applied, it is common to (re)use information from surrounding layers for improvement of the targeted one.

In addition to mandatory correctness of functional behavior of WMN protocols, which is a complex task on its own, applications and network services require various additional non-functional properties. For instance, a general purpose wireless mesh network aims at throughput increase, latency reduction, and traffic fairness while the energy efficiency is of limited concern (most of nodes have permanent power sources). A typical large scale sensor network focuses on extreme energy efficiency in order to extend node and

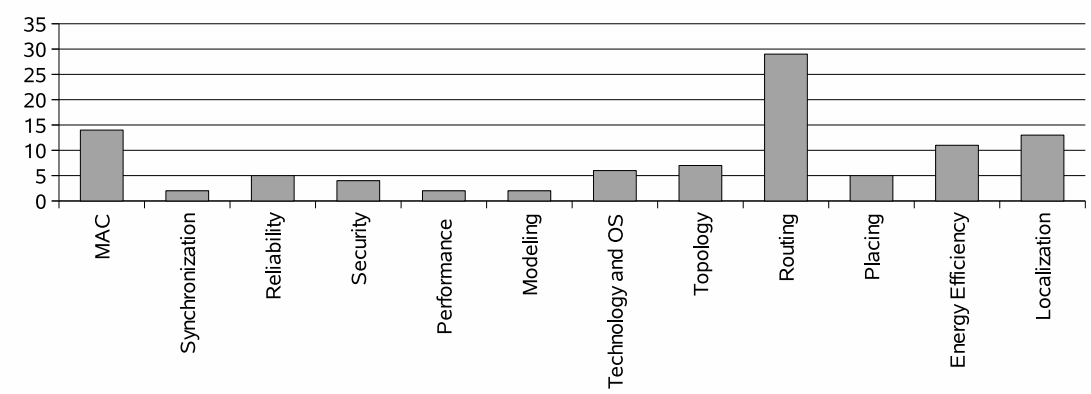


Figure 1.2.: Distribution of topics in WMN research [58].

network lifetime, data volumes that it transports are low to moderate, occasionally going to zero if no events of interest are observed. Fairness may be deliberately excluded to give priority to certain, more important events over others. Additionally, some sort of node localization is needed – usually it is important to know the event coordinates in space-time, not only in time. Obviously, the same routing protocol cannot fulfill the conflicting requirements imposed by sensor and mesh networks, so dedicated protocols must be employed.

Even if a single application scenario is of concern, different layers of communication stack may be targeted for improvement. Cinque et al. [58] have analyzed 114 papers from prominent IEEE and ACM journals and differentiated them by research topics. As it can be seen in Figure 1.2 majority of WMN research is focused on functional improvement of OSI layers two (data link) and three (network). There are some proposals for transport protocol improvement. Some parameters of physical layer are manipulated for topology control, but its core functionality is rarely addressed. Dependability[1] topics are rather neglected and majority of developed protocols assume one of the most important features of each network – its connectivity.

1.2. Motivation

The connectivity can be compromised and network partitioned by removal of a single bridge or an articulation point (communication link or node, respectively, whose removal disconnects a network). In a wireless multi-hop network (mobile or stationary), removal of links can be caused by changes in the environment that affect the properties of wireless channel, node movement or reduced transmission power of nodes. Removal of nodes can be a consequence of node software/hardware failure, user-initiated shutdown, physical destruction of a node, battery failure, etc.

A node or communication link failure does not necessarily partition a network. Therefore, it is important to detect nodes and links that are critical for the connectivity. If it is possible to perform biconnectivity testing (i.e., to identify bridges and articulation points in the network), various corrective or preventive actions can be performed in the

[1] reliability, availability, real time and security

network in order to improve its reliability or dependability of services deployed in it.

If corrective actions are not performed and bridge or articulation point is removed, the network partitions. It compromises the most important functionality of a network: communication between nodes from different partitions is no longer possible. Side effects of partitioning include dissatisfied users and unavailability of services offered by the network, in particular, if composite services are deployed. If density of nodes and their rejuvenation rate is low, or movement patterns unfavorable, the network may remain disconnected for prolonged time periods, considerably reducing availability of its services.

The existence of bridges and articulation points in a WMN additionally increases probability of network disconnection if compared with a wired network. In wired networks, it is possible that a node or an edge fails due to hardware or software errors and this probability is approximately the same for all nodes in the network. Such failure mode applies to WMN nodes as well. However, the energy constraints of WMNs (e.g., mobile or sensor networks) introduce additional node failure mode – the battery failure. Since the articulation points are the only gateways between biconnected components of a network, they tend to forward more traffic than other nodes in network. Thus, it is more likely for articulation points to deplete their energy sources than ordinary nodes, fail, and partition the network.

Furthermore, it is demonstrated in Section 9.7 that even if there exists a route between two nodes, the routing layer occasionally cannot find it and declares the destination node as unreachable (logical partitioning). Analysis has been performed and its outcome is that the main source of problems are bridges which connect large network components. Route-request messages that are lost on bridges do not reach whole components of the network since there is no alternative path connecting the components. In denser parts of network, losses of individual route-request messages are masked by redundancy of their dissemination mechanism.

The issues of weak connectivity have been largely overlooked by the research community, who instead devoted its attention either to dense and (almost always and at least) $2-$connected networks or extremely sparse, almost always disconnected networks.

In a dense network, bridge and articulation point are either non-existent or extremely rare, thus their impact is negligible and their detection brings minuscule benefits. Wide spread artificial node placement algorithms have created the unawareness of fragile connectivity encountered in real WMNs and its effects on deployed protocols. They are incapable of creating connected low-density topologies so in order to produce connected topologies, researchers are forced to use high node densities. If such setups are used in simulation studies, they eliminate bridges and articulation points from network topology and the issues which bridges and articulation points create in evaluated protocols.

The other extreme - Delay Tolerant Networks (DTN) operate under so harsh conditions that existence of bridges as a direct communication channel is observed as a favorable condition for network and its protocols. The protocols deployed in DTNs deliberately sacrifice interactiveness of applications and dramatically increase communication latency. However, reach and usability of DTN protocols is limited since most of applications are interactive – it is difficult to imagine an Internet user that is satisfied with an outdated weather forecast or a sensor network that reports a fire in a forest after it has been spreading several hours.

The measurements from Berlin's and Leipzig's mesh networks show that the truth

may be between these extremes: the networks are connected, but with lower average node density than in the typical artificial models used for analysis and simulation of WMNs. The real networks exhibit high inhomogeneity – there exist exceptionally dense sections of network but also very sparse parts. Finally, the sampled topologies have high share of bridges and articulation points.

The reasons for lack of the detailed approach to bridge and articulation point detection can be summarized in following four points:

- The need for detection and corrective actions was obscured by low number of bridges and articulation points that are observable in existing artificial topology generators [121].
- The protocols to be deployed in WMNs are evaluated in small testbeds [3] [4] or in simulators that use dense topologies. In such environments it is not possible to observe severity of issues that are created by bridges and articulation points.
- The absolute trust in network topology control has been established. It is assumed that node density in a network or their transmission power can be arbitrarily increased until $k-$connected network is obtained [36] [107]. This ignores cost factors (a set of nodes is available for the whole network and it cannot be doubled or tripled just to guarantee $k-$connectivity), physical node limitations (there is a maximum transmission power output of a node), environment factors (world is not flat, three dimensional objects can dramatically influence signal propagation) and most importantly the dynamics of the network: nodes fail during network's lifetime. Even if $k-$connectivity of a network was established during its deployment, this property may be invalidated by failure of several nodes. If node failures disconnect the network, it is no longer possible to execute the procedure of transmission power assignment for k-connectivity since it requires synchronization of node actions in all partitions (that can no longer communicate).
- In rare cases when bridges and articulation points have been observed as a potential issue, it has been taken for granted that proactive routing protocols are capable of delivering very accurate network topology information, that is required for biconnectivity testing by algorithms known from the graph theory [158]. This property of proactive routing protocols has been "confirmed" through simulation of dense topologies, using unrealistic signal propagation models. In reality, packet losses caused by channel fading decrease quality of topologies delivered by proactive topology dissemination. Typically, it is sufficient for routing but it is not precise enough for accurate biconnectivity testing.

1.3. Problem Statement and Goals

It may be said that it is better to eliminate bridges and articulation points from the network, instead attempting to discover them. However, the general approaches proposed in research that increase node density in a network or regulate node transmission power in order to improve network's connectivity have found limited application in reality.

The arbitrary increase in node density, as it was proposed in [36] [107], is not used in practical WMN planning and management. Instead, realistic assumptions of a limited

set of network resources with limited capabilities (e.g. transmission power) are used [92].

In example of community networks, users are located all over a city and will not move the nodes far away from their apartments just to improve network connectivity (the network exists because of users not the other way around). Every additional, non-user node imposes costs on hardware, setup, electricity, so such approach is used rarely or never.

The main assumption in this thesis is that network shape, density and node properties cannot be considerably changed. The involvement of human users must be minimized. They might not possess adequate technical skill, resources, or simply they have no desire to follow guidelines proposed by a networking protocol. It is difficult to imagine that these intrinsic human characteristics will change so that they will act as a protocol requires of them in order to eliminate weak network connectivity. Instead, it is proposed in this work to detect bridges and articulation points in order to be able to apply corrective actions to network, its services and data.

The existing approaches for bridge and articulation point detection rely on generous assumptions regarding capabilities of nodes and communication links. They introduce centralized servers [168], require global topology knowledge at each node [34] [57], depend on precise node localization [57] [80], perfectly circular communication radius [34] [80], instantaneous topology data dissemination [63], etc. They use biconnectivity testing algorithms known from the graph theory. These algorithms are optimized for fast execution and reduced memory consumption but assume perfectly accurate topology knowledge or no message losses if they are executed in the distributed manner. They are widely used in wired networks but they are not suitable for fast changing topologies that are encountered in WMNs, where packet losses are common, communication channels are unreliable, with rather low throughput and high latency. Proactiveness required for such algorithms does not scale and significantly increases contention on channel and energy consumption, while the distributed versions of biconnectivity testing algorithms encounter various data and control flow problems because of lost messages.

A network with unreliable message delivery creates considerable issues for all distributed algorithms executed in it. Since message delivery is no longer guaranteed, algorithms that are correct if no messages are lost, may behave erroneously. Even the "simple" tasks, such as the provision of complete system status[2] between two correct nodes is no longer possible in systems with unreliable communication [27].

Of particular importance for this work are limitations of link detection protocols imposed by unreliable communication in WMNs. It will be proven later that it is not possible to guarantee accurate link detection due to message losses. The accurate topology knowledge is the precondition of successful and accurate application of biconnectivity testing algorithms. This precondition no longer holds in WMNs because of the errors in link detection, biconnectivity testing algorithms are applied to incorrect topologies and their output may be erroneous.

The message losses in WMNs have their origins in physical laws, so they cannot be removed from the system by choosing a different biconnectivity testing algorithm or improving an existing one. The root-cause of issue is not in algorithms but in nature of

[2] In Section 3.1 of [27], the complete status is defined as "both parties know outcome of transaction and agree what happened".

the system, and as a consequence it is no longer possible to guarantee correct outcomes. In this work, a probabilistic approach is taken instead – the goal is to minimize effects of errors and to maximize the number of correct decisions in the system.

This work provides a distributed algorithm for bridge and articulation point detection in WMNs, capable of delivering the same or better accuracy of decisions as the proactive counterparts, but with considerable reduction in communication overhead. It is capable of operating in presence of message losses, accepting some decision errors but guaranteeing some key properties such as the termination of the detection process.

The major issues of the existing detection approaches in real WMNs teach us that idealization of network conditions, although a nice abstraction for theoretical analysis, can be severely punished in reality. The networks where the algorithm is to be deployed are dynamic, nodes fail and rejuvenate, environment changes, no messages exchanged in the network are safe from losses and the area where communication is possible does not have the circular shape.

All these real-life factors influence the biconnectivity testing algorithm, occasionally causing faults in it. Most of the fault causes are inherent properties of nature. They are experienced everywhere and they cannot be completely corrected in closed world of computer science by abstractions that are derived from it. Therefore, a comprehensive analysis of the impact of faults, how they are translated to errors in algorithm, and eventually to failures of detection process was performed.

The analysis of faults and errors provides the guidelines where and how to correct operation of the detection algorithm in order to reduce probability of erroneous decisions. The existing theoretical model had to be extended so that it can operate in reality. As already said, it cannot be expected that introduced changes, abstractions, and heuristics can eliminate all errors, but they have to improve quality of decisions sufficiently so that they can be used by network protocols, applications and services.

To summarize, the main contributions of this work are:

- A distributed protocol for bridge and articulation point detection that operates successfully under uncertainty, in presence of faults caused by inevitable physical laws of wireless communication. The detection algorithm fully utilizes benefits of broadcasting nature of wireless network adapters, so its message complexity is reduced to $O(n)$ from $O(e)$ (which is common in graph algorithms).

- A methodology for evaluation of accuracy of link-detection algorithms in presence of message losses. The methodology shows that the errors in link detection process are inevitable. They are particularly numerous if parameters of the link detection algorithm are improperly chosen. The methodology also serves as an efficient framework for parameterization of the link-detection algorithms so that the probability of errors in link detection is minimized.

- Measurement case study of community networks in Berlin and Leipzig, the analysis of existing topology generators, their comparison with measurements, and demonstration that their topologies do not correspond to reality. In particular, measurement study shows that bridges and articulation points are common in real networks.

- Development of a new node placement algorithm that is capable to create realistic topologies.

- The evaluation of the algorithm indicated the existence of strong locality within network topology. It was important to determine if it was just a stroke of luck in the evaluation or a more general property. Mathematical analysis confirmed the existence of an interesting characteristics of WMNs – if an alternative path between nodes incident to a link exists, its length tends to be very short. This property is utilized by the developed detection algorithm. It uses local searches and obtains comparable accuracy to the complete network searches, which is particularly useful in mobile networks.

The contributions of the thesis are not limited only to theory. If a solution of a problem is to be successful and applicable, it should be devised so that it reuses existing code and easily integrates with existing protocols. In the thesis, the implementation of biconnectivity testing algorithm has been integrated with AODV (Ad hoc On-demand Distance Vector) [137] routing protocol for its evaluation in the Jist/SWANS (Java in Simulation Time / Scalable Wireless Ad hoc Network Simulator). Integration with AODV brings benefit of further reduction of communication overhead – the algorithm cooperates with underlying protocol and reuses information that is already available in it. Finally, a sample application scenario is presented where decisions of the developed distributed biconnectivity testing algorithm are used to improve success rates of route discovery in a reactive routing protocol.

The key challenge encountered during the work was the necessity to step away from the convenience of certainty. Often it seemed effective and easy to apply the existing results known from the graph, multiprocessor or network theory to WMNs. After all, a WMN is also a network with independent nodes which communicate with each other and it can be abstracted as a graph. But again and again the uncertainty appeared, and every time it had profound effects on the existing approaches.

The effects of uncertainty were particularly expressed each time the task in question is to make a binary (dichotomous) decision: it is not possible to reach the consensus necessary for correct transfer of control flow in distributed algorithms; a trivial task of link existence detection in wired networks has turned into an error prone activity, loaded with uncertainty, state changes and conflicting outputs. A location-aware bridge detection protocol has been developed in [123], but it could not cope with unreliable messaging and irregularities in communication, so its development had to be stopped.

1.4. Structure of the Thesis

The structure of the thesis is shown in Figure 1.3. Chapter 2 provides general introduction to WMNs and describes how they are modeled in literature and in this thesis. Chapter 3 addresses existing work that was performed on bridge and articulation point detection, prevention of their occurrence through topology control, and protocols that are resilient to network partitioning.

Stochastic models for analysis of link-detection algorithms are developed in Chapter 4, for static and for mobile WMNs. The results show inevitability of errors in link detection. The probability of errors may be surprisingly high if parameters of link detection algorithm are improperly chosen, so the methodology is also used for HLD parameterization with the goal of error probability minimization.

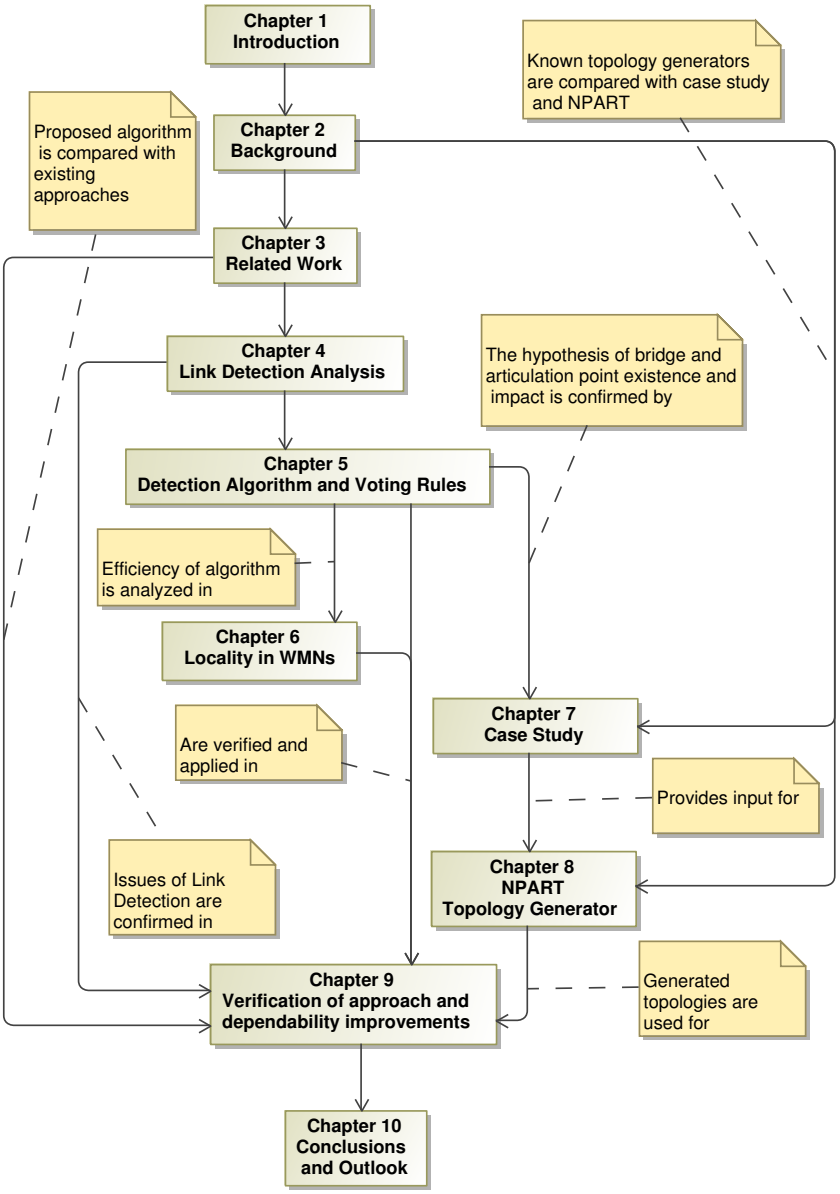


Figure 1.3.: Structure of the thesis.

Chapter 5 describes the developed approach to biconnectivity testing. The detection algorithm is based on the Echo algorithms [55][67], introducing changes that allow the execution in WMNs [120]. Its details are described in Sections 5.3 and 5.4. Faults in the system and failure modes of the algorithm are described in Section 5.5. The method for circumvention of errors, based on the voting theory, is presented in Section 5.6.

Chapter 6 explores locality of topology in order to evaluate performance of algorithm. A mathematical model is developed that provides upper limits for the average minimum cycle size in random geometry graphs as a function of network parameters. Based on the mathematical results, the localized version of the detection algorithm is proposed:

instead of searching the whole network, nodes search only in their neighborhood. It is confirmed in Section 9.5 that mathematically predicted local searches in random geometric graphs are valid and cause only a minor deterioration of detection accuracy if applied in WMNs.

Case study from large community wireless multi-hop networks in Berlin and Leipzig is presented in Chapter 7. It shows that bridges and articulation points are numerous in reality, giving plausibility to this work. It also demonstrates that well-known theoretical node placement algorithms cannot reproduce characteristics of real networks [119][121].

Chapter 8 describes Node Placement Algorithm for Realistic Topologies (NPART) that was developed in order to improve topological modeling of WMNs and quality of WMN simulations [122]. Comparison with the case study measurements shows that NPART topologies are similar to real topologies with respect to multiple properties.

Chapter 9 evaluates the proposed detection algorithm and demonstrates that it can be used to improve dependability in WMNs. Section 9.2 demonstrates that existing approaches based on the global topology knowledge cannot provide expected quality of decisions despite huge traffic overhead they impose on network. The behavior of the developed approach to distributed biconnectivity testing and voting rules is evaluated in two different environments:

- Section 9.3: approach is implemented and deployed in Motelab testbed [12]. Numerous experiments are performed to evaluate its accuracy.
- Sections 9.4 and 9.5: Approach is evaluated in the event-based simulator SWANS using different topology types, signal propagation models, for various algorithm parameters, in presence of node mobility, using localized searches.

The algorithm's accuracy in simulation is similar to that of experiments, showing that the proposed biconnectivity testing approach is not over-optimized for a certain environment. The algorithm is neither resource nor computation intensive as it was successfully run in the Motelab network whose nodes are equipped with 8 MHz processors and only 10 KB of operating memory.

The ultimate goal of biconnectivity testing is improvement of functional and non-functional properties of WMNs. Section 9.7 provides a sample application scenario where data from the biconnectivity testing algorithm is used to improve dependability of route searches in reactive routing protocols for WMNs.

Chapter 10 concludes the thesis. It summarizes the achieved results and discusses possible extensions of this work.

2. Background

This chapter provides an introduction to various topics that are used in the thesis. It describes models of WMNs (Section 2.1), how to abstract WMNs as graphs and random geometric graphs (Section 2.2). Section 2.3 provides overview of evaluation methodologies used in WMN research: simulators, emulators and testbeds. Section 2.4 describes the accuracy metrics used for evaluation of bridge and articulation detection algorithms.

2.1. Modeling of Wireless Multi-hop Networks

Simulation and theoretical studies of WMNs require detailed model as a starting point. Due to the complexity of WMNs we can distinguish six major sub-models:

- **Node model** describes node properties such as: type of radio, available energy source, memory capacity, processing capabilities, whether node knows its geographic location (GPS module), duty cycling, existence of nodes with extended capabilities ("super-nodes"), etc.

- **Node deployment and node mobility** models. Deployment (placement) models describe shape and properties of placement area, and distribution of nodes in the area. They provide node position in case of static networks or initial node positions for mobile networks. Some of the most popular deployment models are uniform and grid models. Mobility models describe movement patterns of nodes. Node movement creates dynamic network topologies – because of mobility, links between nodes are created and broken.

- **Radio model** defines the characteristics of the radio used by the node: its operating frequency, bandwidth, output power, reception thresholds, error correction, MAC layer functionality, energy consumption in idle mode, for packet reception and transmission, etc.

- **Wireless signal propagation model** describes the signal propagation through the air and influence of environment on its quality. In simulators this model is used to calculate the signal to noise and interference ratio (SNIR) at the receiver[1]. It is then typically assumed that if the SNIR is higher than some prescribed threshold (defined in the wireless radio model) the packet is successfully received. Frequently used models for the signal propagation in WMNs are path loss, two-ray ground, shadowing, and Rice/Rayleigh fading models.

[1]Bit error rate derived from SNIR is more precise but it is computationally too intensive for most of simulation studies, so SNIR threshold is preferred

- **Packet loss model**. The losses may be caused only by wireless channel properties and packet collisions on channel or additional packets can be dropped, for instance in accordance with uniform or Markov error models [70]. These models were mostly used to offset the effects of low packet losses created by the unrealistic path-loss-only models. With realistic wireless radio and signal propagation models the need for them is substantially reduced.

- **Traffic models** define which nodes send (sources) and which receive (destinations) traffic in the network as well as the properties of traffic flows. These properties are defined in layers three (transport) and four (application) of the TCP/IP stack. Usually, the transport protocol is either UDP or TCP. The application layer is usually modeled as a constant bit rate (CBR) flow or a file transport protocol (FTP) transfer. The application layer traffic also determines the used transport layer (CBR flows are associated with UDP, FTP with TCP).

The joint model is then built by selecting individual sub-models: e.g., a static wireless ad-hoc network is composed of 100 nodes placed uniformly on a square kilometer area; in addition to path loss, there exists Rayleigh fading on the wireless channel; nodes use ZigBee radios; nodes are unaware of their geographical locations; no additional packets at nodes are lost; there are 10 FTP flows transferring 10MB file between randomly selected source-destination pairs.

Some of listed sub-models are built from real data measurements (e.g., wireless signal propagation in telecommunication research, see [25] for a detailed survey). Other like topological and traffic models are synthetic or borrowed from wired network's research. There exists a notable lack of topological measurements in WMNs and researchers are forced to use artificial node placement models in simulation and emulation of their protocols.

The importance of real data sets for WMN research has been already noticed. The CRAWDAD site [6] is the WMN community archive of various measurements. However, in all CRAWDAD data sets observed network has either a single wireless hop or it was collected from a testbed, leaving the problem of topological analysis of real, user initiated networks open.

2.1.1. Node Placement Models

In order to build a connectivity graph of a WMN, it is needed to place the nodes in an area, to determine the existence of a link between each pair of nodes and its quality. There exists a number of different deterministic and random placement models that shape the connectivity graphs. Three sample topologies created by these models are shown in Figure 2.1. It is obvious that they differ substantially from one another. It is shown in Chapter 7 that they are different from real topologies as well.

In WMN research, the most frequent node placement models are uniform, chain and grid.

The chain placement model places nodes on a line. They may be located on equal or random distance. The grid placement model deterministically places nodes at intersections of a rectangular grid. Usually, the grid has quadratically shaped cells with cell edge length that is close to the communication radius of a node: e.g., distance between nodes is 200m and the communication radius is 250m. It creates networks that are regular in

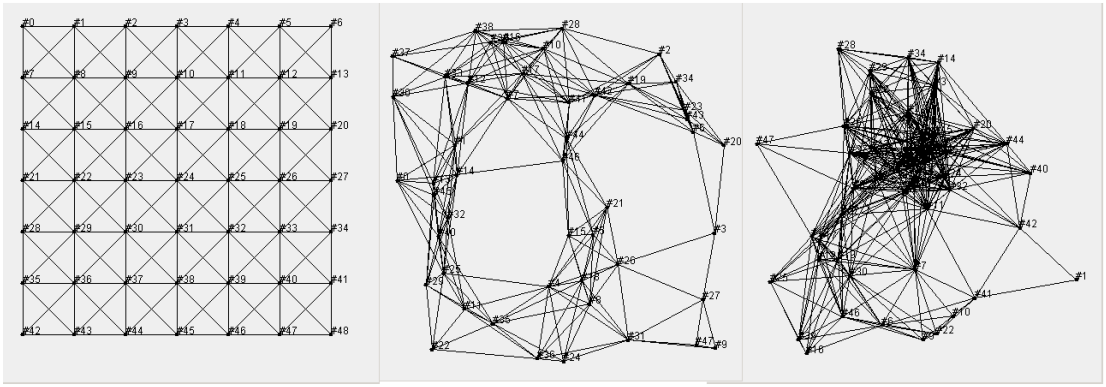


Figure 2.1.: Sample topologies created by grid (left), uniform (central) and RWM (right) models.

shape and provides excellent connectivity (there are no bridges nor articulation points in this model), good resilience to node and link failures (in particular in the central parts of the grid), and large set of disjoined paths between node pairs. Its structure limits interference and node degree, in particular if the grid parameters are chosen so that all nodes that are not on the grid border have a degree equal to four.

In the uniform placement model a placement area (rectangular or circular) of size $|A|$ is chosen and n nodes are placed inside of it with uniform probability $\lambda = \frac{n}{|A|}$. If placement area is rectangular $((0, x_{max}), (0, y_{max}))$, this is typically achieved by sampling x coordinate of a vertex from $U(0, x_{max})$[2] and y from $U(0, y_{max})$.

Analytical and simulation studies have proven that it is particularly difficult to create connected low-density topologies with existing randomized models. To ensure connectivity of simulated network, node density is increased. Bettstetter shows in [36] that in uniform placement model, nodes must have average degree of 10.8 to create a network that is connected with probability of 0.99. Li et al. [107] provide even higher estimation – they claim that obtaining of the same connectivity probability requires that network nodes have 13.78 neighbors on the average. Such dense networks have strong impact on simulation results since:

- For the same number of placed nodes, dense networks have smaller diameter than sparse networks.
- Numerous independent paths between each pair of nodes exist
- Failures of individual nodes do not impact the connectivity nor the functionality of the network

The need for improved node placement model in WMNs has been noticed and several non-homogeneous models have been proposed [40] [110] [134]. It is claimed that such inhomogeneous models improve quality of simulation results, and that they capture the reality, but there exists no experimental proof for such claims.

Bettstetter et al. [40] place nodes in accordance with the uniform homogeneous process and then apply thinning to it. The thinning operation removes nodes from a

[2] $U(a, b)$ is the uniform distribution.

network that have less than k neighbors within radius r (denoted as tr in [40]). The parameters k and r are specified by the user and they control the level of inhomogeneity of the topology. Bettstetter et al. also calculate several statistics for the obtained model, such as probability of nearest neighbor survival and distance to closest neighbor. However, authors of [40] discuss only the node placement, ignoring the properties and connectivity of topologies that can be obtained from it.

Liu and Haenggi [110] propose two quasiregular placement models. In the first, vertex coordinates are Gaussian distributed with the mean given by regular grid points. The second selects vertices from a uniform placement model such that every selected vertex is closest to a regular grid point. The obtained topologies resemble the grid structure but they are not as regular as grids. Various other variations and inhomogeneity models exist: in [71], two-dimensional Gaussian distribution is used to determine location of sensor nodes. This idea can be extended so that there are multiple vertex focal points, each of the focal points having a non-uniform distribution attached to it.

Onat and Stojmenovic [134] propose considerably different approach. They have developed several algorithms for creation of topologies that are connected with a high probability, and allow user to choose the average node degree. The shape of topologies primarily depends on the selected algorithm. The algorithms do not guarantee connectivity of their output - if the end result is not connected, the algorithm is restarted. Their analysis focuses on algorithm complexity and probability that created graph is connected. The probability density function for node degree for each of algorithms is presented and the differences among placement topologies created by different algorithms is informally (visually) demonstrated.

2.1.2. Mobility Models

Movement models change the connectivity graph over time: because of movement, node distance varies which in turn changes communication link quality, breaks and establishes them. Mobility may also alter the initial distribution of nodes.

One of the most frequently used movement models is RWM. In this model user defines the minimum (v_{min}) and maximum (v_{max}) allowed speeds of nodes, and the pause time between two movements. A node chooses a uniformly random point in the placement area and heads towards it with a speed selected from $U(v_{min}, v_{max})$. Once the destination is reached, the node waits for pause time and then repeats the whole process.

When this model was proposed, it was believed that RWM preserves uniform distribution and that the average speed of nodes is arithmetical mean of minimum and maximum speed $v_{avg} = \frac{(v_{min})+(v_{max})}{2}$. Both of these assumptions were false. The average speed is substantially lower than the arithmetic mean [177]. RWM also increases node density in central parts of the placement area and decreases it in vicinity of area borders. This occurs because nodes choose the straight-line shortest path to the selected destination and this shortest path tends to cross the central section of the placement area. The precise stationary node density probability distribution function can be found in [39]. These properties of RWM do not make it "good" or "bad", but they are different from the original assumptions.

Other movement models are not so ubiquitous as RWM and they are used to model specific situations and scenarios: In Reference Point Group Mobility (RPGM) model [168] nodes are divided in groups. Each group has a logical "center" and geographic

From/To	Home	Gathering place	Elsewhere
Home	-	0.8	0.2
Elsewhere	0.9	-	0.1

Table 2.1.: The parameters of the community mobility model [109]

scope. Trajectory of the group is determined by control of center's path. Group's (and center's) trajectory is defined by location, speed, direction, and acceleration. Nodes are uniformly distributed within the geographic scope of a group. Each node has a reference point which follows the group movement. A node is randomly placed in the neighborhood of its reference point at each step, allowing independent random motion behavior for each node within the group.

Group path is explicitly defined by an ordered sequence of check points and expected arrival times. As the group center arrives at a check point, it computes the new group velocity vector v from current and next check point locations and the required arrival time for the next check point.

General framework of RPGM can be specialized for more concrete scenarios. *In-place Model* divides the placement area into several adjacent regions, each having a group assigned to it. Nodes do not move out of the assigned regions and interact only at region boundaries. *Convention model* attempts to describe interaction between exhibitors and attendees on a fair. Several groups present their products in separate rooms. Rooms are connected and a group of attendees visits them, one by one. Attendees either stop in one room for a while (pause) or they pass through it quickly (no pause).

Community model [109] defines placement area that is divided into s subareas, so called "communities" and there exists a "gathering place". Each node has one home community. A node is more likely to visit its home community than other places. In each community, and at the gathering place, there is a fixed (non-mobile) node. Nodes select a destination and moves toward it with speed uniformly selected from (v_{min}, v_{max}). At the destination it makes a pause (user defined), and selects next destination and speed. If a node is at home, it goes with high probability to the gathering place. If it is away from home, it is very likely that it will return home. Table 2.1 shows transition probabilities that are defined in [109].

The authors argue that this model captures human mobility where the communities are for example villages, and sensor network applications for animal tracking – the gathering place may be a feeding ground, and the communities can be herd habitats or pastries.

Node placement and mobility models are used on their own or combined. For instance: Wei and Zakhor [171] use 7x7 grid to evaluate a multipath selection algorithm; Souryal and Moayeri [152] simulate a routing algorithm which adapts itself to link fading in 8x8 grid scenario, combined with uniform placement and RWM scenarios; Jansen et al. [93] have developed a proactive multipath distance-vector routing algorithm and evaluated it in 10x10 grid and RWM scenarios. Aad et al. [23] combine two placement models - a subset of nodes forms a static grid and a subset is moving within the grid in accordance with RWM. *Community* and *Inplace* movement models are used in [159] for evaluation of DTN protocols. Other examples of uniform, RWM and grid placement model usage can be found, for example, in [24] [57] [179].

2.1.3. Wireless Signal Propagation Models

Once nodes are placed, it is required to determine existence of links in the network, interference patterns, their quality, etc. Although the signal propagation and the packet loss are different phenomena, packet loss probabilities in simulation are calculated based on the wireless signal propagation models. Other effects that affect successful packet reception are ignored, unless the link layer is simulated (this is extremely rare in WMN research). For instance, a popular simulator ns2 [70], if used with shadowing propagation model, calculates SNIR ratio at the receiver at the start of a packet reception. If that value is higher than a threshold, the packet is received. The redundancy bits in the packet and individual bit errors are ignored. This is done in order to reduce computation complexity and simulation run-time. In real systems even if SNIR was high at the start the packet can be dropped due to SNIR decrease later on, or due to coding redundancy a packet can be successfully received even if some of its bits were garbled by noise.

If a sender transmits the signal of strength p_t and if received signal has strength p_r, the attenuation of the wireless channel is $a = \frac{p_t}{p_r}$. It is assumed that the packet is successfully received if the channel attenuation is less than some threshold attenuation a_t. The channel attenuation has different components and can be expressed as $a = a_{PL} + a_{SH} + a_{FA}[dB]$ where a_{PL} represents attenuation due to the path loss, a_{SH} due to the shadowing and a_{FA} is the attenuation due to the fading. Path loss is deterministic while shadowing and fading are stochastic processes.

In the path loss model the signal strength decreases with inter-node distance δ in proportion to $\frac{1}{\delta^\alpha}$ where α stands for path loss factor. Parameter α is two for vacuum and it is higher if there are obstacles between sender and receiver. At certain distance, the signal strength falls below the reception threshold a_t and the link is considered to be not functional. This breaking point is called the communication radius R.

If path loss is used exclusively as signal propagation model, a link exists with probability one if node distance is less than or equal to the communication radius R and with probability zero if the distance is larger than the communication radius. Link quality, assuming rather low traffic in the network, is close to one even for nodes on inter-node distances close to the R.

Measurement study [29] has shown that for fixed transmitter-receiver pair (constant distance, frequency, and transmission power) the received signal power is not deterministic but varies due to the objects in and around the signal path. Figure 2.2 shows the idealized path-loss model, where the signal strength is monotonously reduced as the distance to transmitter grows, and measurements inside a warehouse [88]. Because of the obstacles and reflection, signal strength is no longer monotonically reduced with increase in distance δ but forms a highly complex patterns.

The shadowing propagation model abstracts different phenomena affecting the wireless signal propagation that can increase or decrease signal strength: diffraction, reflection, self-interference, scattering, absorbtion. The shadowing variations (in dB) are given by the normal distribution with zero mean, path loss α and shadowing variance σ^2:

$$a_{SH}(\alpha, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{\alpha^2}{2\sigma^2}) \tag{2.1}$$

Consequence of Equation 2.1 is that communication does not have the Heaviside-

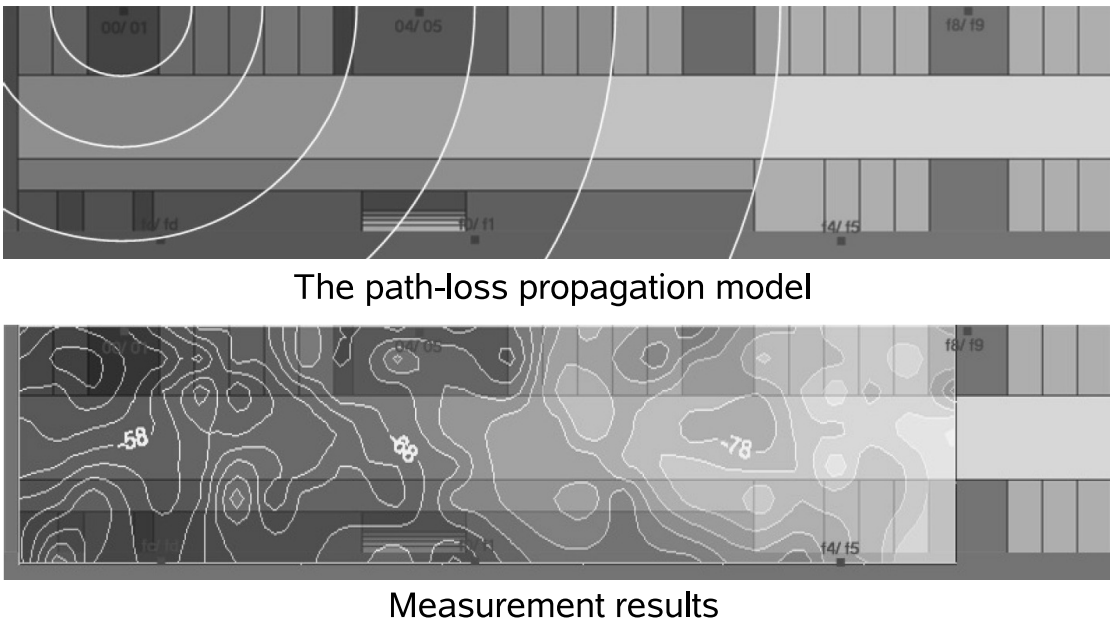


Figure 2.2.: Comparison of the mean signal strength predicted by the path-loss propagation model and measurement results [88].

function dependability on distance like in path-loss-only models – even if two nodes are close, they may experience problems on the communication link and even if they are far away, they may be able to communicate. For instance, a high and thick concrete building impacts all communication links that traverse it, substantially shortening the communication range of nodes, increasing packet loss or even disabling communication. Simultaneously, in proximity of same building large open space may exist, enabling excellent long-range communication over it.

The shadowing model from Equation 2.1 has its limitations as well. One of the most important is that it cannot correlate shadowing (all nodes that communicate through the building from previous example are affected by it). A correlated shadowing model that partially solves this issue has been proposed in the literature but it is intended for single sender-multiple receivers scenarios, not for WMNs.

The last component of attenuation is fading a_{FA}. The effects of fading are observed as rapid and hectic changes of the strength and phase of received signal. The signal variations are experienced in a fraction of a second or shorter. The fading is caused by the multipath signal propagation: receiver's antenna receives multiple copies of the transmitted signal, each having followed a different path. The objects (the scatterers) at and around the propagation paths are reflecting the transmitted radio signal (Figure 2.3).

Paths have different lengths. Because of the finite propagation speed of light, signals traveling along different paths are delayed differently at the receiver. Additionally, each signal copy has its attenuation, since it encounters different obstacles on its way.

At the receiver, multiple copies of a transmitted radio wave result in an interference pattern, where at certain points the waves interfere constructively while at other points they interfere destructively. If the propagation environment is absolutely static (trans-

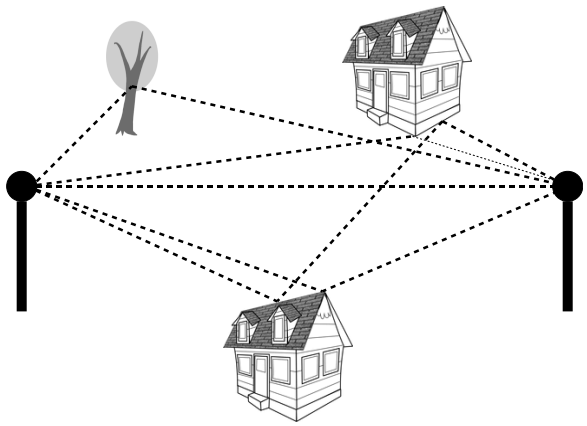


Figure 2.3.: Multipath signal propagation.

mitter, **all** scatterers, receiver), the interference stays constant and the channel is time invariant. If there exists any kind of movement in the environment, the propagation paths will be time variant, and as a consequence the wireless channel also becomes time variant.

Because of the large number of reflection paths and unknown attenuation coefficients of paths, it is not possible to derive deterministic expressions that fully describe fading. Instead, a statistical description is used as a way of characterization of the phenomenon.

If there exists no dominating path (no line of sight between transmitter and receiver), the Rayleigh fading model is to be used and the amplitude of the received signal varies according to Rayleigh distribution. The attenuation distribution is Ricean if there exists line of sight component of signal. The line of sight component dominates other components, its reception power is stronger than of the reflected paths and its delay shorter. Depending on the ratio between power of the direct path and power of the reflected paths, the Rice distribution can model different line-of-sight scenarios.

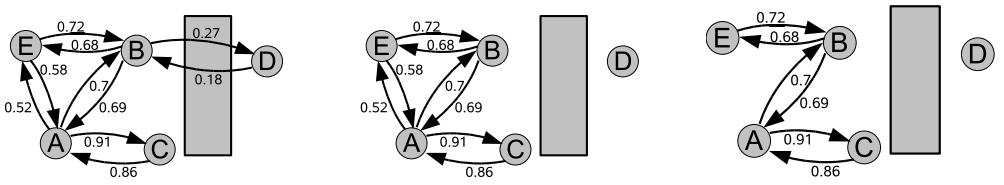
More details and precise mathematical models of signal propagation may be found in [25][51][144].

2.1.4. Medium Access Control Sublayer

Medium access control provides channel access control mechanisms and node addressing. Each network adapter has a unique (within a network or globally) physical address. Addressing allows data packet delivery to targeted destinations. Medium access control is used to avoid and/or detect packet collisions packet-contention channel is used (e.g., CSMA/CA) or to use only the channel resources which are reserved for the network adapter (e.g., sending of data only within the assigned time slot in TDMA scheme).

For implementation of the proposed biconnectivity testing algorithm it is important that MAC layer supports two different means of communication:

- MAC Broadcast: packet is intended for all nodes that receive it. The recipient does not send an acknowledgement on reception of a broadcast packet.
- MAC Unicast(or just unicast): packet is intended for a single recipient. Upon reception of a unicast packet, recipient must acknowledge it. If sender does not



(a) Example of a connectivity graph, $t = 0$.

(b) Example of a connectivity graph, $t = 0.3$.

(c) Example of a connectivity graph, $t = 0.6$.

Figure 2.4.: Graph models of a wireless network.

receive the acknowledgement, the packet is sent again. If recipient receives multiple copies of the same packet, redundant copies of a packet are dropped but acknowledgements are sent. Sender initiates transmission up to *retry* times before it drops a packet.

Unicast messages may be received by multiple nodes because of the broadcasting nature of wireless medium, but the network adapter filters them out by receiver address. Some protocols (e.g., 802.11) use higher signaling speed for unicasts than for broadcasts, making unicast packets more susceptible to losses.

If a message is intended for all nodes in the network, MAC layer functionality is no longer sufficient and the network-wide broadcasts (or just broadcast) is used. In large wireless networks recipients are not directly reachable over MAC layer so a higher layer protocol is used for multi-hop broadcasting. One of the most common network-wide broadcasting algorithms is the flooding: every node that receives a message intended for broadcasting, sends it once using the MAC Broadcast and ignores the subsequent receptions of the same message. In order to reduce packet collisions on the air node randomly decides how long should it wait before starting the MAC broadcast.

2.2. Graph as a Wireless Multi-hop Network Model

The complexity of individual WMN sub-models and their interplay in the composite WMN model impose insuperable difficulties for analytical solvers. In order to contemplate on general properties of network behavior, precision of the composite model is sacrificed and a model of higher abstraction level is introduced.

Graphs are commonest mathematical models of communication networks. Network nodes are represented as vertices and communication links as arcs. In the thesis the weighted graphs are used to model WMNs. If node A is able to communicate with node B there exists an edge AB in the communication graph. Edge may be annotated with weights. The weights can represent throughput, bandwidth, distance of nodes, delay, quality of a link, etc. Unless specified differently, the weight w_{AB} represents the link quality. Link quality w_{AB} is the probability that a packet successfully traverses the link in direction from A to B. Edge also has the weight w_{BA} assigned to it to describe the probability of traversal in direction from B to A. Weights need not be the same (as shown in Figure 2.4), but they must be larger than zero (no unidirectional links are allowed in the model). A pair of arcs between two nodes forms an edge.

ETX	10	30	50	100
Link quality	0.312	0.182	0.141	0.1
Correct reception (U)	0.952	0.8	0.703	0.569

Table 2.2.: Relation between ETX metric, link quality and probability of successful packet reception if MAC unicast (U) (1+7 retries) is used.

2.2.1. What is Topology of a Wireless Multi-hop Network?

Intuitively, the transformation from a real instance of a WMN to a graph is a bijection: every network node is mapped to a graph vertex and every communication link to an edge in the graph.

Such mapping is natural in wired networks – cabling provides the needed information on link existence. In a wireless network the link existence is not so clearly defined.

In a wireless network, whether two nodes are capable of communicating and what is the quality of their communication depends on node characteristics, network traffic, and environmental factors. Signal attenuation, noise on channel and at the receiver radio, and contention on the shared communication medium cause that some packets are received and some lost.

In such an environment a metric is needed that defines which node pairs can communicate in an *acceptable* manner. What is *acceptable* is application dependant – applications have different requirements with regard to allowed packets losses, packet delay and jitter.

In this work, the probability of packet loss/successful reception is chosen as the metric of link acceptance. The packet loss also strongly relates to other application-relevant metrics. Packet losses lead to communication retries. Most of the wireless MAC layers use exponential backoff between successive retries, so with each successive loss the packet experiences longer delay. Since the length of loss sequence is stochastic, consecutive packets at receiver may be delayed differently, thus the packet losses also influence the jitter.

The packet loss probability (equivalently the probability of successful packet reception) between nodes A and B is often different from loss probability in direction B to A. A metric that joins these probabilities in a single value, the so-called ETX, was proposed in [64].

Definition 2.1 *The estimated number of packet retransmissions (ETX) for wireless link between nodes A and B is calculated as* $ETX = \frac{1}{w_{AB} \cdot w_{BA}}$.

Table 2.2 illustrates dependency between ETX value and a successful transmission of a packet if MAC broadcast or unicast is used. The values in the table are approximative since they are calculated under assumption that packet loss probability is symmetric ($w_{AB} = w_{BA} = p_{loss} = \frac{1}{\sqrt{ETX}}$).

It is not possible to determine one link acceptance threshold that suffices to all applications and network services. Obviously, applications prefer higher link quality, but some of them also accept (may function without major issues) lower link quality. We leave the selection of the link acceptance threshold to user. In context of bridge and articulation point detection (in particular, during the evaluation in Chapter 9) the proposed approach is evaluated with different thresholds, to demonstrate that its accuracy is almost independent of the threshold value.

Figure 2.4(a) shows the graph of a WMN with all its links. Its shape would remain the same if the link acceptance threshold t is below 0.18. If the acceptance threshold t is set to 0.3 some of the links in the graph are eliminated and the resulting topology can be seen in Figure 2.4(b). Number of edges in topology is further reduced if the link acceptance threshold increases to 0.6 (Figure 2.4(c)).

2.2.2. Graph Theory Basics

In this section are defined some of the important notions that are used later in the text [173]. The terms "connectivity graph" and "network", "vertex" and "node", "edge" and "link" are used interchangeably so the definitions from text apply to graphs and networks equally.

Definition 2.2 *In communication graph $G(V, E)$ physical nodes are represented as vertices $V(G)$ and communication links as edges $E(G)$.*

Definition 2.3 *Walk of length k is a sequence $v_0, e_1, e_2, ..., e_k, v_k$ of vertices and edges such that $e_i = v_{i-1}v_i$ for all i. A trail is a walk with no repeated edge. A path is a walk with no repeated vertex. A u, v walk is a walk with first vertex u and last vertex v.*

Definition 2.4 *Cycle is ordered list of vertices $v_1, ..., v_n$ such that $v_{i-1}v_i$ and v_1v_n are edges in G.*

Definition 2.5 *Graph $G(V, E)$ is connected if for each pair of vertices a and b there exists a path (a, b). Otherwise the graph is disconnected. The components of a graph are its maximally connected subgraphs.*

Definition 2.6 *Two vertices a and b are adjacent if there exists an edge between them ($ab \in E(G)$). If vertex a belongs to an edge e, e and a are incident.*

Definition 2.7 *The degree of a vertex v in a graph G, written $d_G(v)$ or $d(v)$ is the number of edges incident on v.*

Definition 2.8 *A pendant vertex is vertex of degree* 1.

Definition 2.9 *A bridge in a graph is an edge whose deletion increases the number of components. An articulation point in a graph is a vertex whose deletion increases the number of components in the graph.*

It is clear from Definition 2.9 that vertices incident to a bridge are also articulation points if they are not pendant vertices. If a pendant vertex is removed from G, the removal does not affect other nodes in the network – the number of connected components in the graph does not change.

Lemma 2.1 *An edge $e = xy$ in graph G is a bridge if and only if $G - e$ has no x, y path.[173]*

Corollary *An edge in a undirected graph G is a bridge if and only if it does not belong to a cycle.[173]*

Let us assume the opposite: "an edge belongs to a cycle if and only if it is not a bridge." It is clear that $e = xy$ belongs to a cycle if and only if $G - e$ has an x, y path, which by Lemma 2.1 is true if and only if e is not a cut edge.$\diamondsuit$

This corollary will be extensively used in Chapter 5, where the edges are checked whether they belong to a cycle, and if not, they are declared as bridges.

Definition 2.10 *Let the expression $d(a, b)$ denote the length of the shortest path between two vertices in a graph. Then the diameter of a graph G is $max_{u,v \in V(G)} d(u, v)$.*

Definition 2.11 *Length of a cycle C is number of edges that compose it. Shortest cycle over an edge is cycle $C_s(e)$ whose length is minimal of all possible cycles in which the edge can participate.*

Definition 2.12 *A graph without cycles is acyclic. Tree is a connected acyclic graph.*

If a tree is constructed in a cyclic graph, edges that are not included in the tree are called cross-edges.

Definition 2.13 *A spanning subgraph G_s of G has all vertices as G. Spanning tree is a spanning subgraph that is also a tree.*

Definition 2.14 *Level of a vertex in a tree is its distance from the root.*

Definition 2.15 *In a tree T, a common ancestor of nodes A and B is every vertex which descendants are both A and B.*

Definition 2.16 *Highest common ancestor (HCA) of two vertices in a tree is vertex HCA_{AB} that has highest level of all their common ancestors.*

2.2.3. Random Geometry Graphs

In graph theory, we are often interested in behavior of certain properties on a whole class of graphs, without predefining concrete graphs. The random graph theory is introduced to avoid determinism of studies on predefined graphs.

Definition 2.17 *Random graph is a graph created by a random process.*

Existence of vertices and edges in a random graph is defined by a random process. Thanks to this randomness, instances of random graphs can be considerably different. Simultaneously, they have common properties (e.g., probability distribution of edge existence), allowing analytical and simulation study of their characteristics. One of the most popular random graph models is the Erdos-Renyi graph, where each of the $\binom{n}{2}$ possible edges exists with a constant probability p.

Such models, where edge existence is predefined and does not depend on node location are inappropriate for modeling and analysis of wireless networks. Instead, random geometry graphs are used.

Definition 2.18 *Random geometric graph is created by random placing of vertices in a predefined finite or infinite space. An edge exists between two vertices only if their Euclidean distance is less than a predefined threshold R.*

This model is not ideal: it does not account for all properties of wireless communication such as signal shadowing or interference. Still, it is more precise and applicable than pure random graphs.

The Definition 2.18 does not specify the random process used in graph creation, so further refinement and specialization to a specific class of random geometric graphs is required. Usual approach in WMN research is to observe a set of vertices uniformly distributed in two dimensions in a bounded rectangular or an unbounded area. Stationarity and homogeneity of the placement process are also assumed.

Uniform point distribution of nodes over an area can be seen as an approximation of the homogeneous Poisson point process and vice versa. In the uniform node distribution, the number of nodes in the placement area has always the same value n. In the Poisson point process, the number of nodes in placement area is distributed in accordance with the Poisson distribution with a mean value n. For large n variation around it is relatively small so the results obtained for Poisson point process are applicable to the uniform node distribution scenarios.

Detailed theory of Poisson point processes can be found in [62]. The theory of Poisson point processes is generalized to d-dimensional spaces but of our interest are only two dimensional processes.

In text, the following notation is used: The intensity of the point process is marked as λ, observed area in the plane is marked as A, size of the area A is $|A|$, expected number of nodes n in the observed area is $n = \lambda|A|$, the Euclidean distance of two nodes A and B is $\delta(A, B)$.

A graph has e edges and $c(G)$ components. If a graph is planar it has f faces and a face has size s (s edges are forming the face). During planarization process, a set of edges e_d has been removed (deleted) from the starting graph.

2.2.4. Planarization of Random Geometric Graphs

It has been noted in wireless networks that dense graphs experience communication issues (e.g., due to interference on wireless channel) and unnecessary energy expenditure. To tackle the problem, topology control techniques are used, simplifying the connectivity graph. A set of edges is logically removed from it if they will not be used for communication although the nodes can physically communicate. It is claimed that such simplification of connectivity graph can improve properties of the network (e.g., scalability [95], energy consumption [50] or even reliability of communication [44]).

One of approaches used for topology control is planarization of the connectivity graph. Gabriel Graphs (GG) and Relative Neighborhood Graphs (RNG) are particularly practical for the planarization because the algorithms are distributed and they preserve the graph connectivity.

Definition 2.19 *A drawing of a graph $G(V, E)$ maps vertices V to points in a plane and each edge AB in polynomial curve A, B. A polynomial curve U, V is union of finite number of line segments that share only their endpoints and that starts at U and ends*

in V. A graph is planar if no two polynomial curves intersect, except at vertices of the graph.

Definition 2.20 *An open set in the plane is a set $U \in R^2$ such that for every $P \in U$, all points within some small distance from P belong to U. A region is an open set U that contains a polygonal U, V path for every pair $U, V \in U$. Face of a planar graph is the maximal region of the plane that is disjoint from the drawing.*

Gabriel and Relative Neighborhood Graphs are created from a graph by deciding whether an edge is removed from the graph G. The following rules are used:

Definition 2.21 *An edge AB exists in Relative Neighborhood Graph if the distance between the nodes $\delta(A, B)$ is less than or equal to the distance between every other node W in the graph and the further of the nodes A and B:*

$$\forall W \neq A, B : \delta(A, B) \leq max(\delta(A, W), \delta(B, W)).$$

Equivalent claim is that if there is at least one node (a witness node) W in the shaded lens shaped area in Figure 2.5(a), the edge AB is eliminated from the starting graph, otherwise it remains in the Relative Neighborhood Graph.

Definition 2.22 *An edge AB exists in Gabriel Graph if there is no witness node W in the circle whose diameter is $\delta(A, B)$ and to whom both nodes A and B belong to:*

$$\forall W \neq A, B : \delta^2(A, B) < \delta^2(A, W) + \delta^2(B, W).$$

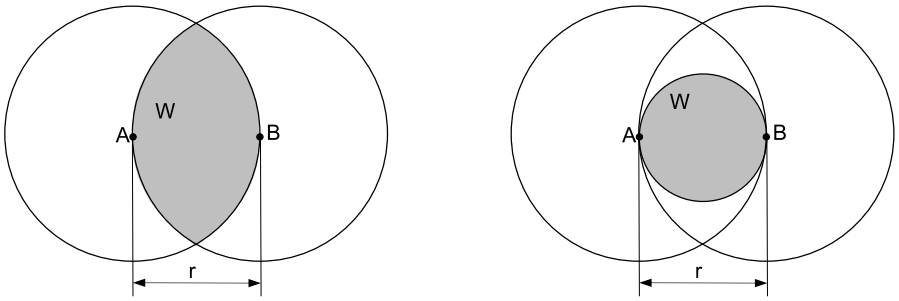
In Figure 2.5(b) the edge AB will be removed if there is at least one witness node W in the shaded circle.

GG and RNG contain the minimum spanning tree (MST) of the starting graph so they guarantee preservation of the connectivity. In case that the starting graph is disconnected, the resulting graph remains disconnected but the number of graph components does not change. It has been proven in [163] and [132] that MST$\subset$RNG$\subset$GG. This also means that $|MST| \leq |RNG| \leq |GG|$ and consequently $n-1 \leq |RNG| \leq |GG| \leq 3n-6$ (the lower bound represents the MST size and the upper bound is the maximal number of edges in a planar graph).

In survey [94], Jaromczyk and Toussaint list graph properties, bounds on the graph size as well as algorithms for construction of proximity graphs (GG and RNG are proximity graphs). Devroye in [66] provides asymptotic lower bounds on graph size independent of the node placement density λ. However, in [66] the existence of a link in a graph depends solely on the presence/absence of witness nodes. That is as if setting threshold R from Definition 2.18 to infinity, which is beyond acceptable approximation for WMN study.

In estimation of cycle sizes in random geometric graphs (Section 6.1), the following definitions and notational conventions are used:

Definition 2.23 *Circular segment is a portion of a disk whose upper boundary is a (circular) arc and whose lower boundary is a chord making a central angle $\theta < \pi$. In text it is marked as $\frown$.*



(a) Relative neighborhood graph. (b) Gabriel graph.

Figure 2.5.: Witness area and construction of neighborhood graphs.

Definition 2.24 *Circular sector is obtained by taking a portion of a disk with central angle $\theta < \pi$. In text it is marked as $\bigtriangledown$.*

Definition 2.25 *Lens is union of two identical circular segments that share the same chord. In text it is marked as $\Diamond$.*

The area of a circular segment is:

$$A_{\frown}(r) = R^2 \arccos \frac{r}{R} - r\sqrt{R^2 - r^2}. \tag{2.2}$$

where r is the distance of circle center to the chord. The area of lens is twice the area of a circular segment. The area of the circular sector is trivially calculated as $A_{\bigtriangledown}(\theta) = \frac{1}{2}R^2\theta$.

Traditionally, RNG and GG have been used in areas such as computational morphology, spatial analysis or pattern classification. Recently, they are applied to the domain of wireless communication and wireless ad hoc networks. Both are used for various applications, but due to its higher density (as derived later in Section 6.1), Gabriel Graphs are preferred to Relative Neighborhood Graphs.

Cartigny et al. in [50] use RNG and its subgraph LMST (local minimum spanning tree) to provide energy efficient broadcast in ad hoc networks. Wang et al. in [169] use Gabriel, Relative Neighborhood, Yao and Delaunay triangulation graphs for efficient Bluetooth scatternet formation. Johansson and Carr-Motyčková [95] use a variation of Gabriel graph to reduce the interference in ad hoc networks.

Bose et al. [44] and Karp and Kung [98] proposed usage of planarized graphs for routing in location aware WMNs. Both routing protocols use greedy strategy to approach the destination as fast as possible and switch to routing along faces of planarized graph in order to avoid local minimums. In [44] is proved that this strategy ensures message delivery. Kuhn et al. [103] propose geometrical routing algorithm GOAFR+ which uses Gabriel graphs and prove that it is asymptotically optimal in the worst case.

These studies claim that planarization may provide desirable properties, but they are incomplete: errors introduced by localization equipment are not discussed, nor effects of shadowing and fading on their properties (e.g., greedy geometrical protocols prefer longest links that usually have poor quality and high probability of packet losses).

2.3. Simulators, Emulators and Testbeds - Their Benefits and Drawbacks

Usually it is not possible to analytically solve detailed models presented in Section 2.1. Even if simplified graph models are used, complexity of their solving may be too high (e.g., exponential complexity for calculation of partition distribution from Appendix A) and for complex protocols no analytical solving methods are known. Instead, protocols are tested in simulators, emulators and testbeds.

Network simulators, such as ns2 [70], OMNET++ [15], SWANS [19], TOSSIM [106] are favored for their fast prototyping and cost efficiency so they are preferred to emulators and testbeds in early stages of protocol implementation.

An additional and important benefit of simulations is their repeatability: same simulator engine and the identical simulation setup must lead to the same outcome (within margins of statistical error). The repeatability is scientifically important since it enables verification of the algorithm by independent researchers and fair comparison with other approaches.

Simulators require careful modeling of the end system if realistic and meaningful results are to be obtained. The worst that can happen is to come to a conclusion that a protocol is good enough to be implemented and deployed in real systems, just to find out that important aspects have been omitted within the simulation study and that protocol is actually unusable.

The discrepancy between model and reality can be very subtle as noticed in an IBM study of Mote nodes [160]. The IEEE 802.15.4 has been implemented on Mote nodes. It was planned to build a WMN for logistics management out of them. Testing revealed that the Mote nodes do not have sufficient processing power to timely respond to all interrupts created by MAC layer. As soon as the traffic in network grows above certain limit (which was very low - one packet per node per second), the nodes cannot operate MAC layer and the network collapses. Such behavior is untraceable in most of simulators which do not account for processing speed of individual nodes.

Emulators are at the border-line between simulators and reality. Nodes in emulator are real, they can run real software instead of simulation code, protocols built for them are deployable to real networks without or with minor modifications. However, the packets are not sent but packet delivery is calculated, like in simulation. Emulators can be large and flexible - for instance, the ORBIT emulator [16] consists of an indoor radio grid emulator consisting of 400 nodes in a 20x20 grid. The grid can be dynamically configured to form differently shaped topologies.

Network testbeds are the best choice for protocol testing because there are no hidden assumptions in them like in simulators and emulators. They are widely used for MAC, networking/routing or transport protocols verification. However, testbeds are limited spatially and in node count. WLAN based testbed do not scale to hundreds of nodes like real networks: MIT Roofnet has between 30 and 40 active nodes [3]; wireless mesh testbed at University of California in Santa Barbara [4] has 25 nodes in one building distributed over five floors; Kotz et al. [100] created a 33 node mobile testbed at University of Darmouth. Wireless sensor testbeds like Motelab [12] or TWIST [84] have more nodes but due to limited communication capabilities of sensor nodes, they are spatially limited to a single building.

Data sampling in testbeds is easy. The complete control of nodes allows detailed logging of events of interest and their forwarding to a central collection server. Still, testbeds are not universally applicable. For instance, if considered for analysis of topological properties, testbeds have several serious drawbacks:

- Network topology in testbeds does not originate from user behavior but from designers assumptions such as initial node placement. It may or may not follow the real user behavior.
- Limited size of a testbed may restrict it to a single type of propagation environment (e.g., an office building in [4]) that in turn creates topologies specific for that type of propagation environment.
- In order to reduce the maintenance effort and costs, it is common to use the same equipment for the whole testbed. In real networks, with the growth of a network grows the diversity of propagation environment and of the equipment deployed in it.

Other measurements in testbeds can be influenced, but to a smaller degree. For instance, traffic patterns and testbed's topology are related with interference patterns in the testbed, providing partially synthetic link quality data.

2.4. Accuracy Metrics of Biconnectivity Testing Algorithm

If a correct algorithm is executed in a deterministic environment with accurate system state knowledge it performs its task correctly (if there are no errors in its implementation). If only a partial system state information is known, if the knowledge is not completely accurate, or if heuristics are used to improve performance of an algorithm, its output may be incorrect. In context of this work, communication and node faults in WMNs introduce errors in the biconnectivity testing algorithms, so the algorithms make erroneous decisions.

If an event occurs and an algorithm decides that the event has occurred, such composite event is a *true positive*; if the algorithm decided that event did not happen, but it happened, the composite event is *false negative*, etc. The confusion matrix in Table 2.3 lists all four possibilities.

An algorithm with correct output does not produce false positives and false negatives. If they are produced (for any of the previously listed reasons), they should be few. The absolute count of events from the confusion table is impractical for comparison of different algorithms since they may be tested on different number of samples.

	actual positive	actual negative
decided positive	true positive TP	false positive FP
decided negative	false negative FN	true negative TN

Table 2.3.: Confusion matrix.

Instead, precision and recall are used as accuracy metrics. Precision and recall were originally introduced for evaluation of performance (quality) of machine learning algorithms. Later, they have been used for evaluation of algorithms in failure modeling. In the thesis they are used for accuracy evaluation of the bridge and articulation point detection algorithms.

Precision is the ratio between the number of correct decisions and all decisions:

$$precision = \frac{TP}{TP + FP}. \tag{2.3}$$

Precision is the probability of correct marking of an observed event. For instance, if an algorithm decides that an event has occurred ten times and its decision was correct in nine cases, its precision is 0.9. However, precision does not capture the ratio of successfully identified events. The recall represents the probability of identification of event of interest:

$$recall = \frac{TP}{TP + FN}. \tag{2.4}$$

For the previous example, if out of 30 events only nine were correctly identified, the algorithm has recall of $9/30 = 0.3$.

These two measures are typically observed together since each of them alone is not sufficient to describe the efficiency of a decision algorithm. For instance, a bridge detection algorithm which decides that every edge in a graph is a bridge has a recall rate of 1 (perfect) but its precision is equal to share of bridges in the graph, which can be very low. Other extreme case would be a very conservative algorithm which decides that an edge is a bridge only when the edge is incident to a pendant node – precision of such algorithm is 1 (perfect) but it fails to recognize all other bridges in the graph, thus having a small recall. Both values can be easily depicted on precision-recall plots: the x axis is precision of the algorithm and the y axis is its recall. A correct algorithm, that does not make errors, is located at (1,1) in the precision-recall plot.

The F_α measure is the weighted mean of precision and recall and it is used to transform precision and recall into one numerical value:

$$F_\alpha = \frac{(1 + \alpha) \cdot precision \cdot recall}{\alpha \cdot precision + recall}. \tag{2.5}$$

The parameter α defines the weight in favor of one or other metric. For instance, $F_{\frac{1}{3}}$ weights recall three times more as precision and F_2 weights precision twice as much as recall. The user of the metric must decide on value of parameter α. In some scenarios it is difficult to exactly determine relevance of metrics and equal importance is assigned to both. If both values are weighted equally ($\alpha = 1$), the traditional F-measure or F_1 measure is obtained:

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall}. \tag{2.6}$$

In context of bridge and articulation point detection and its applications, it is also important to capture the relative location of a cut-edge or -vertex within the graph. A bridge incident to a pendant node is usually less important than a bridge that divides a network in half.

To illustrate the importance of bridge position in a network, let us assume that there exists a single bridge in a network, forming 2 subgraphs connected over it. Each of the subgraphs is either at least 2-connected or consists of a single node. If one of the subgraphs consists of k nodes, the other has $n-k$ nodes.

In such a network, number of node pairs that avoid bridge traversal during communication is $\frac{k(k-1)}{2}+\frac{(n-k)(n-k-1)}{2}$. Number of node pairs which have to use the bridge while communicating is $k(n-k)$. The probability that two nodes selected randomly from the network will have the bridge as a part of their communication path is (probability of **b**ridge **t**raversal P_{bt}): $P_{bt}(k)=\frac{2k(n-k)}{n(n-1)}$.

If the bridge is placed at the border of the network, connecting a single node with the rest of the network, its impact will be negligible $P_{bt}(1)=\frac{2}{n}$. If a bridge divides network in two equal parts, it creates the maximum number of paths traversing the bridge: $P_{bt}(\frac{n}{2})=\frac{n}{2(n-1)}\sim\frac{1}{2}$. This can seriously impact the global functionality of the network – even with moderate traffic, the centrally placed bridge is likely to be heavily congested. Not only that this congestion reduces the available throughput per flow, but it can compromise other properties: for instance due to the long waiting times of packets to traverse a bridge, delivery deadlines for the real time traffic may not be met.

In order to quantify the relative position of a bridge/an articulation point in network, the reward metric is used. Reward is defined as the number of paths that must use the bridge or articulation point.

The reward assigned to a bridge/an articulation point is calculated in the following way. First, the total number of paths that are possible in a graph is calculated. Then, a bridge or an articulation point is removed from the graph, and the number of all possible paths is calculated for the new graph. The **reward** is the absolute difference of the two path-counts.

Depending on the edge/vertex location the metric takes different values. For instance, in a connected network of n nodes, a bridge incident to a pendant node has reward of $\frac{n(n-1)}{2}-(\frac{(n-1)(n-2)}{2}+1)=n-2$. If another bridge divides network into two equal parts, the reward for its detection is $\frac{n(n-1)}{2}-2\frac{\frac{n}{2}(\frac{n}{2}-1)}{2}=\frac{n^2}{4}$. The metric is expressive and vividly differentiates between peripheral and central bridges/articulation points. If network from the example has 100 nodes, the reward for detection of a bridge incident to a pendant is 98, while the centrally placed bridge has reward of 2500.

For a given simulation setup, the average value of reward is calculated over all decisions that are made on all bridges/articulation points. For example, if biconnectivity testing algorithm was executed three times in network described in the former example, so that the peripheral bridge was detected all three times, the central only twice, the average reward equals to $\frac{3\cdot 98+2\cdot 2500}{3\cdot 2}=882.33$.

3. Related Work

Biconnectivity testing is a well-known and researched problem in graph and algorithm theory. The simplest mean of testing is the systematic iteration through all edges and vertices, their individual removal and counting of graph components. If the number of components increases, the tested edge or vertex is critical for connectivity. Although simple to understand and implement such testing is inefficient due to its high run-time complexity.

Tarjan [158] has shown that the extended Depth First Search (DFS) algorithm accurately detects cut edges and vertices in $O(n + e)$ time under the assumption that the graph structure is known at a single computation node. The algorithm has been altered afterwards for use in parallel computing environments, where its run-time complexity can be further reduced. Due to these favorable properties of DFS it is one of the commonest methods for articulation point and bridge detection.

BFS search for detection of biconnected components and cycles has been mostly used in parallel computing. Liang and Rhee [108] provide BFS algorithm for biconnectivity testing, but their algorithm is applicable only to permutation graphs and cannot be generalized. Barnat [32] uses BFS for state space generation and controlled DFS for cycle detection in process of linear temporal model checking. Thurimella in [162] provides a distributed BFS based algorithm for biconnectivity testing, however the algorithm cannot be adopted for application in WMNs since the optimization goals and starting assumptions are completely different: each processor can communicate directly (over shared memory or a bus) with all other processors and each processor knows the graph topology. Such processing model allows state synchronization between processors that are necessary for this algorithm. The algorithm also requires total preordering of node visits which does not fit with the broadcasting nature of wireless medium where all neighbors of a node are visited simultaneously.

The existing algorithms are well suited for application in wired telecommunication and computer networks. In such networks, communication links are rather stable and their status (existence) can be accurately determined without[1] or with low communication overhead relative to total link communication capacity. Additionally, communication links are decoupled, allowing extensive, message-based testing of a link without influence to other communication links in its vicinity. Thus, the data on link status is highly accurate. Due to properties of wired networks, nodes in the network can safely assume that the topology they are aware of is accurate. Then, a node simply applies one of the known biconnectivity testing algorithms to the collected topology data.

[1] Using carrier sensing.

3.1. Biconnectivity Testing in Wireless Multi-hop Networks

Good performance of biconnectivity testing based on the global topology knowledge in wired communication networks has inspired adoption of the same approach in the WMNs. Common assumption is that a proactive routing protocol is executed in a network, and that it provides accurate topology knowledge to all nodes in the network. Cut-edges and vertices are then detected, usually by Tarjan's DFS algorithm. Some authors additionally attempt to predict behavior of nodes in the network and to perform corrective actions that, for instance, prevent partitioning of the network.

Goyal and Caffery [80] offer a location aware method to bridge detection. All nodes are location aware and periodically update their neighbors with current locations. The algorithm uses depth first search (DFS) to find the cut-edges based on the global topology knowledge. After detection of a potential partitioning, the algorithm offers prevention mechanisms, either by changing the trajectories of critical nodes, or by adding a helper node to reinforce the link.

Hauspie et al. [85] use global topology knowledge to construct all disjoint paths between a source and a destination node. For each path in the network, robustness is calculated based on probabilities that some of the links constituting the path will break. They offer two metrics: first is a function of number of disjoint paths. It says that if there is only one disjoint path between two nodes, it is highly probable that a graph will become disconnected if some link on that path breaks. The second metric states that the longer the path is, the weaker it is in the sense of robustness, meaning that all paths do not contribute evenly to the robustness. The problem with this approach, however, is that it does not guarantee correctness, because the network can get partitioned without this algorithm detecting it. The second problem is that it requires global knowledge of the network topology in order to generate all disjoint paths and calculate link robustness. The benefit is that it does not require any information about positions of mobile nodes.

Wang and Li [168] claim that by knowing only local node positions and based on the individual node mobility model, global scale topology changes such as network partitions cannot be predicted nor prevented. They introduce a model for group mobility. Instead of grouping nodes by location, they employ a clustering algorithm that groups nodes by their velocity vectors. A group is then characterized by mean group velocity. They argue that clustering by velocities provides a clearer characterization and separation of mobility groups. Separation of groups leads to the network partition. A central server must exist to which all nodes report their positions and speeds. The server runs clustering algorithm and attempts to predict network partitioning. Thus, the server is the single point of failure of the approach. Another issue are assumptions that velocities of mobility groups and their composition are time invariant, the identical circular shape of all groups and the circular communication radius.

Chen et al. [57] propose an algorithm that determines whether a movement pattern of two mobile nodes leads to network partitioning, assuming location awareness of nodes and the circular transmission range. Each node must maintain an information table, where it stores updates from all other network nodes. If the partitioning is predicted, the algorithm offers data duplication techniques that logically prevent partitioning.

Probably the weakest point of proposed algorithms is their trust proactive routing protocols deliver ideally accurate topology information in presence of shadowing and fading on communication channel. Wireless communication channels impose certain

restrictions on topology discovery and dissemination process as it will be described in Section 3.3.

Such assumption were supported by evaluation of proactive routing protocols under unrealistic and idealized communication channel properties. It will be shown in Section 9.2 that proactive global topology management suffers greatly if a more realistic simulation setup is used.

All listed approaches require global topology knowledge and three of them additionally rely on location information and circular communication range. None of them seriously treats problems introduced by obstacles and signal interference (Figure 2.2), packet losses, topology inaccuracies, errors in determining node location (e.g., GPS introduces errors of up to 15 meters). Only in [57] scalability of global knowledge management is noted as an important problem. Even with the idealized communication model, all location-aware approaches are limited to two-dimensional space. In scenarios where node elevation is important or GPS signal is unavailable (as it was in the office-building where Motelab is deployed and where our detection algorithm has been tested) they cannot be used.

Summarizing all the facts that are in favor and against existing models, it is difficult to envision that the related approaches may operate in reality as they were presented in related work considering the level of abstraction they have assumed, unrealistic assumptions, presentation shallowness, and incompleteness of evaluation.

3.2. Other Approaches for Circumvention of Network Partitioning or Its Effects

Detection of nodes and edges critical for network connectivity and application of corrective actions to them (or to a network in order to avoid negative effects if they are removed) is not the only solution to the problem of network partitioning, bridge and articulation point existence. It is also possible to prevent formation of partitions by controlling topology or to treat disconnection as a valid state of the network.

3.2.1. Topology Control

By changing node and network parameters, such as node density, location or transmission power, it is possible to shape topology of the network so that certain optimization goal is achieved. Typical optimization goal is to build a k-connected network while minimizing the sum of transmission energies at individual nodes. The same theoretical framework can be used for other topology optimization purposes such as construction of spanning trees under same, minimum energy condition.

Depending on a parameter that is being changed and additional constraints, numerous topology control algorithms and protocols have been developed. Some of the algorithms require to be applied before network deployment as they require increased node density or careful positioning of certain nodes. Others are applicable to deployed networks since they require only parameter tuning at individual nodes.

Optimal Energy Assignments at Nodes in a Network

This class of algorithms attempt to solve the following problem: Given a set of nodes and their locations on a line, in a plane or in a three dimensional space, set the individual transmission power of nodes so that a pre-defined topological property is guaranteed, simultaneously minimizing the sum of transmission powers over all nodes in the network. The desired topological property differs depending on the application scenario (e.g., multicast, broadcast and convergecast trees, k-connectivity).

The problem of transmission energy minimization while guaranteeing $k - node$ connectivity is NP-hard. Kirousis et al. [99] proved it for 3-dimensional Euclidean space, while Clementi et al. [60] have proven it for 2-dimensional space. Problem of finding minimal transmission energy allocation for $k - edge$-connectivity is also NP hard [47], even for bidirectional communication links.

As the consequence of NP hardness, approximations and heuristics must be used for all non-trivial network sizes. Some of heuristics, distinguishable by approximation factors, can be found in [47], [49], [60], [99], etc.

Probabilistic Topology Control

For large networks it may be too energy expensive to guarantee k-connectivity and it is difficult to tune parameters of every node in a network. The guarantees are relaxed. Instead, the probability that a network is k-connected for a given set of node and network parameters is calculated. The probability functions usually can be inverted to calculate the network and node parameter which provide the desired connectivity within predefined probability threshold.

The continuum percolation theory is used in statistical physics to model flow in porous media, state transition in polymers, etc. The main topic of interest in the continuum percolation theory is to determine the critical placement intensity of points λ_c in a random geometric graph such that $\forall\lambda, \lambda > \lambda_c$ there is an unbounded component of the underlying graph. The closed form expression for the λ_c is not known for spaces with more than one dimension and only rough boundaries and certain properties have been derived so far.

Gilbert [78] was one of pioneers that introduced percolation theory to networking. He attempted to answer the question if there exists the critical node density that enables long range, multiple-hop communication in a wireless multi-hop network whose nodes have circular transmission range R.

In [41] is analyzed the empirical distribution function of the lengths of edges in a minimal spanning tree on a graph created by a Poisson point process on a lattice in a d-dimensional bounded space. Knowing the distribution of edge lengths in minimal spanning tree would straightforwardly lead to various other statistical properties (e.g., the average number of partitions, probability of graph connectivity for a given R) but unfortunately, the exact distribution has not been derived. The authors only prove existence of the upper bound function for the distribution of the number of edges such that their length is shorter than the communication range R.

Bettstetter [36] investigates the impact of minimum node degree in ad hoc networks on their connectivity. A relation between the two can be used to determine the minimum communication range or the minimum node density at which the network will be $k -$

node-connected with a high probability. The model is developed for the unbounded area of placement and does not take into account the boundary effects (nodes close to the border of placement area have fewer neighbors than nodes in the central part of the placement area). As it was observed in [36], the area-boundary effects can introduce a considerable difference between expected and observed connectivity level in bounded networks. Bettstetter and Hartmann [37] extend this approach to environments with shadowing on the channel using almost identical methodology.

Booth et al. in [43] and Franceschetti et al. [73] show that the presence of non-circular communication range may increase the overall connectivity of the network provided that the average number of functioning connections per node is maintained. As the consequence, networks with irregular communication patterns need lower values of point-process intensity (lower node density) compared with the networks where the communication is within circular disks.

Bates [35] derives the so called "magical number" – the average node degree that produces connected network with a high probability and proposes an algorithm for construction of connected networks. The uniform node placement and circular communication radius are assumed.

Fault-tolerant placement of nodes and $k-$*node*-connectivity in wireless ad hoc networks with boundary effects is studied by Li et al. in [107]. They provide an expression for the communication radius R that creates k-connected network with a high probability in the bounded placement area. The results are used to create a localized method for network topology control. The method reduces the number of maintained links and preserves the fault-tolerant topology.

Summary

The common point for all described algorithms is the power-function for modeling of dependency between energy used for signal transmission and communication radius. The implication of this functional dependency is the capability of nodes to communicate within the communication radius, with a sharp transition to state of impossibility of communication at inter-node distances greater than the communication radius.

While acceptable for theoretical discussion and early protocol development, such assumptions are not applicable in real networks. It is rare that authors attempt to theoretically or practically tackle the differences in communication links that are imposed on network by the environment as it is performed in [43] and [73].

Even if models are general enough to capture reality, the main prerequisite for *application* of these algorithms and protocols is the possibility of providing sufficient number of nodes in the area, or capability to increase their communication range. Often that is not the case:

- The number of nodes cannot be significantly increased in an area due to their cost, increase of channel contention or unwillingness of new nodes to join the network.
- Communication capabilities of nodes are bounded by limitations of radio equipment, electromagnetic radiation standards in used frequency bands, environment and physical laws.
- Node failures during network lifetime may compromise the calculated properties.

3.2.2. Partitioning Prevention by Mobility

Basu and Redi [34] propose two heuristics for node movement in a mobile network for preserving of 2-connectivity. The goal is to minimize the sum of node travel distances which are required for creation of a 2-connected network. Both heuristics require precise global topology knowledge, node locations and assume circular communication range.

The first heuristics is simple – it just moves all nodes towards the geometrical center of the network. The network contraction parameter α regulates the relative location offset towards the center. In turn, network density is increased and connectivity improved. The algorithm tends to group nodes too much, in particular if the parameter α is not properly chosen: as parameter α approaches zero, network may be transformed into a complete graph, or it may even contract to a single point.

Second heuristics divides nodes in so called blocks (2-connected subgraphs) and attempts to move the blocks until 2-connectivity is established. Nodes are required to travel less than in the first case, and fewer nodes are required to move.

Das et al. [63] propose a localized algorithm for detection of articulation points and for deciding which nodes are to be moved in order to create at least 2-connected network. Instead of the complete network topology a node continuously tracks only its k-hop neighborhood. The main contribution is that decisions are made based on partial topological information, accepting the tradeoff that faulty decisions are possible but with benefit of reduced communication overhead.

The authors of [63] make series of unrealistic assumptions: absolute node obedience, circular communication range, no errors in location estimation, instantaneous link break detection and instantaneous topology updates dissemination, without communication overhead. In such favorable conditions the approach is highly successful in very dense networks (average node degree of at least 10) but its accuracy reduces as node density is lowered.

The authors claim that 3-hop neighborhood knowledge already provides almost ideal decisions, but their conclusion must be taken with consideration: the network in their study is small (100 nodes) and very dense, so 3-hop neighborhood actually captures most of the network topology. Prompt deterioration of the algorithm performance in sparse networks confirms that algorithm requires considerable topology knowledge after all.

3.2.3. Disruption-Tolerant Networks

In an extremely sparse network, it may remain disconnected even if node transmission power is pushed to upper limits. If some of nodes are mobile and applications deployed in network are *delay tolerant*, like for instance non-real-time sensing, e-mail, voluminous data transfer, it is possible to create data exchange mechanisms. A simple example of the technology is the Wizzy Digital Courier project [2] that provides Internet access to remote schools by carrying them data on a digital device attached to a motor vehicle. At a remote school, incoming and outgoing data is exchanged and taken to a point with Internet access where it is uploaded, responses collected and brought back to the school. The process is then repeated.

Core idea of all protocols from this category is that source of information forwards data to one or more mobile nodes that will eventually come in contact with data destina-

tion. Existence of mobile nodes is mandatory and crucial for such protocols, since they allow data exchange between distant and independent network components. There exist multiple strategies for selection of nodes to forward the data. The forwarding strategy is always a tradeoff since improving of one performance parameter makes negative impact on another (e.g., reducing communication latency increases communication overhead).

Data replication improves delivery ratios and reduces latency. Source node distributes multiple copies of the same data to nodes it contacts. Thanks to mobility, either the data source or one of nodes that possesses the copy of it will eventually contact destination node and deliver the data to it. As number of replicas increase, the contact probability increases as well. Drawback of data replication is considerably increased traffic overhead and energy consumption. If data buffers at intermediate nodes are overloaded, they have to disregard some of the messages. It means that energy used for transmission of such copies was wasted. Epidemic protocol [166] is an extreme example of the data replication approach: each message is flooded through the network, creating excessive traffic. Data source is the only node allowed to create replicas in approach of Spyropoulos et al. [153]. The source produces number of replicas proportional to network size, reducing the overhead.

Zhao et al. [180] introduce so called "ferry" nodes: they have predictable mobility and can be used for message exchange in highly disconnected networks with stable partitions.

In conditions of higher mobility is beneficial to discover and manage groups: within a group messages can be exchanged using some of known reactive or proactive protocols and between groups a protocol customized for DTNs is employed. Thomas et al. [159] use DSDV for in-group routing and develop BLOB for inter-group routing and group management. However, their work is highly reliant on underlying node mobility models (*Community* and *Inplace*) where node interactions are frequent and highly predictable.

If group existence is unknown or groups do not exist, a more general framework is needed. Lindgren et al. [109] propose Probabilistic ROuting Protocol using History of Encounters and Transitivity (PROPHET). In PROPHET nodes track the connection history and calculate from it how likely it is that a node will be able to deliver a message to a destination (they call it *delivery predictability*). When two nodes a and b meet, a message is sent from a to b if b has higher *delivery predictability* for the message than a.

If applications deployed in network are interactive or real-time, DTN routing protocols cannot be used. Unfortunately, most of applications require interactive traffic, so the applicability of DTN approaches is rather limited.

3.3. Proactive Topology Management in WMNs

Proactive topology management is used by a wide class of protocols that require global topology knowledge at each node in a network. Its primary application is for the proactive routing protocols. Increased overhead introduced by the topology dissemination is compensated by the reduced latency: since each node in network is able to calculate path to any other node, thus avoiding the delay introduced by route discovery process in reactive routing protocols.

Low latency for route calculation comes at a price – even without traffic in the network, routing traffic and energy consumption is considerable. Topology dissemination messages compete for the access to the wireless channel with user generated traffic, re-

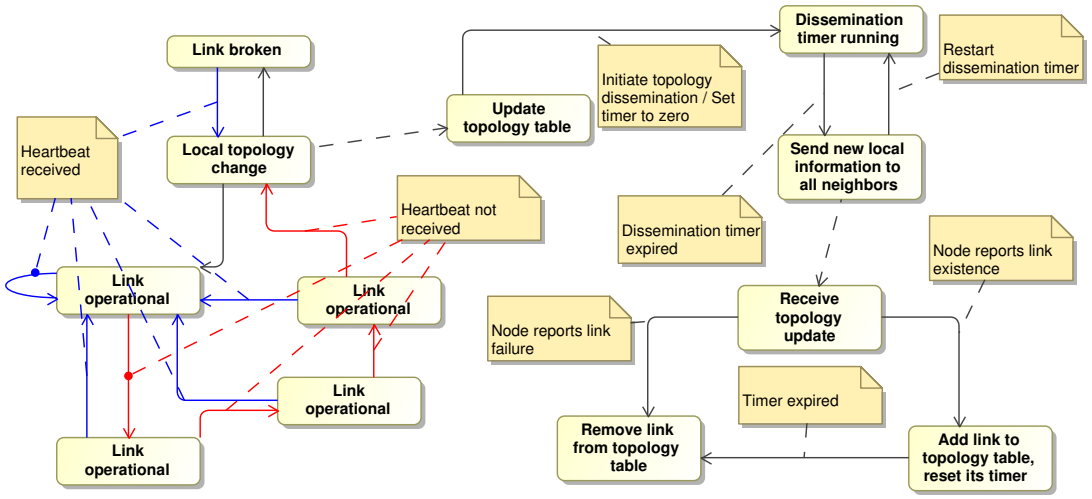


Figure 3.1.: Proactive topology management.

ducing bandwidth available to users and increasing probability of packet losses through collisions.

In order to detect and disseminate topology in a wireless network, three processes must be executed concurrently at each node: management of local neighborhood set (link discovery), dissemination of local topology data and management of global topology data.

The commonest implementation of link detection is through exchange of heartbeat messages (also called *beacons* or *hello messages*) with neighbors. A heartbeat link detector (HLD) recognizes link as active after a successive heartbeat receptions. If a node observes r successive heartbeat omissions on an active link, it declares the link failure due to neighbor removal (movement, failure or shutdown). Such heartbeat link detector will be referred as $HLD(a, r)$ in the text. Figure 3.1 illustrates the processes for $HLD(1, 3)$.

Dissemination of local topology is periodically initiated even if no changes in local set are observed. Nodes new to the network may receive it and build network's topological structure. Some proactive implementations initiate topology update as soon as a change in the local topology has been sensed. Others attempt to preserve bandwidth through periodical dissemination, with the goal of capturing of a set of topological changes in a single message.

Type and detail of disseminated data depends on application scenario and developer's decision. Distance vector routing protocols do not share topology information directly, but hop distance to other destinations. Such information is specialized for routing and cannot be used for other purposes (e.g. calculation of the minimum spanning tree). Others disseminate local neighbor set, allowing route as well as other topological calculations at network nodes.

Dissemination informs all network participants of the observed local topology at a single node. Usually it is implemented as flooding, but other approaches are possible. For instance, in OLSR [59] routing protocol only a subset of nodes in the network, the so

called multipoint relay (MPR) nodes[2] are responsible for forwarding of topology traffic, reducing the contention on the wireless channel.

The dissemination process is further complicated since some proactive topology management protocols allow dissemination of various topological subsets. For instance, in the OLSR protocol, the TC_REDUNDANCY parameter regulates which information is exchanged in a network:

- If the TC_REDUNDANCY parameter of a node is 0, only MPR selector set is disseminated.
- If the TC_REDUNDANCY parameter of a node is 1, node advertises union of its MPR set and its MPR selector set.
- If the TC_REDUNDANCY parameter of a node is 2, node shares its complete neighborhood.

Once a node receives information of distant topology subgraph, it stores links in its topology table and assigns to each of them a validity timer. Each time a node receives an update on link existence, the corresponding timer is reset.

Information on link existence is removed from the topology table if its timer expires (information in the topology table is stale) or if a node incident to the link disseminates message that the link in question is no longer operational.

Theoretically, if all nodes in the network behave in presented manner, all nodes will have precise topology information, or excellent approximation of it, even in presence of node mobility. However, proactive topology management faces difficulties if it is applied in networks with communication links that cannot guarantee message delivery, which is exactly the situation encountered in WMNs. The detailed functionality of link detection and dissemination is described in more detail in the next two sections, with particular focus on issues introduced by the unreliable communication channel.

3.3.1. Link Detection in Wireless Multi-hop Networks

The heartbeat link detectors are widely used in WMNs in protocols that need some sort of topological knowledge since there does not exist another approach to link detection that is as universally applicable as an HLD. One alternative approach to HLD uses MAC feedback to decide upon link existence and it may provide more accurate decisions since it utilizes information which is invisible to the network layer where an HLD resides. However, it has several drawbacks:

- Its accuracy advantage may be result of technology specific details and it cannot be used in other technologies although the network layer protocol which needs the results of link detection is technology independent.
- Dependance on support from hardware vendor. Even if MAC-layer link detection is intended for use in a single networking technology, it may not be possible to access the needed data from MAC layer of a network adapter since it is not supported by drivers of network adapter, or it is not possible to perform changes in the firmware of the network adapter.

[2]Detailed description and definition of MPR nodes and selector set can be found in [59].

- MAC feedback is not necessarily better. In [113] heartbeat and MAC-feedback link detection are compared in simulations facilitated by the ns2 simulator. It is concluded that MAC feedback is better suited to low-traffic scenarios while heartbeat detection is better for scenarios with intensive traffic.

In order to decouple protocols from technology type and to reduce dependance of a protocol on hardware vendor, protocols from the network layer use the heartbeat link detectors, or at least support them (e.g., AODV can be configured to use either heartbeat or MAC-feedback link detector).

The WMN routing protocols that reached RFC status support heartbeat link detection in their default configuration. For instance, AODV uses $HLD(1,2)$ [137], DSR (Dynamic Source Routing) is approximately $HLD(1,8)$ since it accepts a link after a successful broadcast and rejects it on unicast failure [96]. OLSR introduces a complicated hysteresis detection scheme, whose functionality can be reduced to an HLD: a link is rejected after two successive heartbeat omissions, and accepted after two or three successive heartbeat receptions. Thus, the OLSR with default parameters [59] uses either $HLD(2,2)$ or $HLD(3,2)$.

As it can be seen on example of the RFC routing protocols, no consensus on HLD parameterization has been reached in community yet, despite their simplicity. But the importance of HLD parameterization has been noted in research community. In [87] was shown through experiments that common configurations of heartbeat link detectors may not function appropriately in practice because of the shadowing and fading on the channel. It is proposed to divide links into two categories: stable (high quality links, desirable use) and weak (detected but with low quality). Based on simulation and experimental evaluation, $HLD(12,3)$ is proposed for acceptance of stable links. An experimental study of AODV was performed in [52] in order to compare $HLD(1,2)$ and $HLD(1,3)$. $HLD(1,3)$ resulted in better performance of AODV which was attributed to smaller number of false negatives in link detection.

Some WMN simulation studies investigated other HLD parameters, in particular the heartbeat frequency. The location-aware algorithm developed in [56] aims to reduce the routing overhead of AODV through link availability prediction in mobile WMNs. The frequency of heartbeats is not fixed as in AODV [137] but is a function of internode distance: as the node distance increases and approaches the communication radius R, heartbeat frequency is increased. If the distance is small, frequency is reduced. Simulation results demonstrate overhead reduction, but since they are based on the two-ray ground propagation model they do not take the presence of obstacles and channel shadowing into account. Three versions of the heartbeat link detection protocol are compared in GloMoSim simulator in [79]: periodic, adaptive (node moving with greater speed emits heartbeats more frequently, similar to [56]) and reactive (sending heartbeats as reaction to demand coming from upper layers of communication stack). Protocols are integrated in AODV and GPSR protocols, and a simulation study is performed in order to evaluate effects of the proposed link detection protocols on the average throughput and end-to-end communication delay. Same as in [56], the communication range R is used as the clear delimiter between perfect communication and no communication at all, but at least there exists awareness of false positives in link detection caused by node mobility.

The simulation studies provide only a partial insight in HLD behavior, and a comparison of a limited number of their configurations. Additional drawback of simulation studies in [56][79][113] is the simplified communication model resulting in unawareness of HLD issues observed in experiments [52][87]. The effects of shadowing and fading are completely ignored in [56][79][113], assuming that nodes are able to communicate if their distance is less than R and that all losses within this communication range originate from the collisions on the channel. This eliminates most of the HLD issues and they behave as it is expected of them.

Considering the widespread use of heartbeat link detectors in WMN protocols, it is surprising that only studies of characteristics of $HLD(1, \infty)$ exist. A time slotted model where nodes either listen or transmit is used in [114] for analytic analysis of link detection under assumption that packets are lost only due to collisions. The tradeoffs between energy expenditure, probability of neighbor discovery and delay to link discovery are analyzed. The same model is used in [28] to evaluate a slightly different link detection protocol (link AB is detected after successive exchange of heartbeats between nodes A and B in both directions). An important metric is introduced in [28] – the time to detection which is defined as the number of protocol executions (rounds) to link detection. This model is extended in [83] by a sleeping mode of a node where it neither sends nor receives messages, and a similar analysis as in [28] and [114] is performed.

All three papers analyse $HLD(1, \infty)$. They are not concerned with detection of link failure nor with the issues in higher layers of communication stack caused by detection of links with exceptionally poor quality. The $HLD(1, \infty)$ can be modeled as series of Bernoulli trials with the success probability p. The probability of detecting a link after k attempts is then $1-(1-p)^k$. As long as $p > 0$, regardless of how small the probability p is, the probability of link detection asymptotically approaches one after sufficient number of execution rounds, despite the fact that such links may not be suitable for operation of a communication protocol.

Another area where heartbeats are frequently applied is the node failure detection in multiprocessor systems [26][42][156], allowing operational processors to detect the failed processors. The algorithms for processor failure detection differ with regard to their assumptions (e.g., by applicability in different network topologies, synchronous or asynchronous systems, whether message delivery is delayed or instantaneous) but the majority of algorithms share the assumption of reliable message delivery, as in [42][156]. A notable exception is [26] where nodes may fail and messages may be lost at communication links. The algorithm guarantees processor failure detection and tolerates arbitrarily high heartbeat losses if for an infinite number of sent messages by an operational node to another operational node, intended destination also receives an infinite number of messages. As such, it is applicable for detection of node failures in a WMN but it still faces the identical issues in link detection as the $HLD(1, \infty)$ – after large but finite number of attempts it will accept links to all operational neighbors if the link quality is larger than zero.

3.3.2. Local Topology Dissemination

The information on detected links is disseminated through network, using some of existing network-broadcast mechanisms. It has been observed in simulations with simplistic propagation and node placement models that common network-broadcasting mecha-

nisms such as flooding produce undesirable overhead, and most of the effort in community was directed at the overhead reduction.

For instance, Mukherjee et al. [125] assume that links are broken only due to energy depletion or network mobility, and that heartbeats are successfully exchanged with a very high probability. They come to a conclusion that the heartbeat period may be longer than 10 seconds and the topology dissemination period can be up to 90 seconds without compromising the routing protocol capability of data delivery. Under similar assumptions Williams and Camp [174] and Ni et al. [129] through simulation study found that flooding has excellent coverage (reaches majority of nodes even in high mobility networks) and that it produces unnecessary and undesirable overhead, inspiring considerable amount of work on reduction of its redundancy. The simplest approach, presented in [129] is to broadcast a packet with fixed probability p or to drop it with probability $1-p$. Same authors extend their ideas, introducing adaptivity and location-based schemes in [165]. Naserian and Tepe [127] propose a game theory approach to selection of nodes that participate in flooding as an extension of probabilistic selection approaches.

They all assume dense networks, high probability of packet delivery and circular communication pattern if localization is used. The communication overhead is reduced, but also the robustness of dissemination. Contrary to these simulation studies, the experience from large scale community networks [7][8] calls for considerable increase in frequency of local link information dissemination as well as higher robustness of dissemination protocols in order to reduce errors in global topology view at nodes.

3.4. Summary

This section has reviewed approaches for biconnectivity testing and dependability improvements in WMNs. Three classes of approaches can be distinguished in the literature.

Topology control aims at creation of k-connected topologies ($k \geq 1$) and simultaneously attempts to minimize the transmission energy expenditure in the network. Typically, the goal is to produce at least a 2-connected network. The problem is NP-complete so various heuristics are developed.

The second class does not guarantee a topological property, but it is achieved with a certain probability. Usually, the goal is to calculate the minimum node density that establishes the desired topological characteristics with a target probability.

The third class attempts to discover bridges and articulation points and then to apply the corrective actions at network level (e.g., to move nodes into area with reduced connectivity). Biconnectivity testing is performed through application of DFS on a topology that was delivered by a proactive topology management protocol.

The basic ideas of all proposed approaches are sound and well motivated. The employed mathematical models are powerful and give important insights in topological properties of WMNs. However, some important characteristics of WMNs and their environment are ignored in related work:

- Links in WMNs do not have behavior of Heaviside function (ideal communication within communication range, no communication outside of it). Providing a barely functional link brings little in practice – packet losses may be unacceptable for network services and applications. This has the largest effect on topology control

algorithms, since they deliberately reduce transmission power of nodes so that the received signal strength is barely over reception threshold. Even in presence of mild noise on the communication channel, such links may become unusable.

- Signal strength is not monotonically reduced with increase in node distance and the communication range is not circular (Figure 2.2, page 17). Thus, it is not possible to calculate connectivity of a network from node locations. This affects the topology control algorithms which calculate transmission power as a power-function of distance, and the proactive approaches that attempt to reinforce bridges and articulation points.

- Obstacles cannot be ignored. They may reduce quality of communication, completely block it, or prevent node placement and movement.

- Shadowing and fading affect link discovery and local topology data dissemination. Global topology view at nodes may be erroneous.

- Location-aware methods ignore errors in the node-localization process, or location-service unavailability due to technical and environmental factors (e.g., GPS cannot operate inside buildings).

- Networks are three-dimensional. Methodologies operating exclusively in a plane may fail in presence of node elevation.

- Nodes may ignore the orders of algorithm, in particular if demanded action is movement. All approaches that employ node mobility assume absolute node obedience.

The differences between assumptions in related work and reality are huge. This is acceptable in the modeling phase of protocol development since models need to be abstract, sacrificing details in order to become tractable and solvable. However, the results obtained in such abstract models must be evaluated in realistic conditions in order to verify their usefulness and applicability.

Unfortunately, the evaluation methodology of described protocols was as imprecise as the starting assumptions. None of the presented topology control, probabilistic property estimation, or biconnectivity testing protocols and algorithms has been experimentally evaluated. Instead, various simulation methodologies have been employed for protocol testing. But the simulation setup was as imprecise as the starting assumptions: path-loss and two-ray ground propagation models were used without stochastic attenuation components, localization services have perfect accuracy, the world is flat and there are no obstacles in the environment.

Summarizing all the facts that are in favor and against existing models, it is difficult to envision that the related approaches may operate in reality as they were presented in related work considering the level of abstraction they have assumed, unrealistic assumptions, presentation shallowness, and incompleteness of evaluation.

4. Heartbeat-Based Link Status Detection in Wireless Multi-hop Networks

This chapter evaluates existing approaches for link detection in wireless networks that are based on heartbeats. Stochastic models are derived, which describe their behavior as functions of detector parameters and network characteristics. The obtained results allow simple and efficient assessment of heartbeat link detector (HLD) properties for a wide set of parameters in different network types.

The stochastic nature of wireless communication channels is one of the main sources of uncertainty for heartbeat link detectors. Unreliability of communication introduces omissions in heartbeat receptions, and a heartbeat detector cannot distinguish between losses caused by the unreliable channel and omissions caused by link or node failure. The situation is further complicated for HLDs since they are to accept only links with a sufficient quality which is not implicitly supported by them, but it is nevertheless required by the application layer, as explained in Section 2.2.1 (page 20).

Unreliable communication results in errors in the link detection. Errors in the link detection process are false positives and false negatives. A false positive in link detection occurs if an HLD declares a link as active although its quality is below the link acceptance threshold t. A false negative occurs if an HLD declares a link as inactive although its quality is equal to or better than the link acceptance threshold.

The goal of the presented evaluation is to model HLDs (in particular those defined in RFCs) in static and dynamic networks and to determine the probability of errors in the detection process. The existence of these errors is not surprising, but an important result of this chapter is that these errors are inevitable, no matter to which value detector parameters are set, and irrespective of the link acceptance threshold t. The developed models are then applied to two real networks in order to determine the optimal HLD parameters that yield the minimal error probability.

The chapter is organized as follows. The HLD model is introduced in Section 4.1. The behavior of heartbeat protocols at a single unreliable link is analyzed in Section 4.2.1 under assumption that neither of nodes incident to the link fails. The single-link analysis is extended in Section 4.2.2 and it provides an efficient methodology for HLD evaluation at the network level. Node failures and limited link existence duration are introduced in Section 4.3 in order to evaluate effects of mobility on HLDs. Effects of errors in link detection process on biconnectivity testing are discussed in Section 4.4.

4.1. Heartbeat Link Detector Model

For evaluation of accuracy of heartbeat link detection, the graph model of a network from Section 2.2 (page 19) is used: a pair of nodes is connected over a communication channel which does not guarantee message delivery. Instead, messages are delivered with probability p (term "link quality" is also used) and lost with probability $q = 1 - p$.

Probability of message delivery (link quality)	p
Packet loss probability	q
Probability density function of link quality	$f_p(p)$
Heartbeats needed for acceptance of a link	a
Omissions needed for rejection of a link	r
Link acceptance threshold	t
Rate of the link existence duration	μ

Table 4.1.: Parameters used for evaluation of heartbeat protocols and their symbols.

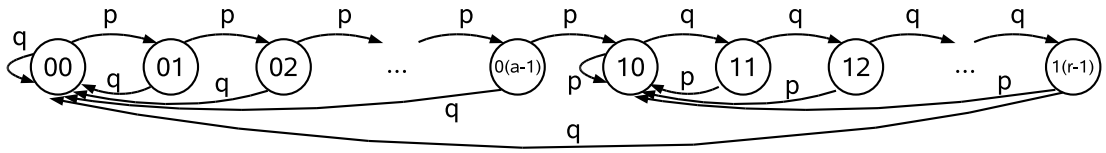


Figure 4.1.: Heartbeat link detector.

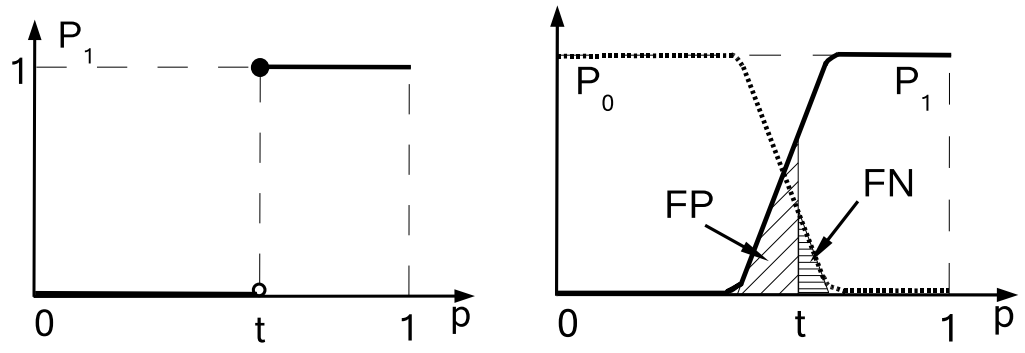
It is assumed that the probability of heartbeat collision and resulting message losses is already included in the probabilities p and q. The probability density function of the random variable p is $f_p(p)$.

A link between two nodes exists if its quality is higher than a link acceptance threshold t. Equivalently, a node B is recognized as a neighbor of node A only if quality of link between A and B is higher than t. In dynamic networks, links between nodes exist for a limited time. The average duration of link existence is $1/\mu$. In static networks it is assumed that neither nodes nor links fail. The parameters of the developed model are listed in Table 4.1.

Nodes incident to a link exchange heartbeat messages in regular intervals, which allows the detection of heartbeat omissions. The task of a HLD is to determine if there exists a link to another node based on the exchanged heartbeats. Essentially, a HLD is estimating the value of random variable p, which is unknown to it. The strategy to decide whether a link has sufficient quality (estimated p is larger than t) consists of two rules: one for acceptance and one for rejection of the hypothesis of link existence.

It is common for HLDs [96] [137] to start with the hypothesis that link does not exist (state 00 in Figure 4.1). If a is larger than one, there exist intermediate states $01, 02, ...0(a-1)$ in which node supports the hypothesis that no link exists. Intermediate states are reached by successive heartbeat receptions (e.g., state 01 is reached after one, state 02 after two receptions, etc.). A single heartbeat omission in any of these intermediate states is sufficient for transition to state 00.

If a HLD observers a successive heartbeat receptions starting from state 00, it enters state 10 and accepts the hypothesis of link existence. If $r > 1$ there exist states $11, 12, ..., 1(r-1)$ in which node also supports the hypothesis of link existence. These intermediate states are reached by successive heartbeat omissions. If system is in these intermediate states, a single heartbeat reception returns it to state 10. If r successive heartbeat omissions are observed from state 10, it is interpreted as a failure of a link,



(a) Ideal link detector.

(b) Imperfect link detector.

Figure 4.2.: Comparison of transition curves of ideal and imperfect link detector and errors in detection process.

leading to transition into state 00.

Such decision strategy is denoted by $HLD(a, r)$. The function that maps probability of declaring a link with quality p as operational by the $HLD(a, r)$ is denoted as the transition curve $P_1(a, r, p)$. $P_0(a, r, p)$ is the probability of rejecting the hypothesis of link existence, $P_0 = 1 - P_1$.

The ideal link detector (detector which does not make errors) declares a link existence if and only if link quality p is higher than or equal to a threshold t. Its characteristics has the shape of step function, as shown in Figure 4.2(a).

Due to the fact that the true link quality p is not known, the step function of ideal link detector cannot be achieved based on a finite number of samples by a $HLD(a, r)$. A sample characteristics of a real detector is shown in Figure 4.2(b). It is possible that $P_1(p) > 0$ for $p < t$ (a false positive in link detection may occur), and $P_0(p) > 0$ for $p \geq t$ (a false negative in link detection may occur), which leads to errors in the link detection process. The probability of detection error for a link with quality p can be calculated as:

$$P_E(p, t) = \int_0^t P_1(p)dp + \int_t^1 P_0(p)dp \tag{4.1}$$

4.2. Analysis of Heartbeat Link Detector Behavior in Static Networks

In this section the HLD behavior in static networks is analyzed – namely, links are assumed to exist for infinite time (i.e., nodes incident to them do not fail). Since it is assumed that links exist for an infinite duration, the transition curve is sufficient to describe behavior of a $HLD(a, r)$. The analysis of behavior of HLDs at a single link is performed in Section 4.2.1. It is extended to the whole network in Section 4.2.2.

states	00	01	02	...	0(a-1)	10	11	12	...	1(r-1)
00	q	p								
01	q		p							
02	q			p						
...	q				p					
0(a-1)	q					p				
10						p	q			
11						p		q		
12						p			q	
...						p				q
1(r-1)	q					p				

Table 4.2.: The transition matrix of HLD state machine from Figure 4.1.

4.2.1. Heartbeat Link Detector at a Link without Node Failures

The complete behavior of the automata in Figure 4.1 can be described by a transition matrix from Table 4.2 and then solved in some of the existing tools, such as SHARPE [147][1]. The matrix is easily derived for arbitrary a, r and p. However, the closed form solutions for the transition curve of a $HLD(a, r)$ are developed here in order to provide better insight into its behavior and discussion of its properties.

Let P_{00} denote probability that automata is in state 00. It can be calculated as the sum of probabilities of all transitions that lead to this state:

$$P_{00} = P_{00}q + P_{01}q + P_{02}q + ... + P_{0a-1}q + P_{10}q^r$$

Knowing that $P_{01} = P_{00}p, P_{02} = P_{00}p^2, ..., P_{0a-1} = P_{00}p^{a-1}$, the probability P_{00} can be expressed as:

$$P_{00}(1 - q(1 + p + p^2 + ... + p^{a-1})) = P_{10}q^r$$

Using $1 + p + p^2 + ... + p^{a-1} = \sum_{i=0}^{a-1} p^i = \frac{1-p^a}{1-p} = \frac{1-p^a}{q}$ yields:

$$P_{00}p^a = P_{10}q^r \tag{4.2}$$

Furthermore, the sum of probabilities of all states of the automata must be one:

$$\sum_{i=0}^{a-1} P_{0i} + \sum_{j=0}^{r-1} P_{1j} = 1$$

$$\sum_{i=0}^{a-1} P_{00}p^i + \sum_{j=0}^{r-1} P_{10}q^j = 1 \tag{4.3}$$

[1]The automata is represented as a Markov chain and it is solved for steady-state probabilities of its states.

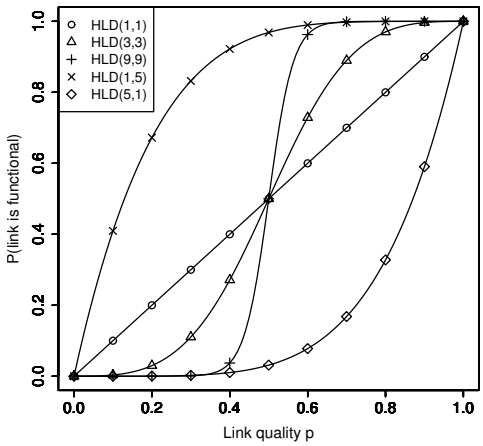


Figure 4.3.: Probability of declaring a link functional by a HLD.

Substituting Equation 4.2 in 4.3 and solving for the probability of state 00 yields:

$$P_{00}(a,r,p) = \frac{q^r}{q^{r-1}(1-p^a) + p^{a-1}(1-q^r)} \tag{4.4}$$

Subsequently, the probability that automata declares the link as not functional is $P_0(a,r,p) = \sum_{i=0}^{a-1} P_{0i} = \sum_{i=0}^{a-1} P_{00}p^i$, or:

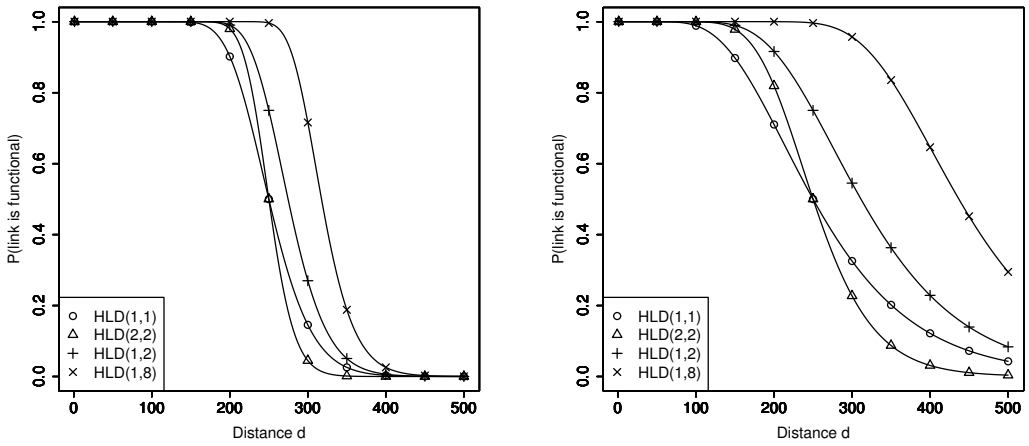
$$P_0(a,r,p) = P_{00}\sum_{i=0}^{a-1} p^i = P_{00}\frac{1-p^a}{q} = \frac{q^{r-1}(1-p^a)}{q^{r-1}(1-p^a) + p^{a-1}(1-q^r)} \tag{4.5}$$

Probability of declaring link as functional is $P_1 = 1 - P_0$:

$$P_1(a,r,p) = \frac{p^{a-1}(1-q^r)}{q^{r-1}(1-p^a) + p^{a-1}(1-q^r)} \tag{4.6}$$

The Equations 4.5 and 4.6 lead to several conclusions (illustrated in Figure 4.3):

- If $a < r$, $HLD(a,r)$ is more likely to accept weaker links. $HLD(a,r)$ is more likely to accept links with higher quality if $a > r$.
- P_0 and P_1 are strictly larger than zero on $0 < p < 1$, so the errors in detection are inevitable regardless of the threshold t and parameter selection (the direct consequence of Equation 4.1).
- Higher values of a and r result in steeper transition curves. Steeper transition curve is closer to the ideal link detector (Figure 4.2(a)) and it results in smaller error probabilities (assuming combination of a and r is chosen so that the transition is close to the acceptance threshold t). Thus, it can be concluded that for links with infinite duration, high values for a and r are preferable.
- If p is close to one, selection of parameters (a,r) is of limited importance – even if they have low values, P_1 is very close to one and error probabilities are negligible.



(a) $\alpha = 4$, $\sigma = 3$.

(b) $\alpha = 4$, $\sigma = 7$.

Figure 4.4.: Heartbeat parameters and probability of link detection as function of node distance in presence of signal shadowing.

If two-ray-ground propagation model is used in WMN simulation, links are either of high quality (losses originate only from contention) or nonexistent. From such assumptions follows that HLDs operate exactly in the best part of their transition curve, in the consequence offering unrealistically high detection accuracy.

Equations 4.5 and 4.6 can also be used to describe behavior of heartbeat link detectors as a function of distance if shadowing is present on communication channel. In accordance with [155], the average probability of successful packet reception at distance d with propagation coefficient α and shadowing coefficient σ is:

$$p(d) = \frac{1}{2} - \frac{1}{2} erf\left(\frac{10\alpha}{\sqrt{2}\sigma} \log_{10} \frac{d}{r_0}\right)$$

where losses caused by contention on the channel are ignored.

This probability can be substituted in Equations 4.5 and 4.6, providing the transition curve of an HLD on the wireless channel with shadowing as a function of distance. Examples of transition curves are shown in Figure 4.4 for two combinations of parameters α, σ.

The probability $P_1(1, 1, p)$ is equal to p as can be derived from Equation 4.6. It can be used as the reference point for comparison of the link acceptance probability P_1 with link quality p. Several important conclusions can be drawn:

- Smaller values of σ reduce the effects of shadowing and the variation of signal strength thus improving the characteristics of the transition curve (making it steeper). If σ is set to zero, there are no variations in link quality – it is one if distance is less than R and zero if it is larger than it. As the consequence, all HLDs have the same transition curve regardless of the parameters a and r: $P_1 = 1 - H(R)$. Potential losses caused by contention are typically limited, so already HLD(1,2) is sufficient to provide stable and correct link detection (within

this context), which partially explains selection of these parameters in RFCs (additional reasons are provided in Section 4.3). As σ increases, transition curves of HLDs with low values of a and r flatten, increasing the error probability and issues of HLDs observed in practice.

- Behavior of an HLD is no longer dependant only on its parameters, but also on parameters of the environment. Higher values of parameter σ (more pronounced shadowing) reduce steepness of transition curve. For instance, if $\sigma = 3$ all HLDs shown in Figure 4.4(a) reject links if internode distance is 500 units, but if $\sigma = 7$ $HLD(1, 8)$ accepts the link to a node on same distance with probability of 0.3. At the same time, more conservative detectors such as $HLD(2, 2)$ reject almost all potential links already at 400 distance units. As the consequence, selecting the appropriate parameters of HLD has much higher importance than in the simplified two-ray ground propagation.

4.2.2. Heartbeat Link Detector Behavior in a Network

The transition curve which describes the behavior of an HLD on a single link has limited expressiveness – its analysis allowed us to derive important conclusions on HLDs. However, a network consists of tens, hundreds or thousands of links, each having its own quality, so it is important to assess the HLD performance at a network level and optimize its parameters for accurate detection at the network level.

The transition curve of a $HLD(a, r)$ depends only on values of a and r, but the probability of errors in the link detection process additionally depends on characteristics of the network, in particular on the distribution of link qualities. Probability that there exists a link with quality p in a network is $f_p(p)dp$. If $p < t$, a false positive occurs with probability $P_1(a, r, p)$. Thus, the probability of false positives in the whole network is calculated as:

$$P_{FP}(a, r, t) = \int_0^t P_1(a, r, p) f_p(p) dp \tag{4.7}$$

The probability of false negatives is calculated analogously as:

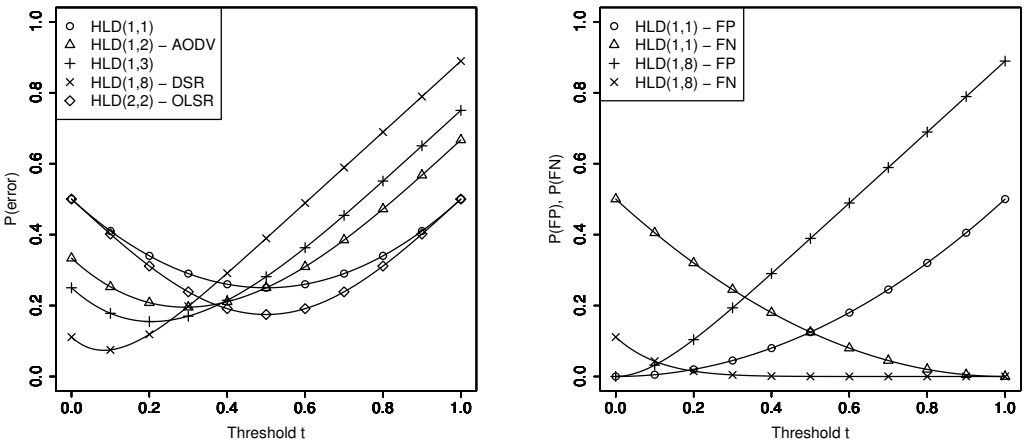
$$P_{FN}(a, r, t) = \int_t^1 P_0(a, r, p) f_p(p) dp = \int_t^1 (1 - P_1(a, r, p)) f_p(p) dp \tag{4.8}$$

Since false positive and false negative events are mutually independent, the error probability is calculated as a sum of these two factors:

$$P_E(a, r, t) = P_{FP \cup FN} = P_{FP}(a, r, t) + P_{FN}(a, r, t) \tag{4.9}$$

The necessary precondition for the error assessment is to know the link quality distribution. In simplest case, it can be assumed that it is uniformly distributed: $f_p(p) = 1$.

For $HLD(1, r)$, $HLD(a, 1)$ and $HLD(2, 2)$ the closed form expressions for the error probability exist. For other combinations of (a, r), the error probabilities have to be calculated numerically. The case where $a = 1$ is of particular interest since AODV and DSR use $HLD(1, 2)$ and $HLD(1, 8)$ respectively. The error probability of $HLD(1, r)$ is:



(a) Error probability.

(b) Probability of false positives and false negatives.

Figure 4.5.: The probability of error, false positives and negatives of a heartbeat link detector in a network with uniformly distributed link quality.

$$P_{FP}(1,r,t) = \int_0^t P_1(1,r,p)dp = p - q^{r+1}r + 1|_0^t = t + \frac{(1-t)^{r+1}-1}{r+1},$$

$$P_{FN}(1,r,t) = \int_t^1 P_0(1,r,p)dp = \int_t^1 (1 - P_1(p))dp = -\frac{(1-p)^{r+1}}{r+1}|_t^1 = \frac{(1-t)^{r+1}}{r+1},$$

$$P_E(1,r,t) = P_{FP}(1,r,t) + P_{FN}(1,r,t) = \frac{t(r+1) - 1 + 2(1-t)^{r+1}}{r+1} \tag{4.10}$$

Figure 4.5(a) shows the probability of errors for several $HLD(1,r)$ detectors. It can be seen that with increase in r, errors are smaller for lower threshold values but increase considerably if threshold is set to high values. Figure 4.5(b) shows the probability of false positives and false negatives. For higher r, the false negatives are rare even for low threshold values (probability of a false negative for $HLD(1,1)$ at $t = 0.1$ is 0.405 while it is only 0.043 for $HLD(1,8)$). Most of the errors of $HLD(1,r)$ detectors with higher r originate from the false positives, which sharply increase for higher values of t. So for instance, $HLD(1,1)$ has P_{FP} of 0.18 at $t = 0.6$ while $HLD(1,8)$ has 0.488 – almost three times higher.

The link quality p and the link acceptance threshold t are defined for a single packet transmission. Due to MAC unicasts and its retries, most of the standard non-real time transport protocols (such as FTP or HTTP) can operate even on links of rather low quality (thus the value of t is also rather low). This is exactly the threshold range where higher r results in smaller error probability in the link detection process, which explains the improved performance of AODV with $HLD(1,3)$ over $HLD(1,2)$ in experiments performed in [52].

Real networks need not to have the uniform link quality distribution but if there exist measurements of link qualities in a network, it is possible to derive empirical probability

	α	λ	k
Leipzig	{0.287, 0.187, 0.15, 0.374}	{16.241, 14.153, 21.732, 24.094}	{1, 4, 8, 16}
Motelab	{0.0904, 0.0581, 0.0916, 0.0768, 0.1883, 0.4948}	{50.529, 73.981, 38.808, 41.638, 51.265, 159.137}	{2, 8, 10, 18, 35, 150}

Table 4.3.: Parameters of the hyper-Erlang distribution for approximation of link quality distributions in Leipzig community network and Motelab testbed.

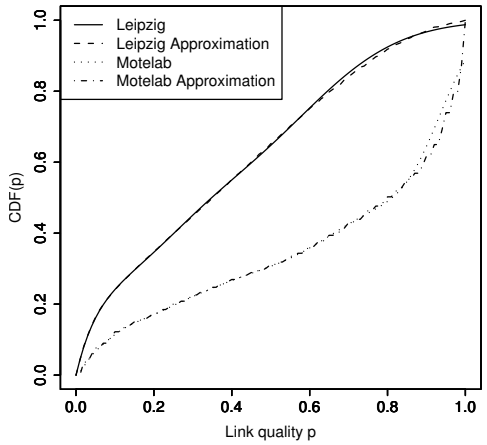


Figure 4.6.: Distribution of link quality in real network and its approximation with a hyper-Erlang distribution.

density function from them. A variety of methods exist for approximation of an empirical distribution with a (set) of closed-form distributions, such as exponential or Weibull distributions. In this work, the measured link quality distribution is approximated by hyper-Erlang distribution [161] which is frequently used in networking [145][161][135]. The hyper-Erlang distribution is actually a weighted sum of Erlang distributions:

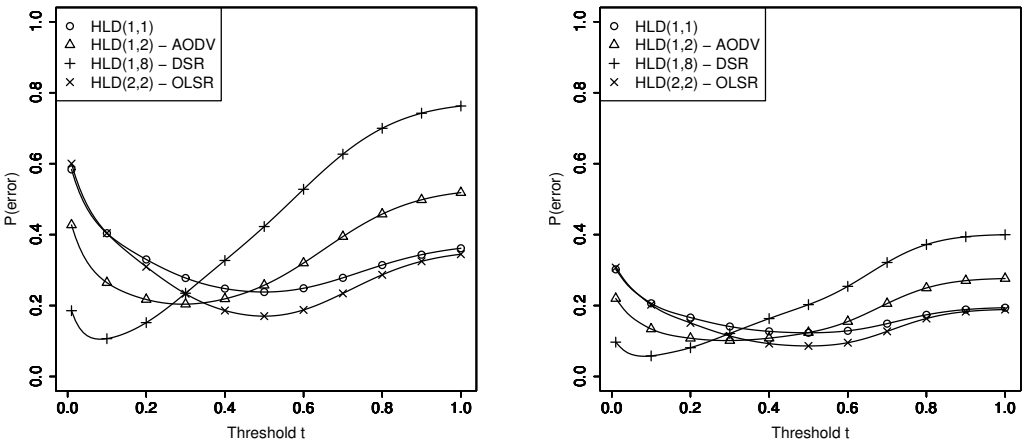
$$f(x,k,\mu,\alpha) = \sum_i \alpha_i \frac{\lambda_i^{k_i} x^{k_i-1} e^{-\lambda_i x}}{(k_i-1)!} \tag{4.11}$$

where the weights α_i sum to one.

The measurements from a community network in Leipzig and Motelab testbed (the measurement methodology is explained in Chapter 7 on page 123 and in Section 9.3 on page 166 respectively) are used as the reference real networks. The differences in probability density functions are notable due to different environment properties, node placement and characteristics of network adapters.

In Figure 4.6 it can be seen that the link quality distribution in Leipzig network has slightly more weight for low values of p, while Motelab samples have most of their weight focused in higher ranges of p. The G-FIT tool [161] was used for fitting and the obtained parameters of the hyper-Erlang distribution are shown in Table 4.3. As it can be seen in Figure 4.6, the hyper-Erlang approximations of the measured distributions provide almost a perfect fit.

The hyper-Erlang distribution (Equation 4.11) with parameters from Table 4.3 is



(a) Leipzig.

(b) Motelab.

Figure 4.7.: Errors in heartbeat link detection in real networks.

substituted in Equations 4.7, 4.8 and 4.9. The integrals are numerically evaluated and the calculated error probability is shown in Figure 4.7 as a function of the threshold t.

If compared with HLD behavior in a network with the uniform distribution of p in Figure 4.5, the error probability curves have different shape and they are generally smaller in both real networks. Additionally, due to prevalence of links with rather high quality in the Motelab testbed, the error probabilities in it are considerably smaller than in Leipzig network where a considerable number of links has low quality.

It is particularly important to notice in Figures 4.5 and 4.7 that errors in link detection process are present regardless of the parameters a, r, t, and the link quality distribution. Errors are not only inevitable but also far from negligible. For instance, the lowest error probability of all HLDs shown in Figure 4.7(a) is $P_E(1, 8, 0.08) = 0.105$. However, if unfavorable parameters a and r are selected for a given network type and threshold value, error probability can be considerably higher. For the same $HLD(1, 8)$, the error probability is larger than 0.74 for $t > 0.9$.

The large variations in error probability of the same HLD, depending on the needed link acceptance threshold t, make the selection of an appropriate HLD a necessity. Assuming that t is known, an adequate HLD should be selected in order to minimize the error probability of the link detection process in a network.

The presented methodology for assessment of P_E enables to select optimal values for a and r so that the probability of errors is minimized. The threshold t, probability density function $f_p(p)$, the set A_c of candidate values for a and set R_c for candidates of r are provided as the input to the evaluation routine. The evaluation routine applies Equation 4.9 to each of members of set $A_c \times R_c$ and finds the best combination from this set. Thanks to the efficient numeric integration algorithms the computational complexity is manageable and the whole process is executed swiftly. For instance, the results in Table 4.5 where $a, r \in \{1..10\}$ have been calculated in the R statistics tool [143] in less than a second per table entry (one hundred possible combinations of (a, r) are evaluated for each table entry) on a commodity PC.

$f_p(p)$		t=0.1	t=0.2	t=0.3	t=0.4	t=0.5	t=0.6	t=0.7
Uniform	(a,r)	(1,5)	**(1,5)**	(2,5)	(4,5)	(5,5)	(5,4)	(5,2)
	P_E	0.11	0.12	0.11	0.1	0.08	0.1	0.11
Leipzig	(a,r)	(1,5)	(2,5)	(2,5)	(4,5)	(5,5)	**(5,3)**	(5,2)
	P_E	0.13	0.14	0.11	0.1	0.08	0.1	0.09
Motelab	(a,r)	(1,5)	(2,5)	(2,5)	(4,5)	(5,5)	(5,4)	(5,2)
	P_E	0.07	0.06	0.05	0.04	0.04	0.05	0.06

Table 4.4.: Optimal HLD(a,r) for a given threshold and network type. $a, r \in \{1..5\}$

$f_p(p)$		t=0.1	t=0.2	t=0.3	t=0.4	t=0.5	t=0.6	t=0.7
Uniform	(a,r)	(1,10)	(2,10)	(4,10)	(6,10)	(10,10)	(10,6)	(10,4)
	P_E	0.06	0.07	0.06	0.05	0.04	0.05	0.06
Leipzig	(a,r)	(1,10)	(2,10)	(4,10)	(6,10)	(10,10)	(10,6)	(10,4)
	P_E	0.1	0.07	0.06	0.05	0.04	0.05	0.05
Motelab	(a,r)	(1,10)	(2,10)	(4,10)	(6,10)	(10,10)	**(10,7)**	(10,4)
	P_E	0.06	0.04	0.03	0.02	0.02	0.03	0.04

Table 4.5.: Optimal HLD(a,r) for a given threshold and network type. $a, r \in \{1..10\}$

The optimal combinations of parameters a and r are presented in Table 4.4 if $a, r \in \{1..5\}$ and in Table 4.5 if $a, r \in \{1..10\}$ for three link quality distributions: the uniform, and the two hyper-Erlang approximations from Table 4.3. In this example, it can be seen that the minimum achievable probability of errors in link detection varies considerably between network types (link quality distributions). For instance, the minimum probability of errors in Leipzig-type of network is two times higher than in the Motelab. As it was already explained in Section 4.2.1, higher values of a and r reduce the errors, so the probability of errors if $a, r \in \{1..10\}$ is up to 0.1. If we compare the optimal HLDs for a given threshold and network type with an arbitrary combination of (a,r), the range of P_E is large: for instance, $HLD(1, 8)$ has P_E at t=0.7 of 0.626 which is 12.5 times more than $HLD(10, 4)$ or 6.95 times more than $HLD(5, 2)$ which are optimized for this threshold.

An interesting property which can be observed in the examples from Tables 4.4 and 4.5 is that for the same threshold, the identical parameter combinations are selected almost always for all three distribution types. Only in three cases there exists a mismatch in parameter selection between distribution types (marked by bold typeface in Tables 4.4 and 4.5). Let us observe the case where t=0.2 in Table 4.4. The optimal HLD configuration for uniform distribution is (1,5) while the optimum for Motelab and Leipzig networks is at (2,5). But even if $HLD(2, 5)$ is chosen in network with uniform distribution of p instead of $HLD(1, 5)$, the P_E is increased from 0.121 to 0.132 or approximately 10% only. The differences are even smaller in other cases: for t=0.6, if $HLD(10, 6)$ (optimal for uniform and Leipzig distributions) is applied in Motelab instead of $HLD(10, 7)$, the change in error probability is negligible – only 0.6%. The consequence of this observation is that even if link quality distribution of a network is unknown, it is worth to determine the values for a and r on a known distribution, knowing that the variation from optimum is acceptable.

4.3. Node Failures and Limited Duration of Link Existence

In addition to the steady state HLD behavior evaluated in the previous section, it is important to address HLD behavior in presence of node failures when the duration of link existence is limited. If link exists for a limited time, transient behavior of heartbeat link detectors needs to be considered.

Limited link existence duration can be used as a model of topological changes introduced by node mobility, where links are created and destroyed as nodes move. It has been derived in [148] that the duration of link existence in a mobile WMNs can be modeled with the exponential distribution $\mu e^{-\mu d}$ where μ is the rate and $\frac{1}{\mu}$ is the average link existence duration. For typical mobile WMNs, $\frac{1}{\mu}$ is in range of tens of seconds.

Instead of explicitly working with time, this section adopts the approach of heartbeat ticks (or rounds): if a link exists for duration d and heartbeats are sent with frequency f_{hb}, the number of heartbeats exchanged during link existence is $\lfloor \frac{d}{f_{hb}} \rfloor$ ticks. Such approach simplifies the presentation of models without loss in detail or quality of the developed models.

As soon as the link duration is limited, the number of rounds an HLD needs to recognize this existence starts affecting the probability of declaring the link as active. In the simplest case of a perfect link which exists for duration d, a $HLD(a, r)$ can declare it as active only for $d - a$ ticks, resulting in probability of link acceptance of $P_1 = \frac{d-a}{d}$ (obviously, for finite a, as $d \to \infty$, $P_1 \to 1$). For $p < 1$ the probability of link acceptance will be further reduced, since a longer sequence of attempts may be necessary before a series of a heartbeat receptions is observed.

Let us define a "successful run" as a series of a successive heartbeat receptions on a link. The length of sequence of reception and loss events on a link before the successive run occurs is the waiting time (WT) of a successful run. Feller [72] has applied renewal theory to show that the average waiting time of a successful run is:

$$E(WT(a, p)) = \frac{1 - p^a}{(1 - p)p^a} \tag{4.12}$$

Figure 4.8 shows the average waiting times for different values of parameter a as the function of link quality p. As the parameter a is increased, the waiting time sharply rises for low values of link quality p. For instance, if $a = 9$, the waiting time for acceptance of a link with quality $p = 0.5$ is 1022 rounds which renders its detection practically impossible in a mobile network. For the same link and $a = 5$, the waiting time is 62 which is still long but it may be acceptable under low mobility.

If the waiting time for acceptance of a link is longer than its existence, it prevents its detection. This property of HLDs in mobile networks may have both positive and negative effects on the detection process, depending on the selected threshold:

- If threshold t is low, mobility has negative effects since the probability of false negatives increases due to the long detection waiting times of links with low quality.
- If threshold t is high, mobility effects are positive. Probability of false negatives is low anyway (see Figure 4.5(b)) but the waiting times reduce probability of false positives and thus the total error probability.

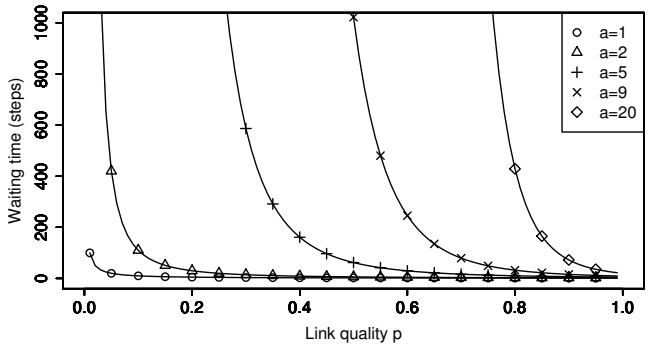


Figure 4.8.: The average waiting time of a successful run for link acceptance.

The average waiting time for the rejection of link existence hypothesis can be calculated from Equation 4.12 by replacing p with q, and a with r. However, the effects of link rejection waiting times to HLD behavior are not the same as for its detection. At the start of link existence, the acceptance waiting time depends on a and p as it is defined in Equation 4.12, but the rejection waiting time of a link that was marked as active at the time of its failure is up to r ticks.

An important consequence of long waiting times for a successful run is that it may not be feasible to construct heartbeat detectors with a sharp transition curve in mobile WMNs. As explained in the previous section, parameters a and r should be set to rather high values in order to obtain sharp transition and reduce probability of errors. However, the Equation 4.12 clearly describes the strong bias against the hypothesis of link existence made by a heartbeat detector with the sharp transition curve if it is applied on a link of limited duration.

The analysis of the average waiting time indicates additional issues in link detection in presence of mobility, but it cannot be used for exact error analysis. In order to assess HLD characteristics in a mobile network, a similar analysis as in the previous section will be performed, starting with calculation of P_1.

Let the vector S_0 be the vector of initial probabilities of states and let P be the transition matrix of a Markov chain. Based on the theory of Markov chains [72], it is known that the vector $S^{(k)}$ of state probabilities after k transitions in the chain is:

$$S^{(k)} = S_0 \cdot P^k \tag{4.13}$$

The transition matrix of an HLD is given in Table 4.2 and the vector S_0 is [1 0 0 .. 0] since a HLD always starts from state 00. Assuming that a link exists for exactly d ticks, we can calculate the probability $P_0(a, r, p, d)$ of rejection of link existence hypothesis as:

$$P_0(a, r, p, d) = \frac{1}{d} \sum_{i=0}^{d} \sum_{j=0}^{a-1} P_{0j}^{(i)} \tag{4.14}$$

where $P_X^{(i)}$ is the probability of state X after i transitions in the chain, $X \in \{00, 01, ... , 0(a-1), 10, 11, ..., 1(r-1)\}$. The first sum iterates through all ticks during link exis-

tence.[2] The second sum adds the probabilities of states that reject the link existence hypothesis – states 00 to $0(a-1)$. Additionally, it is assumed that ticks are equidistant so the result is multiplied by the factor $\frac{1}{d}$. Since the transition matrix P of an HLD and vector of initial state probabilities S_0 are known, Equation 4.13 allows us calculation of $P_{0j}^{(k)}$, which after substitution in Equation 4.14 yields:

$$P_0(a,r,p,d) = \frac{1}{d}\sum_{i=0}^{d}\sum_{j=0}^{a-1}(S_0 \cdot P^k)_{0j} \tag{4.15}$$

The probability of link acceptance is:

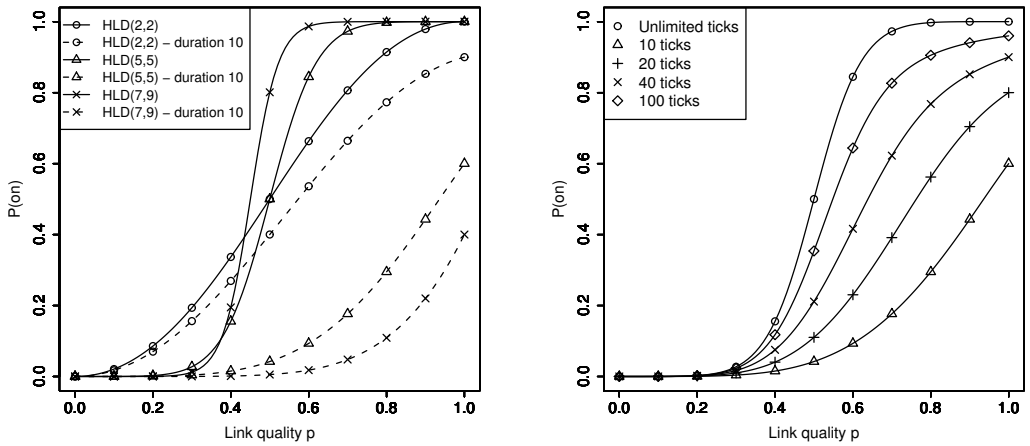
$$P_1(a,r,p,d) = 1 - P_0(a,r,p,d) = \frac{1}{d}\sum_{i=0}^{d}\sum_{j=0}^{r-1}(S_0 \cdot P^i)_{1j} \tag{4.16}$$

The behavior of HLDs in presence of mobility from Equation 4.16 are shown in Figure 4.9. Figure 4.9(a) compares the behavior of different HLDs if they are applied to a permanent link and to a link which exists for ten ticks. The differences induced by limited link existence duration are particularly large for the HLDs which are excellent for detection of permanent links (i.e., with high values of a and r,as explained in Section 4.2.1). For instance, the $HLD(7,9)$ detects practically all permanent links with quality larger than 0.5, but if a link exists for ten ticks, $HLD(7,9)$ can only detect links of exceptional quality, with a rather low probability. Since it needs seven successive heartbeat receptions for acceptance of link existence hypothesis, even if link is ideal with quality one, $HLD(7,9)$ declares it as functional in only three ticks out of ten, resulting in $P_1(7,9,1,10) = 0.3$.

Figure 4.9(b) demonstrates behavior of $HLD(5,5)$ if it is applied on links of various duration and compares it with the transition curve of $HLD(5,5)$ at a permanent link. As expected, longer link existence brings the transition curve of $P_1(a,r,p,d)$ closer to the transition curve of $P_1(a,r,p)$ derived for the permanent link. The improvements (in terms of getting closer to the transition curve of a permanent link) are much more pronounced at the beginning – notice the huge difference between curves for 10 and 40 ticks, and much smaller improvement between curves for 40 and 100 ticks.

The error probability is defined at time segment $[0, d+r]$ and consists of two components. The first component originates from errors occurring during HLD operation while the link is still active. It is defined on time interval $[0,d]$. It is similar as in static networks (Equation 4.9) and can be caused both by false positives and negatives. The second component is caused by false positives that can occur after failure of the link – if link is recognized as active at d^{th} tick, an HLD will continue supporting this hypothesis until it observes r successive heartbeat omissions. The number of ticks needed for rejection of link existence hypothesis after its failure depends on the state in which HLD was at the time of d^{th} tick. For instance, if HLD was in state 12 in d^{th} tick, it needs $r-2$ additional ticks to declare it as failed.

[2]For presentation simplicity, it is assumed that frequency of heartbeats $f_{hb} = 1Hz$. It is easy to generalize developed models for $f_{hb} \neq 1Hz$.



(a) P_1 of different HLDs for link existence duration of ten ticks.

(b) P_1 of HLD(5,5) for varying link existence duration.

Figure 4.9.: The probability of link acceptance P_1 for links with limited duration of existence.

Thus, probability of these additional false positives after link failure P_{FP-f} is:

$$P_{FP-f}(a,r,t,d) = \frac{1}{d+r}\sum_{i=0}^{r-1} P_{1i}^{(d)}(r-i) \tag{4.17}$$

where the sum counts the number of false positives and the factor $\frac{1}{d+r}$ normalizes it to a probability.

The probability of erroneous link detection for a network with distribution of link qualities $f_p(p)$ for link duration d is:

$$P_E(a,r,t,d) = P_{FP\cup FN\cup FP-f} = \frac{d}{d+r}\Big(\int_0^t P_1(a,r,p,d)f_p(p)dp$$

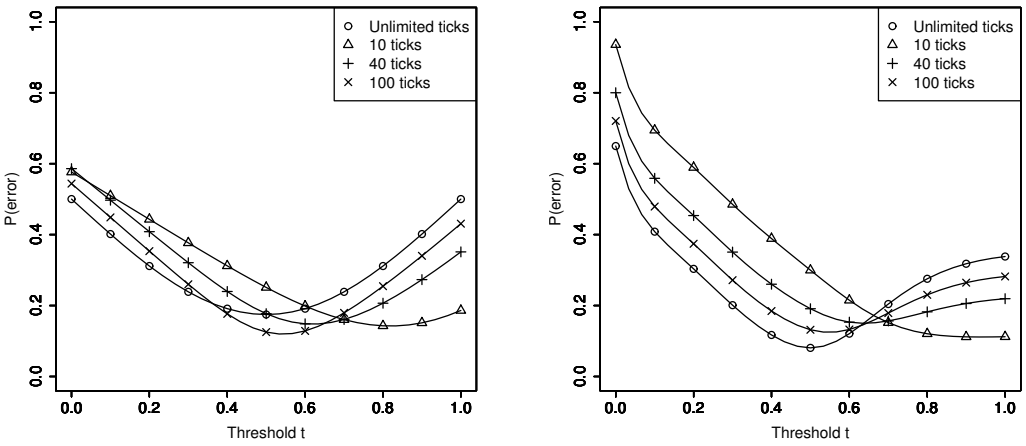
$$+\int_t^1 (1-P_1(a,r,p,d))f_p(p)dp\Big) + \frac{1}{d+r}\sum_{i=0}^{r-1} P_{1i}^{(d)}(r-i) \tag{4.18}$$

where $P_1(a,r,p,d)$ is defined in Equation 4.16.

The empirical distribution of link qualities can be determined in the same manner as in Section 4.2.1 from the measurement set from a mobile network.

In this section are used the uniform and the link quality distribution from the Leipzig network. Although the Leipzig network is static, its link quality distribution is considered acceptable in this analysis because packet losses at a link are caused by environmental effects (e.g., obstacles, multipath signal propagation) that is captured in $f_p(p)$ while the effects of link breaking due to increase of internode distance are already modeled by the limited duration of link existence.

Figure 4.10 shows effects of limited link existence duration on the error probability



(a) The uniform distribution of link quality.

(b) The Leipzig-network distribution of link quality.

Figure 4.10.: The probability of errors under mobility for HLD(5,5).

for uniform and Leipzig-like distribution of link quality. In both cases, two clear trends are observable: if the link acceptance threshold is low, the probability of errors for links with short existence increases considerably if compared with P_E of a permanent link. However, for high values of threshold, the limited link existence duration has positive effects on the error probability and reduces it. This feature is particularly pronounced for links with a short duration. The reasons for this can be seen in Figure 4.11. For high values of the threshold t, the probability of false negatives is small regardless of the link duration and most of the errors originate from false positives. The limited link duration reduces the probability of false positives since it is more difficult for a link with lower quality to be recognized as active in such a short time (Equation 4.12, Figures 4.8 and 4.9(a)) and P_E is reduced.

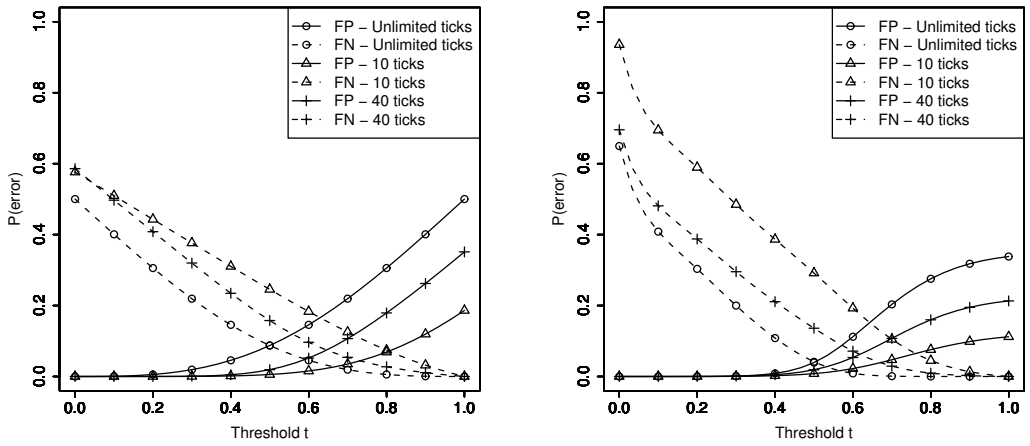
Equation 4.18 provides the error probability only for links with the duration d. In a dynamic network, links have varying existence duration, which can be described by probability density function $f_d(d)$. The distribution of link existence durations in mobile WMNs $f_d(d)$ was derived in [148], which can be used for calculation of error probability in link detection process:

$$P_E(a, r, t, \mu) = \int_0^\infty P_E(a, r, t, d) f_d(d) dd \tag{4.19}$$

If we set $f_d(d) = \mu e^{-\mu d}$ and replace continuous time for discrete ticks, the integral is transformed into a sum, yielding:

$$P_E(a, r, t, \mu) = \sum_{d=0}^{\infty} (P_E(a, r, t, d)) \mu e^{-\mu d} \tag{4.20}$$

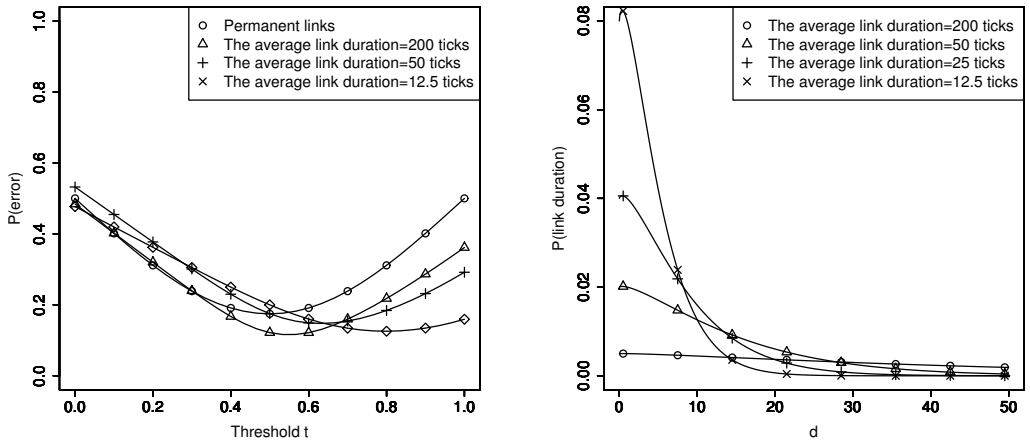
where $P_E(a, r, t, d)$ is defined in Equation 4.18 under assumption that the link duration and the link quality distributions are independent.



(a) The uniform distribution of link quality.

(b) The Leipzig-network distribution of link quality.

Figure 4.11.: The probability of false positives and false negatives under mobility for HLD(5,5).



(a) The uniform distribution of link quality, variation of link duration rate, $HLD(5,5)$.

(b) The distribution of link duration d by [148].

Figure 4.12.: HLD behavior in a mobile network. The uniform distribution of link quality, varying link duration rate.

The important difference between Equations 4.18 and 4.20 is that the former calculates the error probability as a function of a single duration d, while the later is calculated for the whole probability distribution function of duration, capturing the HLD error probability in a mobile WMN.

Figure 4.12(a) shows the error probability of $HLD(5,5)$ if applied to networks with varying degrees of mobility. The distribution of link existence duration for these net-

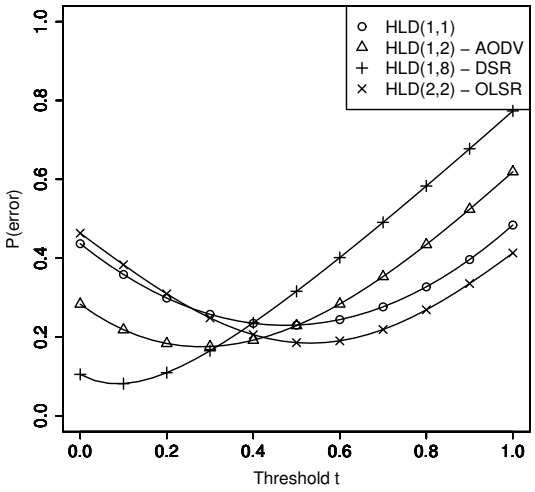


Figure 4.13.: The error probability of different HLDs for the average link duration of 25 ticks.

		t=0.1	t=0.2	t=0.3	t=0.4	t=0.5	t=0.6	t=0.7
$\mu = 0.08$	(a,r)	(1,5)	(1,5)	(1,4)	(1,2)	(1,1)	(2,2)	(3,2)
	P_E	0.19	0.17	0.19	0.21	0.24	0.23	0.2
$\mu = 0.04$	(a,r)	(1,5)	(1,5)	(1,4)	(1,2)	(2,3)	(3,2)	(4,2)
	P_E	0.17	0.16	0.18	0.21	0.22	0.21	0.18
$\mu = 0.02$	(a,r)	(1,5)	(1,5)	(1,4)	(2,4)	(3,4)	(4,3)	(5,2)
	P_E	0.145	0.143	0.17	0.19	0.2	0.18	0.16

Table 4.6.: Optimal HLD(a,r) in mobile networks for a given threshold and link duration ratio. $a, r \in \{1..5\}$.

works is shown in Figure 4.12(b). The error probability of a mobile WMN follows conclusions drawn from Equation 4.18. So, if the link acceptance threshold is low, the probability of errors for links of short duration is higher than the P_E of a permanent link, and the roles are reversed for high values of the threshold.

The error probabilities depend both on link existence duration distribution and the parameters of HLD as it can be seen in Figure 4.13.

The derived expressions allow us to determine the optimal HLD configurations in the same manner as in Section 4.2.2. The optimal values of HLD parameters for networks with the different link duration rates are shown in Table 4.6. If we compare the results of optimization in mobile (Table 4.6) and in static networks (Table 4.4), the error probabilities are higher in mobile networks. The second and more important difference is in parameter values. In static networks, it is always beneficial to put either a or r to the maximum (value 5 in this example) but in a mobile network this is no longer the case: with increasing mobility of nodes, the values of a and r are reduced. This characteristics can be clearly seen for t=0.5 where $HLD(1,1)$ is optimal for $\mu = 0.08$, $HLD(3,4)$ is optimal for $\mu = 0.02$ while $HLD(5,5)$ provides the optimum for static networks. As

the link duration rate decreases and their average existence time increases, the optimal HLD values are moving toward values determined for the static network, and the error probability of link detection decreases.

Table 4.6 sheds light on the selection of HLD parameters of routing protocols from RFCs which have been criticized in the previous section. The analysis of HLD behavior in mobile WMNs has shown that the values from RFCs which were unsuitable for static networks are much closer to the optimal values of mobile networks.

4.4. Effects of HLD Errors on Proactive Topology Management Protocols

In proactive topology management protocols, HLDs are applied in a network consisting of hundreds of edges. The probability of erroneous detection of e_{err} edges in a network of $|e|$ edges with a $HLD(a, r)$ with error probability $P_E(a, r, t)$ has the binomial distribution $B(|e|, P_E(a, r, t))$, assuming the independence of errors in link detection.

Since the number of edges in a network is rather large even in middle sized networks and the error probability in detection of as single link is far from trivial, it is possible to approximate the binomial distribution $B(|e|, P_E(a, r, t))$ with the normal distribution $N(|e| \cdot P_E(a, r, t), |e| \cdot P_E(a, r, t) \cdot (1 - P_E(a, r, t))$.

The approximation is the consequence of the central limit theorem. The precision of the approximation depends on the parameters of the Binomial distribution. The approximation of $B(n, p)$ with $N(np, np(1 - p))$ is considered accurate if both np and $np(1 - p)$ are larger than five which is fulfilled even for medium sized networks. For instance, if a HLD with $p = P_E = 0.05$ is applied in a network with 50 nodes with the average node degree of five, $|e| = 50 \cdot 5/2 = 125$, yielding $np = 6.25$ and $np(1 - p) = 5.9375$.

Let us observe HLDs from Table 4.4: $HLD(2, 5)$ for $t = 0.3$ and $HLD(5, 5)$ for $t = 0.5$. The probability of having no errors in link detection in a network where they are used is very low: if $|e| = 50$ edges and $P_E(5, 5, 0.5) = 0.11$, the probability of having all edges correctly detected is 0.006, if $|e| = 200$ this probability is $3.3 \cdot 10^{-7}$. Despite the optimally selected HLDs, the number of erroneous link detection is considerable. Its mean value is $|e| \cdot P_E$ and grows linearly with increase in number of edges. In a network with 600 edges there will be 48 erroneous detections on the average for $t = 0.5$ and 66 errors for $t = 0.3$. This behavior is depicted in Figure 4.14 which shows the cumulative distribution function of number of edges which are erroneously detected in networks which consist of 300 or 600 edges.

HLDs are one of the building blocks of proactive management protocols and HLD errors are directly transferred to them. The inaccuracies in topology discovered by a proactive topology management protocol have important consequences for all topological algorithms applied in WMNs which require accurate global topological knowledge. Most of the existing algorithms that operate on topologies were not developed to operate under uncertainty and they assume correctness of the topology on which they operate. So, although the algorithms are correct itself, their decisions are incorrect since they are applied to a topology that is incorrect (different than the topology seen by an omniscient observer).

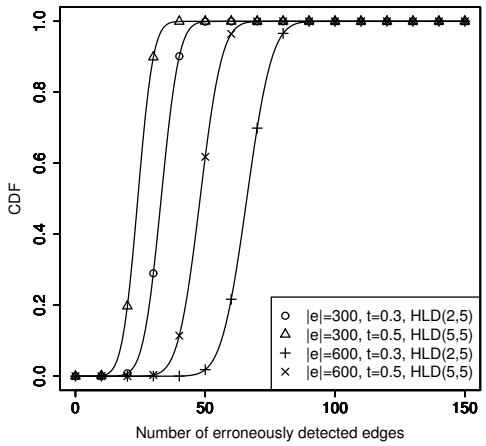


Figure 4.14.: Effects of HLD induced errors on proactive topology management in networks of different size. Optimal HLD parameters are used for each of thresholds.

4.5. Summary

In this chapter, the behavior of heartbeat link detectors in presence of heartbeat losses has been evaluated with the goal of estimating the probability of erroneous link detection. A model applicable to static and dynamic (mobile) networks has been developed. It is shown that errors in link detection inevitably exist no matter which HLD is applied in a network.

The definition of the link acceptance threshold in Section 2.2.1 has been left deliberately vague as "application dependant". As such, it left some open questions with regard to its impact on topology detection and whether its choice may eliminate the errors in the link detection process. Important result of the analysis performed in this chapter is that the threshold selection cannot eliminate errors in link detection process.

The evaluation methodology developed in this chapter allows the optimal selection of HLD parameters so that the errors in the link detection process are minimized. The methodology is more efficient than simulation. It is flexible and has been demonstrated how to apply it to two real network datasets.

Even with the optimally selected parameters, the probability of erroneous detection of a link remains non-negligible. For example, in a static network, within the candidate set $(a, r) \in \{1..5\} \times \{1..5\}$ for $t = 0.3$ the optimal detector is $HLD(2, 5)$ and its error probability is 0.11. In the same example for threshold $t = 0.5$ the optimal detector is $HLD(5, 5)$ and its error probability is 0.08. The HLD used in DSR routing protocol provides a good example of improvements brought by optimal parameterization of HLDs – its error probability can be more than 12 times larger for some link acceptance thresholds than of a designated HLD with optimized parameters for the threshold and network type.

If errors in output of an HLD are to be kept within acceptable range, its parameters have to be selected appropriately to fit the network characteristics and the link

acceptance threshold. This necessity of using the optimal HLD parameters can result in undesirable consequences. Optimal parameters for one network type and link acceptance threshold may be completely inappropriate for a network with different characteristics. For instance, it was observed that the optimal HLD parameters for mobile networks produce high probability of errors in link detection in static networks. So, even if a network uses optimally configured HLDs, its functionality and performance may get compromised if characteristics of the network change during its lifetime (e.g., a subset of its nodes become mobile) or if an application with a different link quality demands (and acceptance threshold) is deployed in the network. Optimal values from the previous setup may render network unusable in the new setup, a new combination of optimal parameters has to be derived, and network nodes reconfigured.

The effects of HLD errors on behavior of proactive topology management protocols are profound. In proactive topology management protocols, HLDs are applied in a network consisting of hundreds of edges. The probability of erroneous detection of e_{err} edges in a network has the normal distribution $N(|e| \cdot P_E(a,r,t), |e| \cdot P_E(a,r,t) \cdot (1 - P_E(a,r,t))$. The mean value of the number of erroneous link in a network consisting of e edges is $|e| \cdot P_E$ and it grows linearly with increase in number of links in network.

The inaccuracies of HLDs are transferred to proactive topology management protocols so the topologies they deliver to network protocols are inaccurate. Such inaccurate topologies have important consequences for all topological algorithms applied in WMNs. Vast majority of the existing algorithms that operate on topologies are not created to operate under uncertainty – precondition for their application is the accurate topological knowledge of a graph on which they are executed. In context of this work, decisions of the existing biconnectivity testing algorithms which are applied on topology delivered by a proactive topology management protocol will be erroneous as the consequence of erroneous topologies provided by HLDs and proactive topology management protocols.

5. Distributed Bridge and Articulation Point Detection Algorithm for Wireless Networks (DIBADAWN)

An approach to detection of bridges and articulation points in wireless multi-hop networks is described in this chapter. A distributed algorithm is presented that combines the approach of the Echo algorithms [55][67][68] and Tarjan's DFS. The algorithm introduces numerous modifications in order to utilize the advantages provided by WMNs and to comply with limitations imposed by them.

DIBADAWN algorithm detects all bridges and articulation points in a network under ideal conditions (nodes and links do not fail, messages are delivered reliably). However, node failures and message losses are inevitable in wireless networks and they cause faults in the algorithm. The faults and their effects have been analyzed in detail and the detection algorithm was extended in order to reduce impact of faults. The changed algorithm is able to guarantee some properties in presence of packet losses such as the termination, but due to unpredictability of message losses and node failures it cannot guarantee correctness of decisions. Since it is not possible to eliminate effects of all faults caused by the environment, voting theory is applied in order to improve the accuracy of the detection algorithm.

An important characteristic of the approach proposed in this chapter is rejection of the notion of having the same perception of the network topology at each of its nodes, which is a precondition for bridge and articulation point detection algorithms which rely on global topology knowledge. The global knowledge is powerful and useful paradigm in networks with reliable communication, but it is counterproductive in systems loaded with uncertainty such are the WMNs. The proposed voting schemes exploit the diversity of knowledge about network's topology obtained from successive searches at each of network's nodes in order to reduce probability of erroneous decisions.

5.1. Introduction

Numerous solutions for biconnectivity testing have been proposed in graph theory in the past [55][67][108][158][162]. In a graph or in an omniscient[1] network, the search for biconnected components is very efficient – for instance, a slightly modified depth first search identifies bridges and articulation points in $O(n+e)$ time, where n is the number of nodes and e is a number of edges. Biconnectivity testing performance has been further improved in parallel systems. For instance, [130] proposes an algorithm that is capable of biconnectivity testing in $O(log_2 n)$ time using $O((n+e)/log_2 n)$ processors.

[1] Perfectly accurate topology knowledge. The cost of obtaining it is disregarded.

If biconnectivity testing is applied in WMNs, the main concern is no longer its efficiency in terms of execution complexity. The major issue is that omniscient communication networks do not exist. Providing up-to-date topology information is costly and daunting task in a large distributed system, not always possible to accomplish even in wired networks [102].

Obtaining accurate topology of a WMN is particularly challenging: wireless links are instable, they have higher delay, lower throughput and an order of magnitude higher unreliability than their wired counterparts. Unpredictability of communication link characteristics caused by environment, physical limitations of the channel and receiver, unreliability and mobility of nodes, all put high strain on a topology management protocol. Furthermore, dissemination of topology data consumes resources, increases contention on the air and does not scale well. For instance, the authors of [57] have noticed scalability issues of their method for partitioning prediction which relies on the centralized bridge detection algorithm and global topology knowledge.

It was shown in the previous chapter that the existing link detection mechanisms are unable to accurately detect links to other nodes in a WMN because of the uncertainty introduced by unreliable wireless communication channel, Thus, the local topology data which is disseminated through the network may be already erroneous at a node which initiates the dissemination.

Combination of issues in link detection and information dissemination in WMNs creates serious issues for proactive topology management protocols and the topology they deliver to nodes is inaccurate with high probability (Section 4.4, page 63). The existing biconnectivity testing algorithms are intended to be applied to the global topology view of the network and they are not made to operate under uncertainty or in inaccurate topologies. If they are applied to partially incorrect topology, they deliver incorrect decisions on bridge and articulation point existence. So, instead of having correct decisions like in traditional biconnectivity testing algorithms from graph theory, the decisions in algorithms applied to WMNs are correct only with a certain probability. This is not a particular characteristics of a biconnectivity testing algorithm but the consequence of unreliable communication in WMNs.

The simulation study in Section 9.2 confirms that proactive topology management cannot provide sufficiently accurate topology knowledge for bridge and articulation point detection. As a consequence of inaccuracies in topology knowledge only the decisions local to a node[2] are of some practical importance.

In this work, a distributed approach to biconnectivity testing is chosen. The goal is to create less communication overhead than its proactive counterparts, and simultaneously to detect bridges and articulation points with similar or better accuracy.

Figure 5.1 shows the complete proposed approach to bridge and articulation point detection. In its core is DIBADAWN. It combines the ideas of Echo algorithms and of Tarjan's DFS biconnectivity testing, and extends them in order to produce a distributed biconnectivity testing algorithm that is capable of operation in WMNs. The algorithm builds a tree from the connectivity graph and detects the cross-edges in it. If it encounters a cross-edge, it has found a cycle in the graph. All edges that belong to that cycle are subsequently marked as ordinary edges (they are not bridges). Articulation point detection is somewhat different since a node may belong to multiple cycles and still be

[2] A node decides whether it is an articulation point and if its incident links are bridges in the network.

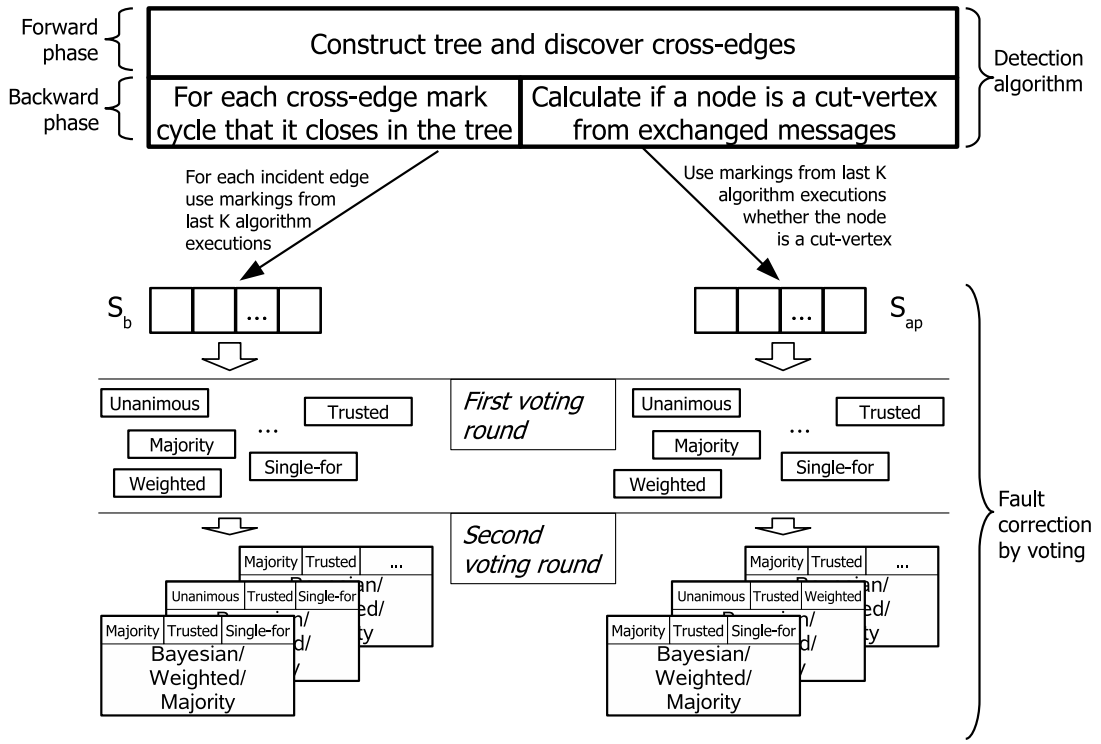


Figure 5.1.: Overview of the proposed approach.

an articulation point. Similar as in work of Tarjan [158], equivalence classes of relation "edge belongs to a cycle" are used to detect them [31]. An example of algorithm's execution can be found at page 82.

The algorithm is tailored so that it is suitable for use in wireless networks. For instance, in graph theory is taken for granted that immediate neighbors of a node are known, while in WMNs they may be unknown prior to execution of the algorithm. Finally, algorithm can detect some communication faults and reduce their effects on detection accuracy.

The inevitable packet losses and node failures in WMN environment cause the algorithm to deviate from its correctness. Two voting rounds are introduced in order to improve accuracy of decisions. A small set S of latest algorithm decisions is preserved and used as the input to the first voting round. Various voting rules are applied to this set S, such as the unanimous, single-for, and majority rule. For instance, the unanimous voting rule requires of all markings in the set S_{ap} to agree that a node is an articulation point in order to support that hypothesis. The voting procedures and rules from this round are described in detail in Section 5.6.

Voting rules from the first round produce results that differ one from another and consequently have different precision and recall. In Section 5.6.3 is explained how the second voting round profits from these differences and further improves the accuracy of decisions through increase in utilization of information delivered by the detection algorithm. For instance, majority voting can be applied to outcome of the unanimous, weighted and trusted voting rules in order to produce the final decision whether an edge is a bridge.

In addition to the majority rule, the Bayes classifier [72] and the weighted voting rule [131] can be used in the second voting round. Such rules require certain apriori knowledge (they require execution of a learning phase before they can be used) but their decisions should provide even higher increase in precision and recall.

The chapter is organized as follows. Applicability of existing biconnectivity testing algorithms in WMNs is assessed in Section 5.2. It is shown that in addition to the issue of compromised correctness of their decisions, there exist control flow and termination issues in some of the approaches. In Section 5.3 are introduced the necessary changes to the Echo algorithm in order to enable its execution in WMNs. The changes are presented in detail in Section 5.4. The proof of correctness of DIBADAWN can be found in Appendix B. The issues introduced by unreliable communication channel, which are unavoidable in reality, are analyzed in detail in Section 5.5. Since fault removal is not possible in a system where decision makers have such a limited knowledge about the system as they have it in this case, a fault masking mechanism in form of voting rules is proposed in Section 5.6.

5.2. Biconnectivity Testing Algorithms in Context of Wireless Multi-hop Networks

Most of the biconnectivity testing algorithms can be adapted for distributed execution in a network which provides reliable communication even if they are not originally envisioned for such use: method calls, checking a state of an adjacent node and returning values to the caller are performed by inter-node message passing; the remainder of the code is executed unchanged at nodes. However, not all of the candidates are equally adept at operation in wireless environment.

The wireless communication channel has the following characteristics which influence execution of distributed algorithms:

1. Neighbors of a node are not necessarily known. Either an additional algorithm for their detection must be employed (e.g., a HLD), or the distributed algorithm itself must perform the discovery.

2. A MAC broadcast visits all neighbors simultaneously, even the already visited neighbors.

3. If algorithm requires visits of neighbors in a specified order, it is implemented as a series of MAC unicasts.

4. Message exchange may be overheard by a subset of neighbors of nodes that communicate. While one node is sending a message, its neighbors should be silent. If they also transmit, messages may be lost.

5. It is not possible to guarantee message delivery, even if retries in message delivery are employed. As a consequence, inconsistent algorithm states may develop in the network.

The distributed biconnectivity testing algorithm should maximize the utilization of the MAC broadcast (2) in order to reduce number of sent packets. Proactive discovery

of links to neighbors is to be avoided due to its communication overhead. MAC unicasts (3) and ordered node visits in the exploration phase of the algorithm are undesirable due to increase in number of sent messages and contention on the channel.[3] The state inspection between nodes should be also minimized in order to reduce channel contention and probability of packet losses (4) caused by it. Even if the number of messages generated by the detection algorithm is kept at minimum, it is still possible that messages are lost (5) either because of the collisions on the channel or because of the shadowing and fading of the channel, so the algorithm must possess certain resilience to message losses.

DFS and BFS are taken as two primary candidates. DFS is one of the best-known algorithms for bridge and articulation point detection in graphs so it is tempting to adapt it for distributed execution in WMNs. Breadth First Search (BFS) also produces a tree in its execution and detects all cross edges. Additionally, it is frequently used in existing WMN protocols[4] where it is also known as flooding: each node in the network rebroadcasts the message exactly once, upon its first reception.

dBFS fully utilizes the broadcasting nature of wireless medium: it is not necessary to know whether there are any nodes in the neighborhood – a MAC broadcast reaches them all (assuming that packet is not lost on the channel). This simultaneously transfers the flow control and discovers neighbors.

dDFS needs to select exactly one node that will continue the algorithm execution. Regardless of the neighbor selection method, a node that executes dDFS must know its 1-hop neighborhood, introducing mandatory proactivity to the tree-construction phase. Detection of 1-hop neighborhood can be easily performed by HLDs, but as it has been shown in Chapter 4, HLDs cannot guarantee correctness of their decisions. HLDs produce both false positives and false negatives in link detection. Thus, dDFS may ignore existing neighbors due to false negatives in link detection, possibly causing false positives in bridge and articulation point detection. A node may also attempt to communicate with a neighbor over a link with insufficient quality (link quality is below acceptance threshold) possibly causing false negatives in bridge and articulation point detection. A false positive in link detection caused by a failed node (HLD did not realize the failure at the moment of link's use) is not an issue since a node will recognize its failure once it attempts to communicate with such non-existent neighbor. dBFS may also produce false positives and negatives in the detection process due to similar reasons.

A much more serious issue for the dDFS are the effects of packet losses on its control flow. DFS blocks a node in the forward phase (tree discovery), and node remains blocked until its parent in DFS tree returns control of the algorithm to it. If applied distributively in a network with unreliable communication channels, control sequences can be lost and the node can remain in the blocked state.

Let us observe two dDFS walkthrough orders rooted at node A in Figure 5.5(a) (page 82): ABHCEFGD and ABCEFGDH, and assume that no messages are lost. If the node B first visits its neighbor H, it gets a response immediately. However, if it visits the node C first, waiting time before it unblocks and visits the node H will be longer

[3] A single MAC broadcast of a node A reaches $d(A)$ nodes. At least $d(A)$ transmissions are required to reach the neighbors of node A by MAC unicasts.

[4] To avoid possible confusion in the text, general functionality of algorithm and the centralized (applicable to complete topology knowledge) implementations are named DFS and BFS and their distributed counterparts are dBFS and dDFS.

since dDFS visits five nodes instead of one. If there would exist a subnetwork with hundreds of nodes beyond node G, it would increase the waiting time for unblocking of node B proportionally. If communication is reliable, a node patiently waits until it eventually receives an answer from its tree-child. In a real network, messages can be lost in either forward (parent to child) or backward phase (child to parent) of the dDFS and a node may remain blocked indefinitely long, preventing termination of the algorithm's execution. MAC unicast of dDFS messages reduces the occurrence of such events at a cost of increased contention on the channel and communication overhead, but it still cannot resolve the problem completely nor provide adequate robustness.

For example, if dDFS starts at node B and decides to visit node H as the first one in the whole search and that message is lost, it remains blocked and cannot continue searching through the larger part of the network. In dBFS, if the node H does not receive message, the search will be continued in remainder of the network independently of the loss event at the link BH, assuming that at least one of the nodes A, C, or D have received the message.

The correctness of dDFS can be sacrificed in order to ensure its termination by introduction of a timeout mechanism: if no response from a successor in DFS tree is received for a long time, a node reclaims the control over dDFS execution and continues it at some other neighbor. However, it is unclear when to select a different neighbor (how to determine the timeout duration) and resume with the execution of dDFS based on local information that is accessible to node – it does not know the size and structure of the network beyond its immediate neighbors. Use of dDFS timeouts would introduce additional complexity to dDFS algorithm, since a methodology would have to be devised that resolves conflicts if two or more execution flows exist in the network instead of one: in case that a node interrupts one execution flow and starts an additional flow although the first execution flow is still successfully executing somewhere in the network.

There exists additional control flow issue which is not limited only to dDFS but it affects every distributed algorithm applied in networks with unreliable communication channel if the algorithm requires transfer of exclusive right to continue execution of the algorithm between pair of nodes.

In order to perform the control flow transfer from node A to node B, both nodes must reach consensus on the transfer of control. Let us assume that node A is supposed to transfer the control to node B. It sends a message to B initiating the transfer, and pauses the execution. Node A needs a confirmation (acknowledgement) from B, since the message to B may have been lost, or node B may be failed. If A would stop the execution and B has not received the message, the execution of the algorithm would break since A would remain blocked, believing that B executes the algorithm further.

Upon reception of the request for control flow transfer, node B is ready to start the execution but it must notify node A that it is ready to execute the algorithm. It is necessary to send this acknowledgement to A because if A is unaware that B has received the message, it may assume failure of B and transfer control to another neighbor, creating two flows instead of one. After node B has sent its acknowledgement, it is ready to execute algorithm, but it must wait for confirmation from A, otherwise both of them may continue executing the algorithm. If A has received the notification from B, it has to send notification on notification, and it expects confirmation from B on reception of this second notification.

Same as in two-general problem [27], A and B are involved in infinite message ex-

	dDFS	dBFS (+ DIBADAWN)
messages sent	$2e$ + proactive 1-hop neighborhood	$2n-1$
message overhead	$2e$ + proactive state + no use of search data	$2n-1$ or $n-1$ if assisted by the network layer
detection in presence of message losses	causes errors	causes errors
robustness	very low	high

Table 5.1.: Comparison of dDFS and dBFS for bridge and articulation point detection in WMNs.

change, otherwise they risk that either the search stops, or that two flows are executed instead of one. If link between nodes is of sufficient quality, probability of erroneous execution termination or undesired parallelization of execution is small, but the correctness guarantees cannot be established within a finite messaging sequence.

dBFS is implemented with the help of MAC broadcasts and its control flow issues in the forward phase are less pronounced. It creates parallel execution flows by its definition so this error source does not exist. Same as dDFS it may stop the execution prematurely because of message losses. Probably the biggest advantage of dBFS over dDFS in WMNs is that dBFS does not mix backtracking phase with the forward (exploration) phase of the algorithm. It is either in forward phase, when it explores the network, or backward when it routes the collected information on network toward the root of the search tree. This enables construction of simple and efficient timeout schemes at nodes.

Additional advantage of dBFS over dDFS for tree construction in WMNs is the usability of data directly obtained by each of the search strategies. In an operational and used network, dBFS is executed regularly by network layer protocol: e.g., for route discovery in reactive protocols [54] [137], localization service in geo-routing protocols [48], or dissemination of interests of sinks to sensor nodes in wireless sensor networks [91]. If dBFS is used as basis of bridge and articulation point detection, the communication overhead is reduced since network is already performing dBFS for its own purposes and the biconnectivity testing algorithm can reuse data obtained from the search.

Contrary to dBFS, applications of dDFS in WMNs are rare and not accepted in practice. As a rare example, Stojmenovic et al. [154] have performed a pure topological study of DFS applicability in reactive routing protocols – they have compared path lengths produced by dDFS and dBFS. It seems that authors of [154] are unaware of control-flow issues caused by packet losses and they did not use simulations nor experiments to demonstrate that it is possible to operate distributed DFS on a communication channel with packet losses.

Table 5.1 summarizes this discussion. dDFS imposes higher overhead for its execution than dBFS since it requires neighbor detection and discovers one edge per transmission. Since it traverses all tree edges two times, its message complexity is $2e$. Message complexity of dBFS is $2n-1$ since every node broadcasts exactly one message in the forward phase while in backward phase all nodes except the root send one message to their parents. dBFS has even greater advantage (lower overhead) if we consider that it

is frequently executed by other protocols, and biconnectivity testing algorithm can use this data. Both in dDFS and dBFS message losses may result in incorrect decisions so neither has the advantage. Depending on topology of the network and location of the loss, effects on detection accuracy can be more or less severe. Robustness of dDFS is very low. In addition to data flow, it has control flow and termination issues. Because of the weak coupling between execution phases of algorithm, dBFS terminates even in presence of packet losses. It can be concluded that a dBFS-based biconnectivity testing algorithm outperforms the dDFS in almost every category in the wireless environment.

5.3. Adaptation of the Echo Algorithms for Application in WMNs

The echo algorithms [55][67] are a class of distributed network algorithms that fulfills the requirements that are placed upon the biconnectivity testing algorithm which is to be executed in a WMN (they are listed in the previous section). Most importantly:

- They have separated forward (exploration) and backward phase (processing of data from the forward phase) so they do not face the backtracking problems like DFS.
- They operate on various tree types so the dBFS can be used in their forward phase. Instead of sending one-by-one explorer as in [55][67], a MAC broadcast sends a set of explorers saving node energy and reducing channel contention.

Echo algorithm has already been adopted for biconnectivity testing in multiprocessor networks [55]. However, the direct application of it in WMNs is not possible and several crucial changes had to be made:

- The algorithm which is described in [55] relies on direct reply (echo) after detection of a cross-edge. A node that sends an explorer over a cross edge immediately obtains an echo which indicates that the discovered edge is a cross-edge. This information is used for construction of sets which contain INT(ernal) and TERM(inal) nodes[5]. In WMNs such implementation is inappropriate since it creates peaks in contention intensity. This issue does not exist in wired networks where links are mutually independent, but communication medium in WMNs is shared by a group of nodes.

  For instance, let us assume that node A in Figure 5.11 (page 96) starts the search, and that node C is the second node which explores the graph further. As it sends its message to neighbors, it discovers three cross edges CE, CD, CF. By [55] every node at a cross edge must respond with an echo to notify node C of cross edge existence. They must not do it immediately, otherwise echoes will collide at C and will not be received. Furthermore, if node E sends an explorer immediately after node C, it is received by nodes B,C,D,F. They are all visited so they are supposed to send echoes to node E, further increasing the probability of packet collision in vicinity of node C.

[5] INT and TERM are the names of sets used in [55].

A node X of degree $d(X)$ can trigger up to $d(X) - 1$ echoes from its neighbors. In real networks (as it will be demonstrated in the case study in Chapter 7), nodes have degree of up to 20 and some network sections are much denser than in the small example from Figure 5.11. The discussed issues in the cross-edge discovery would escalate in such dense network sections. The issues caused by echo messages are similar to the so-called "ACK implosion" problem [90][133] that occurs if acknowledgements are used for reliability improvement of network-wide multicast and broadcast protocols.

In addition to the issue of increased packet collision probability created by these echo message storms, the message complexity of such an approach is high. For a network consisting of n nodes and e edges, total number of transmission by this scheme is n (each node in the network performs one MAC broadcast to discover the tree and/or cross edges) plus $e - n + 1$ echoes for the cross edges (the number of network edges reduced by $n - 1$ edges which belong to the tree), yielding $e + 1$ messages in the forward phase of the algorithm.

In this work is introduced cross-edge detection based only on MAC broadcasts, without echoes at cross-edges. The detection of cross-edges is performed independently at nodes incident to them which enabled reduction of the number of sent messages in the forward phase of the algorithm to the minimum: n packet transmissions.

- Construction of the INT and TER sets as it is demanded in [55] requires the total ordering of node visits. This can be achieved by ordered node visits, utilizing a neighbor discovery protocol and MAC unicasts. Such approach in undesirable (as explained in Section 5.2) and deprives the detection algorithm of advantages brought by the MAC broadcasts in the forward phase. It is possible to implement the total ordering with MAC broadcasts if nodes have globally synchronized clocks. However, the uncertainty of wireless communication confronts us again: results of [74] and [81] show that it is not possible to perform global synchronization of drifting clocks in WMNs. It is possible to reduce the difference between clocks to order of hundred of microseconds [124] but the clock precision required for total ordering in this application scenario is in range of nanoseconds.

- In order to resolve termination issues in presence of packet losses, the implicit detection of tree-leaves had to be eliminated and substituted by a timeout mechanism. If network offers reliable message delivery, it is possible to execute the algorithm until tree-leaves have been reached and then to roll the execution of the algorithm backwards: a node executes its backward phase only after all its children have executed it and responded to the node. However, in presence of message losses this approach is ineffective and incorrect. More details on its issues can be found in Section 5.4.1.

The independent cross-edge detection and association of a separate marking to each cross-edges not only resolved some of issues of [55] in WMNs, but it also enabled introduction of important classes of voting rules. As it will be shown in the evaluation, the direct decisions of DIBADAWN achieve limited accuracy in WMNs, but with the extensions provided by the decision rules, its accuracy is considerably improved.

Field Name	Data Type	Forward Phase	Backward Phase
searchId	unique ID	√	√
msg. type	boolean	√	√
TTL	integer	√	√
treeParrent	node ID	√	×
forwardedBy	node ID	√	√
payload	variable	×	BRIDGE or NOBRIDGE unique identification

Table 5.2.: DIBADAWN message format.

5.4. Distributed Biconnectivity Testing in WMNs

In this section are presented changes introduced to Echo and DFS algorithms that are necessary for the distributed bridge and articulation point detection in WMNs, in accordance with the previous discussion. The derived algorithm accounts for issues that are present in a distributed, asynchronous systems with unreliable communication channels such are wireless multi-hop networks. Since the changes considerably alter the functionality of the detection algorithms that inspire it [55][158], the proof of correctness of this new version of the algorithm is presented in Appendix B (for networks with reliable message delivery)[6].

The bridge and articulation point detection is preformed through message exchange. The messages are used both for information exchange and control of execution flow. Their format is shown in Table 5.2. Most of message fields are used in both execution phases with few exceptions (e.g., *treeParrent* is used only for forward phase).

More than one DIBADAWN instance may be executing simultaneously in the network so it is necessary to distinguish between search instances (field *searchId*). For each of searches, its data structures are managed independently. The identifier of a search is unique. For instance, it may be composed as concatenation of initiator node's MAC address (unique) and hash value of the search start time. The *message-type* field is used to determine whether a message belongs to forward or backward phase of algorithm, so that it can be correctly decoded by a node that receives it.

The searches are initiated in method **start_search** (Figure 5.2). Time to live field of the first message is set to a value $maxTTL$. The value should not be considerably larger than network's diameter but it must not be smaller than it. Since this is the first message to be sent within the search, its tree parent field is set to null. The $forwardedBy$ field is set to a node that sends the packet. The distributed execution of the algorithm starts after the message is MAC broadcasted.

Upon reception of a forward search message by a node, it is handled in the method **receive_forward_message**. If a node has not been visited by this search instance and if the TTL (time-to-live) field of the message has not reached zero, the message should be re-broadcasted (construction of the BFS tree). The node updates $treeParent$, TTL, and $forwardedBy$ fields in the message. Before it performs MAC broadcast of the message

[6] A pseudo code of the simplified version of algorithm can be found in Appendix B. Algorithm functionalities are the same, but the version from the appendix may be easier to understand since it does not include implementation details.

```
list of variables shared across searches:
    list edgeMarkings;
    list APdecisions;
    //these lists contain tuples (time, search ID, decision, edge, competence)
each search has following data structures:
    boolean visited, isArticulationPoint;
    list crossEdges;
    node parent; integer myTTL
    backward message buffer messageBuffer;
    list of message-sets for each neighbor msg_i
    //(created dynamically upon neighbor first discovery and set to BRIDGE_ni);
start_search():
    create empty message;
    msg.initiator=this; parent=null;
    msg.id=this::hash(time)
    msg.ttl=maxTTL; msg.forwardedBy=this;
    visited=true;     MAC broadcast the message msg;
    start timeout
receive_forward_message(message msg):
    if(not visited) {
      visited=true;parent=msg.forwardedBy;
      start timeout;
      myTTL=ttl;
      if(myTTL>0) {
        msg.ttl=ttl-1; msg.forwardedBy=this;
        msg.parent=parent;
        MAC broadcast the message after U(0,jitterMax) seconds;
      }
    } else if(msg.parent!=this) // ignore edges belonging to tree
      crossEdges.add((msg.forwardedBy));
on timeout:
    detect_cycles;
    forwardMsg;
    execute APdetection(), add decision to list APdecisions
    execute voting rules for bridge and articulation point detection
```

Figure 5.2.: Implementation of the DIBADAWN (1).

it waits for random time, drawn from U(0, jitterMax). Jitter reduces probability of simultaneous MAC broadcast of several neighbor nodes that causes packet collisions and losses.

If a node has already participated in the search instance, incoming message may indicate a discovery of a cross-edge. Detection of cross-edges is simple: they belong to connectivity graph but do not belong to the tree. Tree messages are easily identified thanks to the *treeParent* field:

- The first forward message received by a node belongs to the tree and leads to its parent.
- If a node receives a forward message and the field treeParent in the message is equal to its identification, the message originates from its child and it is ignored.
- If a node receives subsequent forward messages (after the first message) and its identifier is not in the *treeParent* field, it has discovered a cross-edge.

The field *treeParent* is added to the forward message in order to avoid queries to neighbor nodes on their parent status that are necessary for the cross-edge detection. This field is not necessary in a wired network, since a node is able to control over which

```
detect_cycles():
    for each m in crossEdges {
      if(this<m.forwardedBy) cycleId=m.id::this::m.forwardedBy;
      else cycleId=m.id::m.forwardedBy::this;
      add (time, m.id, NOBRIDGE, (this,m.forwardedBy), 1) to edgeMarkings;
      msg_{m.forwardedBy} = msg_{m.forwardedBy} ∪ {NOBRIDGE_{cycleId}}
      bufferBackMsg(parent, cycleId);
    }
}
receive_backmessage(backward message m):
    if(m.payload=BRIDGE)
      add (time, m.id, BRIDGE, (this,m.forwardedBy), BR − competence) to edgeMarkings;
      msg_{m.forwardedBy} = msg_{m.forwardedBy} ∪ {BRIDGE_{m.payload}}
    else if messageBuffer.contains(m.payload) {
      removedMsg=messageBuffer.remove(m.payload);
      add (time, m.id, NOBRIDGE, (this,m.forwardedBy), 1) to edgeMarkings;
      add (time, removedMsg.id, NOBRIDGE, (this,removedMsg.forwardedBy), 1)
to edgeMarkings;
    } else messageBuffer.enqueue(m);
}
bufferBackMsg (node destination, identifier cycleId):
    create new backward packet m;
    m.forwardedBy=this; m.ttl=maxTTL;
    m.payload=NOBRIDGE; m.id=cycleId;
    messageBuffer.add(m);
forward_msg ():
    if(messageBuffer = ∅) {
      add (time, msg.id, BRIDGE, (this,parent), BR − competence) to edgeMarkings
      send BRIDGE message to parent;
    } else {
      for each message m in messageBuffer
        update message fields: msg.ttl–, forwardedBy=this, etc.
        if(m is in possibleTrustedVotes)
          add (time, msg.id, NOBRIDGE, (this,parent), 1) to edgeMarkings
        else {
          add (time, msg.id, NOBRIDGE, (this,parent), msg.ttl) to edgeMarkings
          add (time, msg.id, NOBRIDGE, (this,msg.forwardedBy), msg.ttl) to edgeMarkings
          msg_{m.forwardedBy} = msg_{m.forwardedBy} ∪ {NOBRIDGE_{m.cycleId}}
        }
        msg_{parent} = msg_{parent} ∪ {NOBRIDGE_{m.cycleId}}
    }
    group messages from messageBuffer in packets and send them to parent
APdetection() {
    create zero filled matrix closure[|adj()|][|adj()|]
    for all pairs (i,j) from adj() × adj()
      if(msg_i ∩ msg_j ≠ ∅) closure[i][j] = 1;
    for all pairs (i,j) from adj() × adj()
      if(closure[i][j] == 1)
        for k in adj()
          if(closure[j][k] == 1) closure[i][k] = 1;
    if (closure is not 1-matrix) isArticulationPoint=true;
```

Figure 5.3.: Implementation of the DIBADAWN (2).

link the outgoing message is sent. In a WMN, a node uses the MAC broadcast, the message reaches all neighbors, including its tree-parent. Without this field, parent node would erroneously conclude that the link to its child in the tree is actually a cross-edge (parent as an already visited node is visited again). Other option for parent would be to initiate additional message exchange in order to determine whether a message it just received came from its child or over a cross-edge but it would increase communication overhead. These changes enable the exclusive use of MAC broadcasts in the forward

phase, so instead of individual neighbor visits and direct echo replies that are required in graph version of echo algorithm, cross-edges are detected independently and without explicit notifications (echoes).

Nodes are using timeout mechanism to ensure the proper execution order in the backward phase: after reception of the first message in the forward phase, a node starts a timeout that will trigger the backward phase. The timeout at a node on i^{th} tree level is reduced proportionally to the reduction of TTL of forward packets:

$$timeout_i = TTL_i * maxTraversalTime \tag{5.1}$$

This ensures that nodes closer to the root of the tree enter the backward phase later than the nodes higher in the tree, and the correct execution order of backward phase. The parameter $maxTraversalTime$ should be sufficiently large to include the packet transfer time, jitter, queuing and processing delays. Some routing protocols already include similar parameters: for instance, the maximum traversal time per node can be taken from NODE_TRAVERSAL_TIME of AODV (default value in [137] is 40ms).

The backward phase is activated by expiration of the timeout. The mutual functionality of methods **detect_cycles**, **forward_msg**, and **receive_backmessage** is detection of HCA cycles. Once the cross edge is discovered, it is forwarded towards the HCA where it gets paired, "closing" its cycle. The algorithm does not discover all cycles in a network but only the HCA cycles. This characteristics is actually beneficial for its performance, since redundant cycle detection causes unnecessary message exchange between nodes, increasing the contention on the communication channel. The proof that it is sufficient to discover only HCA cycles in order to mark all edges in a network as NOBRIDGE can be found in Section B.1 at page 220.

As the first step of the backward phase, a node executes the method **detect_cycles**. Each cross-edge identifies a cycle. The identifier of a cycle is created by concatenation of the pair of identifiers of nodes incident to the cross-edge (node with the higher identifier is always placed first, as a rule of creation of identical identifier at both nodes incident to a cross-edge). Since the unique *searchId* field is included in the backward packet, all cross-edge markings are also unique.

The markings are not sent instantaneously down the tree but they are buffered at a node in order to reduce traffic overhead: the amount of useful data that is sent remains the same but if more markings are placed in one MAC frame, the total overhead on the communication channel (e.g., MAC header, interframe spaces) is reduced.

Markings of an edge are added to the list *edgeMarkings* in order to support the voting rules. Each marking in the list is represented as a tuple that consists of the local time of marking, search identifier, edge identifier, marking and its competence. The competence of a marking is probability that it is correct, as it will be defined in Section 5.6.2, where it is shown that some markings have higher competence than others and that marking differentiation may be beneficial for detection accuracy.

A non-leaf node receives backward messages through the method **receive_backmessage**. The BRIDGE markings are directly stored in the *edgeMarkings* list. NOBRIDGE messages are queued in *backQueue* if they are encountered for the first time. If pairing occurs, both paired messages are deleted and markings for both of them are added to the *edgeMarkings* list. The pairing guarantees existence of a cycle so competence of such markings is one. The pairing is key part of backward

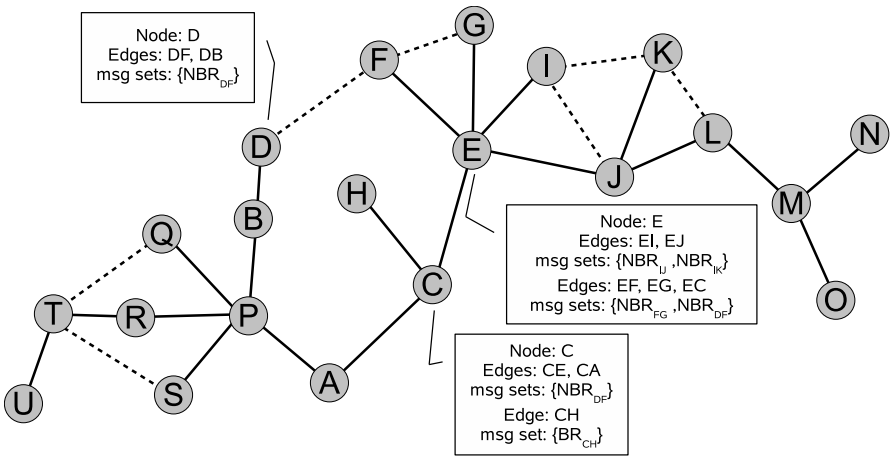


Figure 5.4.: Message sets used for articulation point detection (after transitive closure).

phase, since it prevents cycle-marking messages to escape from their cycles.

If there are no messages to be forwarded to the tree parent, the link to the parent obtains BRIDGE marking and the parent in the tree is notified of it by an appropriate BRIDGE message.

Buffered NOBRIDGE messages, if they exist, are sent at the end of the method **forward_messages**. An appropriate tuple is added to the list $edgeMarkings$ for each forwarded message. Finally, voting decision rules are applied on collected markings to decide which links are bridges.

After it has forwarded markings from the backward phase, node processes all messages it has received in this execution instance of the algorithm in order to calculate whether it is an articulation point. Bridge detection and articulation point detection are related, but a simple reuse of bridge detection (assuming that all vertices incident on a bridge are articulation points) is incorrect since:

- It does not include all articulation points: an articulation point may belong to multiple cycles (nodes E and P in Figure 5.4)
- It may include vertices that are not articulation points: edge incident to a pendant node is a bridge but a pendant node is not an articulation point (nodes H, N, O in Figure 5.4).

Therefore, additional functionality is required in the algorithm to detect them.

Definition 5.1 *Let E' be the set of edges of graph $G(V, E)$ that are incident to a node P. Edges PQ and PR from E' are in relation $\circledcirc$ if there exists a common cycle in $G(V, E)$ to which they both belong.*

The relation $\circledcirc$ is an equivalence relation and it divides edges of G in equivalence classes that correspond to bicomponents [31] (Lemma B.4, page 223). As it is common in literature [158], the relation is used for detection of articulation points. The proof that the implementation proposed in Figures 5.2 and 5.3 is correct and that algorithm delivers sufficient amount of data to nodes in order to correctly identify articulation points can be found in Section B.2 (page 223).

Within every algorithm execution, every node P has a set msg_{PQ} attached to each edge PQ discovered in that execution of the algorithm. Initially the set consists of markings msg'_{PQ} that the algorithm has sent over the edge PQ. The algorithm associates the markings also with cross-edges for completeness and correctness reasons, although it actually does not send them. As a consequence, each edge PQ incident to the node P has a non empty set msg'_{PQ}.

Note: Relation $\circledcirc$ from Definition 5.1 is defined for all edges in the graph. Since the DIBADAWN is distributed, it is not even aware of all edges in the graph. So, a node can calculate it only for its incident edges.

Definition 5.2 ***(Implementation of the relation $\circledcirc$ in DIBADAWN)*** *The relation $P \circledcirc Q$ holds in DIBADAWN if $msg_P \cap msg_Q \neq \oslash$.*

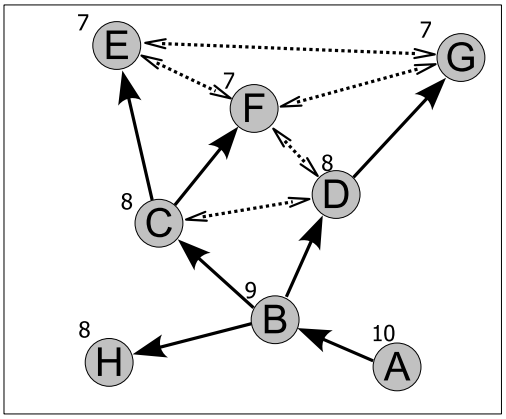
If message sets remain as they are initialized (containing only messages that are sent over them) the relation is incomplete. For instance, in Figure 5.4 the message sets are $msg'_{PQ} = TQ$ (node Q delivered marking that reports existence of cross-edge TQ) and $msg'_{PS} = TS$ (node S reports existence of cross-edge TS) so based on them edges PQ and PS are not in relation although there exists cycle PQTS. In order to correctly calculate the relation, transitive closure of the relation $\circledcirc$ is implemented in the algorithm and applied to the initial message sets msg'. Only after the closure, the implementation of the relation (Definition 5.2) can be used for articulation point detection.

The transitivity closure places edges incident to a node into same equivalence class if and only if there exists a common cycle to which they both belong, or equivalently if they have a common NOBRIDGE message. So, in the previous example, $msg'_{PQ} = TQ$, $msg'_{PS} = TS$ and $msg'_{RP} = TS, TQ$ (node R forwarded markings originating from T that report existence of cross-edges TQ and TS). After transitive closure, they are all in relation $\circledcirc$ since $(PQ \circledcirc RP) \& (PS \circledcirc RP) => (PQ \circledcirc PS)$. In method **APdetection**, the well known and theoretically proven Warshall's algorithm [170] for transitive closure of relation $\circledcirc$ is used. If its output is 1-matrix, node is not an articulation point (all edges belong to a single equivalence class).

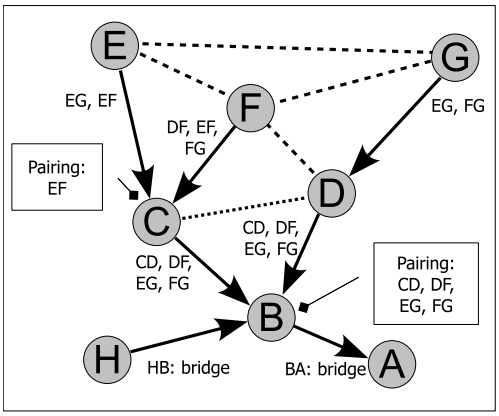
In Figure 5.4 can be seen that node D is not an articulation point since both of its edges belong to the same class. Node C has an incident bridge that creates two equivalence classes, so it is articulation point. Node E is an articulation point although it does not have incident bridges – its edges are divided into two equivalence classes of relation $\circledcirc$: one equivalence class contains edges GE, FE and CE while the other contains edges IE and JE.

The Lemma B.4 [31][158] shows that the equivalence class calculation correctly detects the articulation points. However, DIBADAWN does not operate on all cycles, but only on the HCA cycles. Additionally, this data is propagated through the network and some of it is deleted through pairing of markings. The Theorem B.2 (page 223) proves that DIBADAWN has sufficient data for correct calculation of equivalence classes.

Note: The articulation point detection in method **APdetection** may seem to be a local decision of a node since it is made based on the markings of edges incident to it. However, the data may originate from distant nodes in the network and captures the general graph structure.



(a) dBFS tree with TTL values. Cross-edges are marked with dotted lines.



(b) Backward phase of the algorithm and pairing of messages.

Figure 5.5.: Example of bridge detection in DIBADAWN execution on a BFS tree.

Event	Methods
1	A.start_search()->A.MACbroadcast(), B.receive_forward_message();
2	B.MACbroadcast(), {A, C, D, H}.receive_forward_message()
3	H.MACbroadcast(), B.receive_forward_message()
4	C.MACbroadcast(), {B, D, E, F}.receive_forward_message(), DC is cross-edge
5	D.MACbroadcast(), {B, C, F, G}.receive_forward_message(), CD and FD are cross-edge
6	E.MACbroadcast(), {C, F, G}.receive_forward_message(), FE and GE are cross-edges
7	G.MACbroadcast(), {E, F, D}.receive_forward_message(), EG and FG are cross-edges
8	F.MACbroadcast(), {C, D, E, G}.receive_forward_message(), DF, EF, GF are cross-edges
9	E.timeout(), C.receive_backmessage({NBR-EG, NBR-EF})
10	G.timeout(), D.receive_backmessage({NBR-EG, NBR-FG})
11	F.timeout(), C.receive_backmessage({NBR-DF, NBR-EF, NBR-FG}) C pairs NBR-EF
12	H.timeout(), B.receive_backmessage({BRIDGE-BH})
13	D.timeout(), B.receive_backmessage({NBR-CD, NBR-EG, NBR-FG, NBR-DF})
14	C.timeout(), B.receive_backmessage({NBR-CD, NBR-EG, NBR-FG, NBR-DF}) B pairs NBR-CD, NBR-EG, NBR-FG, NBR-DF
15	B.timeout(), A.receive_backmessage({BRIDGE-AB})
16	A.timeout()

Table 5.3.: Sequence of method invocations for the example in Figure 5.5.

Example of DIBADAWN Execution

Figure 5.5 and Table 5.3 provide an example of execution of the detection algorithm, if it is coupled with dBFS as the tree construction algorithm. In the first phase of the algorithm (Figure 5.5(a)), tree is constructed. The number next to a node represents TTL of sent packets.

The timeouts expire in reverse order to the order of visits in the forward phase. Timeout order in the second phase of algorithm (Figure 5.5(b)), is shown by events nine to sixteen in Table 5.3. Each node marks all cross-edges that it observed and sends corresponding markings to its parent. A single packet to the parent in a tree may carry more than one NOBRIDGE marking: e.g., three NOBRIDGE messages are transported over link CF (11^{th} event in Table 5.3).

Beside the cross-edge detection, the nodes perform pairing and forwarding of unpaired

messages. For instance, node C first receives messages from nodes E (event 9) and F (event 11), pairs two EF markings (event 11), then it detects cross-edge CD and forwards remaining three messages to its parent B (event 14).

If no messages are to be forwarded, either because node had no children nor incident cross-edges (node H) or because all messages have been paired (node B), node informs its parent of bridge existence. Once the root finishes execution of its backward phase, the execution of this algorithm instance is terminated.

When to execute algorithm

There exist two opposing constraints for timing of DIBADAWN executions. In order to capture possible topology changes in the network (e.g., a node has failed), DIBADAWN has to be occasionally executed.

The algorithm executions must not be too frequent, otherwise DIBADAWN overhead may approach the overhead of proactive topology management protocols. The following, non-obligatory recommendations should be used to determine when to execute the DIBADAWN:

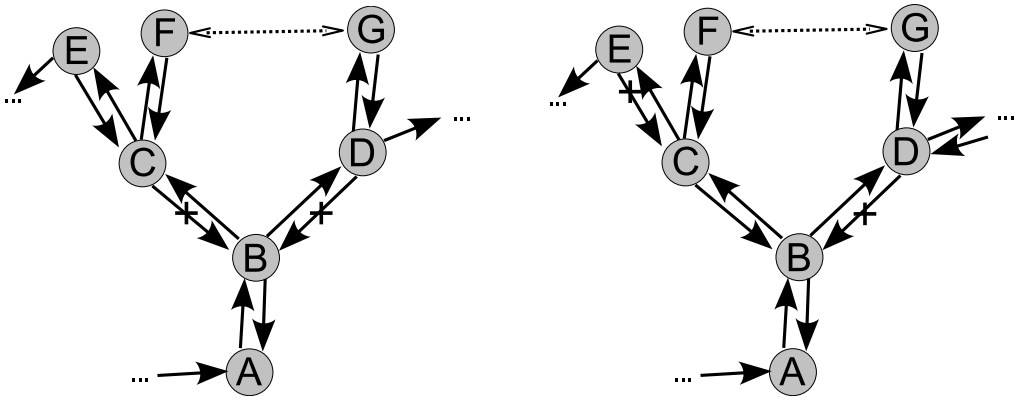
- Concurrently with execution of dBFS by other network protocols. The overhead of forward phase does not exist since data from routing protocol is reused.
- Each time a new node is added to network or when existing node reboots. Every new node changes the topology and requires update of bridge and articulation point decisions. A node is aware of its status change and executes DIBADAWN.
- If a node does not observe any DIBADAWN-related activity in the network for time $t_{inactive}$, it executes DIBADAWN with a certain probability (to avoid simultaneous execution by all nodes in a network).

5.4.1. Execution Issues of Echo Algorithms in WMNs

Beside the global timeout scheme described in Equation 5.1 a version of the algorithm that operates as in [55][67] was tested in [120] (with regard to algorithm rules that regulate the execution order in the backward phase). The idea is to detect the leaves in the tree and to start the backward phase from leaves. Since a leaf has no children, it receives only one message in the forward phase from its tree parent. If a node does not receive a response from its possible children within $jitterMax$ seconds, it is a leaf in the tree. A non-leaf node is supposed to track existence of its children in the tree. Once all of its children responded, it can process back-messages and perform necessary communication with its parent in the tree.

The issues of such execution-flow control in WMNs are:

- A node may erroneously conclude that it is a leaf (e.g., node B in Figure 5.6(a)). DIBADAWN continues execution at its child(ren), but due to message losses, the node is unaware of its children, concludes it is a leaf and immediately starts the backward phase.
- A node may correctly conclude that it is not a leaf but it can be still unaware of all its children if some of messages from its children are lost (node C is unaware



(a) Erroneous detection of a leaf in search tree.

(b) Premature execution of backward phase at a non-leaf node.

Figure 5.6.: Issues of echo algorithms introduced by channel fading and message losses.

of its child E in Figure 5.6(b)). The node may start the backward phase earlier than it should. As the consequence, pairing of markings may be erroneous. Backtracking and correction of errors in pairing is possible but considerably increases communication overhead and unnecessarily complicates the algorithm.

- Propagation of errors to lower tree levels: premature execution of the backward phase at a node may trigger the same premature reaction at its parent. This may trigger a chain reaction of premature backward phase executions in the network. For instance, node B in Figure 5.6(b) is unaware of its child D and executes its backward phase as soon as node C responds even if D has not responded yet. Since B is the only child of A, node A will also execute its backward phase prematurely.

- Gain in execution speed is seldom noticed: it is sufficient that a node does not receive the backward-phase message from just one of its children which it has detected in forward phase (e.g., node D does not get backward phase message from node G in Figure 5.6(b)). By the original idea the node remains blocked in this state indefinitely. An additional timeout has been introduced in [120] to resolve this termination issue (similar to the timeout defined in Equation 5.1) and once it expires, the node executes its backward phase, assuming that its non-responding child has failed. This second timeout delay is propagated all the way to the root since node's parent cannot execute its backward phase before it gets answer from this, blocked node.

This approach has been implemented and evaluated in [120]. All of the listed issues have been observed in simulation. Particularly often has been observed that a node completes the backward phase, sends results to its parent and then receives messages from its (up to that point in time unknown) children. Some of these new backward phase messages may have been paired with other messages that are already passed down the tree. Such "late" messages were ignored causing significant information loss without any gain in contention level reduction: already delivered messages are not used by the recipient.

The implementation of bridge detection in [120] was unclear and complicated. The accuracy of detection results was worse than in implementation with the timeout from Equation 5.1 (see Figure 5.10, page 95). The timeout scheme from Equation 5.1 simplifies implementation, increases the information usage and reduces number of erroneous decisions.

5.4.2. Analysis of the Communication Overhead

In this section, the communication overhead produced by proactive topology management is compared with the overhead produced by DIBADAWN and the Echo algorithms.

The proactive topology management protocols which are used in WMNs need to be executed periodically in order to be able to react to changes in the network topology. Since DIBADAWN also requires this periodicity for the same reasons, its message complexity should be multiplied by frequency of its execution in network f_D in order to get the communication overhead in [packets/s]. For a single execution round, the communication overhead is equal to the message complexity (the number of messages sent by a single execution of an algorithm [38][46]).

Overhead of Proactive Topology Management

A proactive topology management protocol has two components of communication overhead. The first component is caused by detection of communication links. The heartbeats are the commonest approach to link detection (Section 3.3, page 37). Each node in a network sends a heartbeat with a frequency f_H, producing overhead of $n \cdot f_H$ packets per second.

The second component is caused by the dissemination of the local link information through the network. Assuming periodic dissemination of local link information, each node in the network initiates it with frequency f_B. If flooding is used, total number of generated packets is $f_B \cdot n^2$ (each of $f_B \cdot n$ broadcasts is repeated by n nodes in the network).

Due to packet losses, only a portion of the whole network may be covered by a broadcast, and the constant $c_{coverage}$ captures this reduction (depending on quality of links and network topology $0 \leq c_{coverage} \leq 1$). The coverage of individual disseminations is time and topology dependant and the parameter $c_{coverage}$ represents the average coverage of the individual disseminations.

The sum of these two components is the overhead of a proactive topology management protocol with flooding as the dissemination method:

$$OH_{proactive-flooding}[packets/s] = n \cdot f_H + n^2 \cdot f_B \cdot c_{coverage} \tag{5.2}$$

Definition 5.3 *A dominating set of a graph $G = (V, E)$ is a subset V' of V such that every vertex in G is either in V' or it is adjacent to a member of V'.*

Definition 5.4 *A connected dominating set of a graph $G(V, E)$ is a set of vertices $V' \subset V$ such that V' is a dominating set of G, and the subgraph induced by V' is connected.*

If the minimal connected dominating node set of a network is known, it can be used for reduction of data dissemination overhead. It is no longer necessary that all nodes forward messages of the local topology information as in flooding. Instead, only the initiator of the broadcast and the nodes from the dominating set are required to broadcast messages in order to cover the whole network (assuming no messages are lost). This reduces the number of sent messages by a factor c_{DS}.

$$OH_{proactive-DS} = n \cdot f_H + n^2 \cdot c_{DS} \cdot f_B \cdot c_{coverage-DS} \tag{5.3}$$

Identification of the minimal connected dominating set is a NP problem [76] and various heuristics have been proposed for its calculation. The overhead reduction c_{DS} depends on the graph topology[7], quality of heuristic that is used for creation and maintenance of the connected dominating set and overhead induced by the heuristics. For instance, the multipoint relaying (used in OLSR), reduces number of retransmitted packets to approximately one third.

This reduction of overhead does not come for free. The robustness of dissemination protocol is reduced. In presence of packet losses it has considerably lower coverage than the flooding (Figure 8 in [141]), reducing its applicability in WMNs where packet losses are common. Furthermore, the evaluation in [141] shows that overhead in WMNs is reduced by a constant factor (probably caused by WMN topologies that are inherently spatial and spread, without connectivity hubs as in, for instance, random graphs with power-law node distribution), so the complexity of data dissemination remains $O(n^2)$.

DIBADAWN Overhead

Communication overhead of DIBADAWN has two components. The first component is caused by tree construction and cross-edge detection. If no packets are lost, all n nodes participate in the forward phase. Same as in the dissemination of local link information in proactive topology management, the coverage of a search may not be ideal, which is captured by the factor $c_{coverage}$, yielding overhead of $n \cdot c_{coverage}$ packets.

In the backward phase of DIBADAWN, all nodes which participated in the forward phase are participating in forwarding of backward messages. Instead of MAC broadcast, which is used in the first phase, MAC unicast is used for forwarding of BRIDGE and NOBRIDGE markings. On the average, each node has to send a packet snd times during its MAC unicast. Thus, the DIBADAWN communication overhead is:

$$OH_{DIBADAWN} = c_{coverage} \cdot n + c_{coverage} \cdot (n-1) \cdot snd \tag{5.4}$$

If MAC broadcasts are used in the forward phase but the direct echo replies (as in [55][67]) are used for tree and cross-edge discovery, the message complexity of the forward phase is $e + 1$ (page 75). In the backward phase, each of tree edges sends a message using the MAC unicast. Thus, the message complexity is:

$$OH_{direct-echo} = c_{coverage}(e + 1 + (n-1) \cdot snd) \tag{5.5}$$

[7] In a chain of n nodes, the minimal connected dominating set has $n-2$ nodes and provides insignificant improvement. In a complete graph, the size of the minimal dominating set is one, providing huge overhead reduction.

In an ideal network (without message losses) there are no retries, so the parameter *snd* in Equations 5.4 and 5.5 is one, as well as the $c_{coverage}$. The message complexity of DIBADAWN is then $2n - 1$. The detection which utilizes MAC broadcast for edge discovery and direct echo messages for cross-edge detection has message complexity of $e + n$. If edges would be individually visited as in [55][67] the biconnectivity testing complexity would be $2e$ plus additional overhead needed for edge discovery (heartbeat exchange), yielding total overhead of $2e + f_H \cdot n$.

Summary

It can be seen from Equation 5.2 that the proactive topology management protocols have message complexity of $O(n^2)$ even if they use minimal connected dominating sets for topology dissemination. The original echo algorithms have message complexity of $O(e)$. DIBADAWN uses the independent cross-edge detection and benefits provided by the wireless communication medium (MAC broadcast) which enabled us to reduce its message complexity to $O(n)$.

The advantage of proactive topology management is that once the topology is discovered and disseminated, nodes may execute various topological algorithms at the obtained topology without additional communication overhead. However, since WMNs are dynamic and unpredictably changing, the dissemination has to be executed periodically in order to be able to track the changes in network topology. Of course, DIBADAWN also has to be executed periodically with frequency f_D in order to detect the changes in the topology. The absolute number of messages sent by the periodic invocation of algorithms grows linearly with the frequency of their invocation so it cannot change the advantage of the smaller communication overhead produced by DIBADAWN.

5.5. Algorithm Behavior in Presence of Packet Losses and Node Failures

In an ideal communication network, without internal and external faults, the presented algorithm detects all bridges in a single pass. In reality, messages are lost and nodes fail, introducing errors in the detection algorithm. The issues created by the faults cannot be ignored, so this section analyses classes of faults present in the system and their effects on detection of bridges and articulation points.

Message losses are particularly frequent in WMNs and they are the main source of faults. Due to the message losses, network topology perceived by a node may be different from the actual topology, visible to the omniscient observer. For example, let us observe a simple network consisting of three nodes A, B, C and links AB, AC, BC (Figure 5.7(a)). The links have qualities $P_{AB}, P_{AC}, P_{BC} \in [t, 1]$ where t is the link acceptance threshold. Let us further assume that node A initiates the graph traversal using the flooding. There exist several important implications of the fact that the link qualities belong to interval [t,1] to the graph traversal, perceived node topology and biconnectivity testing.

In a network with reliable message delivery ($P_{ab} = P_{ac} = P_{bc} = 1$) tree edges to nodes B and C are always discovered (marked with continuous line in Figure 5.7(b)) as well as the cross edge BC (marked with a dotted line).

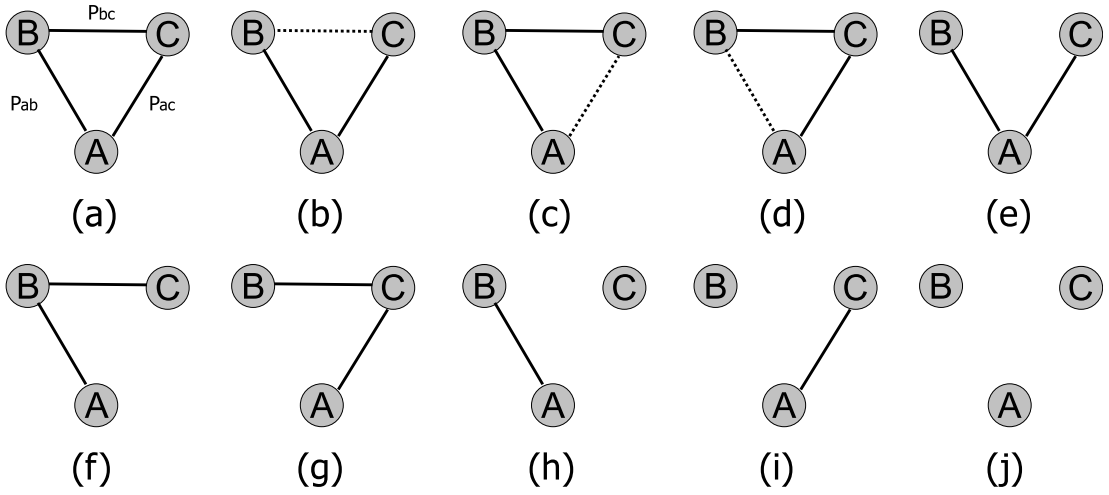


Figure 5.7.: Example outcomes of network traversal in presence of unreliable communication channel.

In presence of message losses, there are no longer guarantees on order of node visits (cases (c),(d)). For instance, if MAC broadcast initiated by node A is not received by node C, but node C receives broadcast from B, and node A receives broadcast from C, the resulting perceived topology is presented in case (c).

The completeness of traversal is also jeopardized as shown in cases (e) to (j). For instance, if neither node B nor C receive message from node A, the perceived topology is shown by case (j) and node A concludes that it is isolated. It can be seen that already in this simple network with only three nodes and three communication links, there exists nine different traversals if flooding is started from A. If the directed traversal is observed (e.g. node B received message from C, but C did not receive message from B), number of possible traversals is even larger.

It is particularly important for bridge and articulation point detection to notice that out of eight traversals, only three have detected the cycle ABC. In the remaining five cases, topologies perceived by node A and obtained trees would result in erroneous bridge and/or articulation point markings. For instance, in case (e), node A would be erroneously marked as articulation point and edges AB and AC as bridges.

The presented issues of network exploration can also occur if heartbeat link detectors are used. Since HLDs cannot guarantee correctness of their decisions in wireless environment, the perceived topology may deviate from the correct topology.

5.5.1. Analysis of Faults, Errors and Failures of the Detection Algorithm

In order to improve understanding of DIBADAWN's behavior in reality, in this section are discussed faults in the system, error states caused by them and detection failures. The methodology from [30] is applied. The methodology requires the definition of system, its function, and service provided to the end users.

The **system** consists of computing nodes in a wireless multi-hop network, and implementation of the DIBADAWN algorithm that is executed at every node. It is assumed

that there are no errors in implementation of DIBADAWN algorithm so nodes either operate it correctly or they are silent. This is a safe assumption since DIBADAWN implementation is fairly small. There are no malicious nodes that propagate random or deliberately incorrect information through the network.

The **function** of the system is to timely and distributively detect bridges and articulation points in a wireless multi-hop network. The behavior of the system is described by the algorithm and presented in Figures 5.2 and 5.3.

A **service** is correct if it implements the system function. Service can be observed in the system as a whole, and it operates correctly if all bridges and articulation points are detected. For a single **user** (node) service is functional if the node correctly detects the bridges on its incident links and whether it is an articulation point. In presence of faults, some users will receive correct service while some will not. Faults and errors induced by them are time-dependant – in one execution of DIBADAWN node may perform correct detection but in the next it may reach erroneous decisions because of packets losses in the network.

Service **failure** (or just failure), occurs when a service deviates from its specified functionality. The service failure modes distinguish different types of deviation from correct service execution. For biconnectivity testing, failure modes of the algorithm are the incorrect decisions. There exist four failure modes of DIBADAWN in the system:

1. A bridge is marked as an ordinary edge.
2. An ordinary edge is marked as a bridge.
3. An articulation point is not marked (it is marked as an ordinary node).
4. A ordinary node is marked as an articulation point.

The service of bridge and articulation point detection is described by a set of states defined in Section 5.4 – nodes keep track of tree structure, cross-edges, adjacent nodes and exchanged messages. In case of an error, one of the states deviates from its correct state. Whether an error produces a service failure depends on behavior of the system and type of error: some errors may be eliminated before they cause failures (e.g., in Mode 2 an edge is not marked as NOBRIDGE due to error in cross-edge detection, but if the edge belongs to a detected cycle, it is correctly marked and the first error is masked) or they do not influence system's capability to deliver service (e.g., Case 5 in Table 5.4). An error need not be restricted to a single node as it can be propagated between nodes in the network (e.g., Case 7 in Table 5.5).

The cause of an error is called a **fault**. Same as for errors that do not always cause failures, faults can be active (causing an error – for instance all faults in Table 5.5) or dormant (not causing an error – like Cases 12 and 14 in Table 5.6). A fault for the algorithm is not restricted to faults of node or its software. As it is shown later, a fault may be even the rejuvenation of previously failed node. Faults can be internal or external to the system. Node failures are internal faults for service of bridge and articulation point detection. The most frequent faults in WMNs are external communication faults.

The communication faults cause loss of packets in the network and errors in the algorithm. A packet can be lost because of a natural fault (physical signal propagation properties, increased white noise on receiver) or caused by node behavior (high contention on the wireless channel). For DIBADAWN, reasons of communication faults are

Case	Fault	Error	Failure	Severity (Bridge)	Severity (AP)
1	Message losses in the forward phase	Search may stop completely. It can be caused by a single packet loss (traversal of a bridge) or by simultaneous multiple packet losses in the network	Modes 1, 2, 3, 4	**Very High**: parts of the network are not searched by the algorithm. In the worst case, $n-1$ out of n nodes in the network could be affected.	
2	Double message loss in the forward phase	A cross-edge is not detected at either of its incident nodes.	Modes 2, 4	**Medium**: Cross-edge is not discovered thus it receives no marking. Some of tree edges belonging to the cycle may be marked as BRIDGE. Severity is limited since tree edges frequently belong to more than one cycle, allowing their correct marking.	**Medium**: Although a cycle is not detected, equivalence class may be calculated correctly (due to redundancy of cycles).
3	Message loss in the forward phase	Asymmetrical cross-edge detection (one node detects the cross-edge, the other node does not). Node that detected it is not aware of asymmetric detection.	Modes 1, 2, 3, 4	**High**: some edges that belong to a cycle may not be marked as NOBRIDGE. Additionally, NOBRIDGE message may go below HCA of the cycle, marking bridges as NOBRIDGE	**High**: Some nodes may conclude that they are articulation points although they are not (because of failure mode 2). If NOBRIDGE message is forwarded lower than its HCA, some nodes may falsely calculate equivalence classes and declare themselves as ordinary nodes although they are articulation points.
4	Message loss in the forward phase	Asymmetrical cross-edge detection. Node is aware of asymmetric detection.	Modes 2, 4	**Medium**: Failure modes are same as in Case 2. Since this event is detectable, corrective actions may be taken.	
5	Order of node visits in forward phase changes due to jitter	All edges in graph are covered but tree is no longer BFS.	none	**None**: Although this is a deviation from expected behavior of dBFS, it does not impact functionality of the detection algorithm – DIBADAWN can operate on any tree construction algorithm that traverses each edge at least once, which is satisfied in this case.	

Table 5.4.: Errors and failures introduced to the algorithm by communication faults (1).

not important. Two communication fault classes are distinguished: loss of message(s) in forward phase of the algorithm and loss of message(s) in backward phase of the algorithm. Each of faults may create different errors, depending on message type, its payload and the fault timing.

The losses in both phases of algorithm result in incorrect edge and node markings (Tables 5.4 and 5.5). Due to message buffering in backward phase of the algorithm and simultaneous transmission of them in a single packet, burst losses (multiple faults) may occur. They are treated as series of independent, successive faults, since they together

Case	Fault	Error	Failure	Severity (Bridge)	Severity (AP)
6	Message loss in the backward phase	Bridge message is lost.	Modes 1, 3	**Medium**: the edge obtains no marking which is better than obtaining false marking.	**Medium**: A single articulation point may be declared as ordinary node due to bridge marking loss.
7	Message loss in the backward phase	Out of two NO-BRIDGE messages, one is lost before HCA, other after HCA (or it is not lost and reaches root).	Modes 1, 2, 3, 4	**High**: same as in Case 4 both types of failures occur.	
8	Message loss in the backward phase	Out of two NO-BRIDGE messages, one is lost before HCA, other at HCA. Although the message pairing is impossible the message gets lost in attempt of HCA to send it down the tree.	Modes 2, 3, 4	**Low**: Only part of the cycle is not marked as NO-BRIDGE. Same as in case 3 it is likely that these edges will be marked by other NOBRIDGE messages.	**Medium**: same as in Case 2.
9	Message loss in the backward phase	Out of two NO-BRIDGE messages, both are lost before reaching HCA.	Modes 2, 4	**Low**: Same as in Case 8.	**Medium**: Same as in Case 2.
10	Message losses in both phases	After asymmetric cross-edge detection in case 3, NO-BRIDGE message is lost before or at HCA.	Modes 2, 3, 4	**Low**: Same as in Case 8.	**Medium**: Same as in Case 2.
11	Message losses in both phases	After asymmetric cross-edge detection in case 3, NO-BRIDGE message is lost after HCA.	Modes 2, 3	**High**: The message may be propagated down to the root. Some bridges may be falsely marked as NO-BRIDGE.	**High**: Articulation points may be marked as ordinary nodes.

Table 5.5.: Errors and failures introduced to the algorithm by communication faults (2).

do not change effects of individual faults.

The detection algorithm runs an equivalent of $HLD(1,1)$ for the implicit link discovery: it accepts links after a single successful packet reception (MAC broadcast) and rejects it as soon as the execution of the execution round is finished. The algorithm is memory-less with regard to topology discovery because it does not utilize the knowledge from the previous executions in successive rounds[8].

This choice of link detection strategy has both positive and negative effects to the detection process. Accepting a link after one successful MAC broadcast resolves the termination issues of link-state based biconnectivity testing algorithms (explained in Section 5.2) and reduces communication overhead (it is not necessary to execute HLDs prior to biconnectivity detection algorithm execution) but it also introduces additional errors in the biconnectivity testing. It was demonstrated in Chapter 4 that the probability of erroneous link detection of $HLD(1,1)$ is considerable for all values of link

[8]It is explained later that decisions are preserved and used by voting rules in order to provide higher accuracy of the detection, but in a single decision round no information outside of the round is utilized.

Case	Fault	Error	Failure	Severity (Bridge)	Severity (AP)
12	Search has started, node fails before it is visited by algorithm.	Topology of network is changed.	none	**None**: the node is not visited yet, so the algorithm will be executed in new topology, providing accurate markings.	
13	Search is not active, a node fails.	Topology of network is changed.	Modes 1 and 4	**High**: node failure has changed the network topology. Some of ordinary edges may become bridges and ordinary nodes articulation points.	
14	Search is active, a node fails after it has been visited by the algorithm	Topology of network is changed but algorithm operates under assumption that topology has not changed.	Modes 1, 2, 3, 4	**Very high**: consequences are numerous and difficult to estimate. It can be expected that multiple errors from Tables 5.4 and 5.5 are simultaneously encountered. Control flow of algorithm is not affected.	
15	Search is active, a node is restored, but the algorithm has not reached it yet.	Topology of network is changed.	none	**None**: same as in case 12.	
16	A node is restored, search has passed through its area.	Topology is changed and some bridges may have become ordinary edges. The node can eliminate errors by initiating new algorithm execution.	Modes 2, 4.	**Low**: Some of edges may be marked as BRIDGE although they are no longer a bridge. Restored node does not participate in search, so it cannot compromise functionality of the algorithm.	**Low**: a node believes that it is articulation point although it now belongs to a 2-connected subgraph.
17	A node is restored, search is not active.	Topology is changed and some of edges are no longer bridges.	Modes 2, 4.	**Very low**: The node can eliminate errors by initiating new algorithm execution.	

Table 5.6.: Error and failures introduced to the algorithm by node restoration and failures.

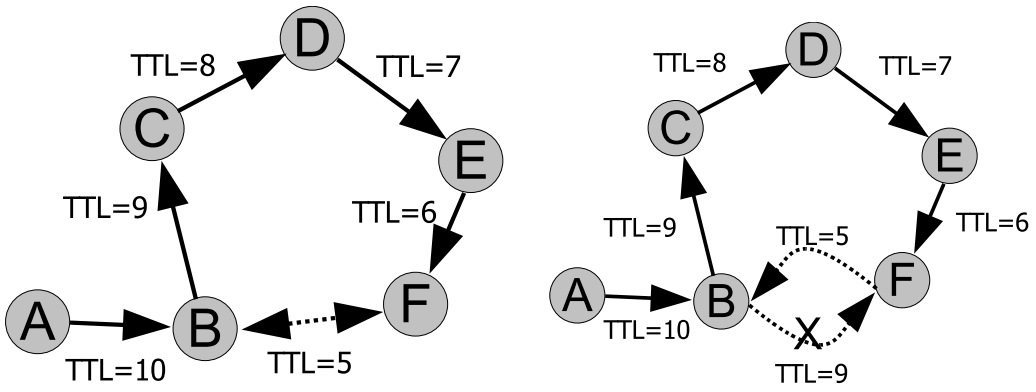
Search ID	1	2	4	6	7	8	11	12	15	16	16	16
Bridge	true	true	true	false	false	true	true	false	true	false	false	false
Hops	1	1	1	3	2	1	1	4	1	6	4	1

Table 5.7.: Example of successive contradictory markings of a link.

acceptance threshold t.

Errors caused by internal faults are described in Table 5.6. Node failure and recovery change the topology of the network, possibly invalidating the system state: new bridges and articulation points can be created by node failure or their number can be reduced by a recovered node. A node failure is particularly dangerous if it occurs during algorithm's execution: such event combines impossibility of determining effects of node's failure on network topology with communication issues – its children in the tree expect that node forwards the messages down the tree.

As a result of internal and external faults, a node may receive different markings for an edge in successive DIBADAWN executions, as the example in Table 5.7 shows (note that node may be excluded from an execution because of packet losses, and multiple



(a) If there are no packet losses, order of node visits plays no role.

(b) Asymmetric loss may result in incorrect message pairing. If BFS tree is used, this fault case is detectable.

Figure 5.8.: Effects of asymmetric losses on DIBADAWN (1).

NOBRIDGE markings within an execution).

5.5.2. Explicit Reduction of Effects of Errors

It may seem that effects of the communication faults may be reduced, assuming that every message can be delivered after large enough number of retries. Such solution is acceptable in presence of low latency, high bandwidth, low error rate, dedicated communication channels between nodes. However, in WMNs it is absolutely unacceptable since communication channel is shared among nodes. Repeated transmissions increase contention on the channel thus increasing packet loss probability in surrounding area and reducing quality or even compromising other services in the network. Furthermore, it is not difficult to envision a scenario where a sender attempts to send a message to a recipient node that has failed or moved away, so the recipient cannot receive the message regardless of the number of retries.

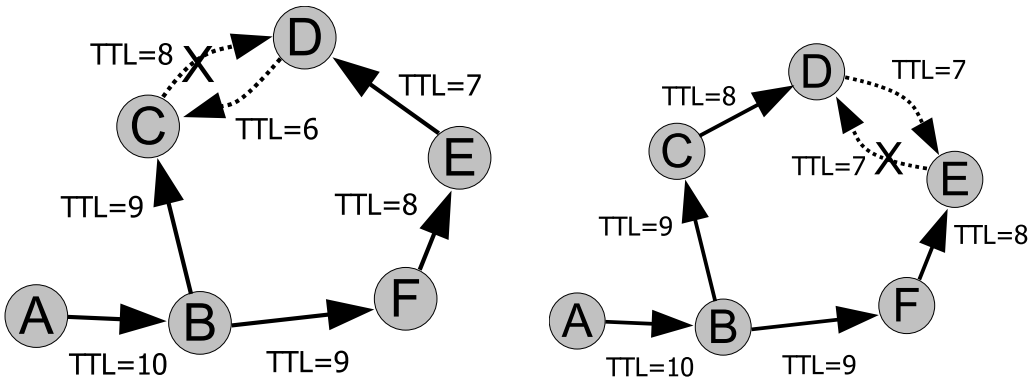
Thus, it is not possible to completely eliminate the faults from the system, but some of their effects can be reduced.

Case 1 errors can be reduced through repeated broadcast of forward phase messages by nodes that are incident to bridges. Same as for the route request messages (it will be discussed in Section 9.7), the bridges are the critical points for search propagation.

Situation shown in Figure 5.8(a) is an excellent example of an error that cannot be observed if algorithm is studied under the idealized conditions. If communication is fault-free, node B simply waits that a cross-edge marking BF reaches it over nodes E, D and C, it executes pairing and continues correct operation of the algorithm.

In presence of message losses, asymmetric cross-edge detection may occur. Their impact can be decreased if DIBADAWN is used with dBFS tree construction algorithm. In a dBFS tree, TTL values of adjacent nodes in the tree can differ only by one. If this difference is higher, it indicates a forward message loss and the asymmetric cross-edge detection.

An example of such situation is presented in Figure 5.8(b). A forward message from node B did not reach node F. Instead, it had reached node F over nodes C, D and E.



(a) Asymmetric cross-edge detection that produces both correct and incorrect markings.

(b) The undetectable asymmetric cross-edge detection in forward phase.

Figure 5.9.: Effects of asymmetric losses on DIBADAWN (2).

Node B can detect this irregularity in behavior by comparison of TTL values received from F (it should be eight, but B receives a packet with TTL of five) and acts in accordance with encountered situation. The node has three choices in this situation:

- Node B adds a NOBRIDGE message that marks cross-edge BF to its queue. Such message cannot be paired since its pair does not exist. Once it is transmitted, it can cause errors in lower tree levels. In this example, the message would mark the bridge AB as NOBRIDGE.

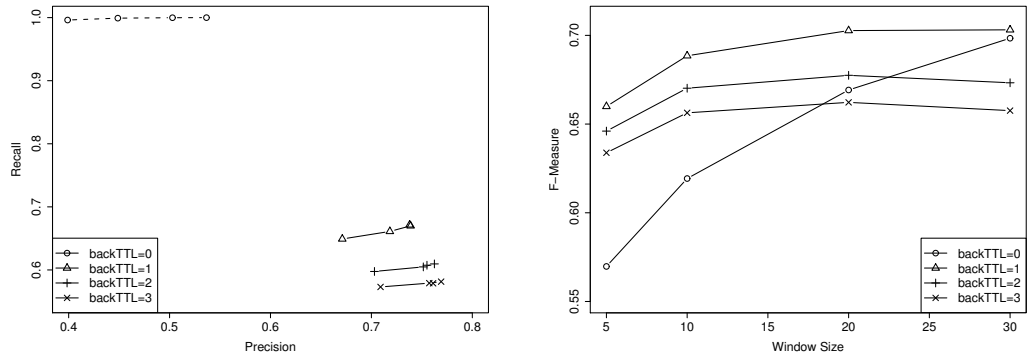
- Another broadcast by node B may lead to detection of cross-edge BF and generate the pair of the cross-edge marking at node F, but there are no guarantees that this additional broadcast reaches node F. These subsequent broadcasts may also invalidate the timeout schemes.

- Node B only marks the edge BF as cross-edge but it does not send a message down the tree. The node utilizes the available topology data but prevents possible errors.

In case that loss location is slightly different, like in Figure 5.9(a) sending of message down the tree provides both good and bad consequences: it marks correctly the edge CB but the edge AB is still erroneously marked as NOBRIDGE. In WMN topologies, cycle detection messages are numerous if compared with bridge messages, so a small gain in cycle marking is not worth the error on link AB.

In both cases, node B cannot influence markings of a set of edges (edges FE, ED, DC and CB in Figure 5.8(b)), but if it does not send a NOBRIDGE message of asymmetrically detected cycle it prevents error propagation to lower tree-levels. This is considered to be favorable behavior and implementation of the algorithm supports it.

An example of Case 3 error from Table 5.4 is shown in Figure 5.9(b). The loss occurs during cross-edge detection but the BFS tree is correctly formed, so the error cannot be detected. Also, the error cases from Table 5.5 are undetectable. They result in incorrect markings and degrade the accuracy of the algorithm.



(a) Precision-Recall plot.

(b) F-measure.

Figure 5.10.: Accuracy of bridge detection in DIBADAWN if limited propagation of NOBRIGDE messages us used.

It seems that the major sources of faulty decisions are the erroneous NOBRIDGE markings, that are forwarded below their HCAs. When such a marking travels down the tree, it marks all the links it traverses as non-bridges, even if some of them are actually bridges. Additionally, it causes errors in the transitive closure of relation $\circledcirc$, creating errors in the articulation point detection.

In order to reduce effects of erroneous NOBRIDGE marking forwarding, TTL value of backward messages was reduced to several hops in [120]. In that version of the algorithm, only unanimous voting rule was supported. The parameter w took values $\{5, 10, 20, 30\}$. The results of this approach are presented in Figure 5.10. It can be seen that a choice must be made – either to choose high recall and very low precision, or to choose very low recall and to obtain acceptable precision. In [120] the parameter w takes rather large values, limiting algorithm's capability to adapt to topology changes.

These results were not satisfying and it was obvious that a more subtle, less invasive solution for resolving the effects of faults to the algorithm is required, with a shorter reaction time to topology changes (i.e., smaller sets of votes from previous algorithm executions should be sufficient). The consequence of evaluation results in [120] was rejection of concept that messages in backward phase of algorithm should be restricted in their forwarding. The idea that more than one algorithm execution can be used for decision making demonstrated its potential, and in the next section it is developed in more detail.

5.6. Improving Algorithm's Accuracy by Voting

The goal of the algorithm can be described as follows: from a set of markings it has received, a node decides if it is an articulation point, and for each of its incident edges if it is a bridge. As explained in the previous section, the stochastic properties of communication channel impact the algorithm's functionality, so it is possible that some of the markings are incorrect. Thus, a node faces a set of binary choices under uncertainty [131] [150].

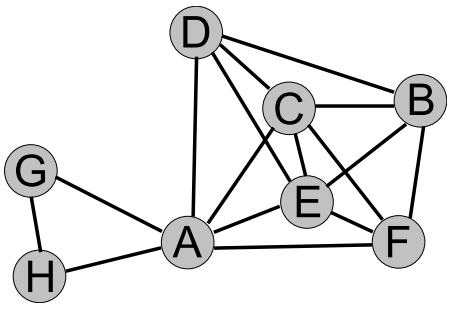


Figure 5.11.: Utilizing different network views for accuracy improvement.

It was shown by Marquis de Condorcet [111] in 1785 that if the majority voting is used by a set of voters with identical probability of selecting the correct hypothesis, their joint decision has a higher probability of being correct than decision of a single voter (assuming that the probability of selecting the correct hypothesis of a voter is larger than 0.5).

The strategy of multiple voters is adopted in this work and extended by introduction of other decision strategies/voting rules. The votes are delivered by successive DIBADAWN executions and various voting rules are applied to them in addition to the majority voting.

The successive votes in bridge and articulation point detection are do not have identical accuracy as it is assumed in [111]. The successive searches in which a node participates may originate from different nodes, so the perceived topology and resulting decisions may be different.

This property may work in favor of the detection process. Let us observe an example network in Figure 5.11. If search originates at nodes G or H, a single improper NO-BRIDGE message pairing at node A will result in its marking as an ordinary node (a false negative in articulation point detection). Due to high node density in part of the network which is depicted right from node A, collisions and asymmetric detection of cross-edges is likely, as well as the subsequent erroneous marking of node A. However, if search is started by one of nodes A, B, C, D, E, it is more likely that the short cycle AFG is detected and its messages properly paired at A, resulting in correct marking of node A as an articulation point. Thus, depending on the source of the search, node A observes the network from different perspectives through backward-phase messages it gets. The probability of correct detection varies among perspectives, so if A makes a decision based on several perspectives (observations), it may increase the probability of making a correct decision.

5.6.1. Voters, Votes and Voting Rules

There exists a set of voters (also called decision makers) $\{1, .., n\}$. They are supposed to make a binary decision under uncertainty. Each of the voters selects an option out of two that are offered, in accordance with its understanding of the problem. It is assumed that all voters share the same view of the decision task (share a common utility function), i.e., there are no malicious voters that deliberately act against consensus and selection of correct option. In the general choice-under-uncertainty model, voters may be undecided

(undecided voters give a neutral vote), but in case of DIBADAWN, neutral votes do not exist. There is no difference in terms of cost between incorrect decisions – the same penalty applies to false positives and false negatives.

Definition 5.5 *Let us name the options available to voters as a and b. If voter i chooses option a, its vote is* $x_i = 1$ *and if it chooses option b, its vote is* $x_i = -1$.

In this work, the set of *voters* is the *set of latest executions* of DIBADAWN algorithm. They deliver votes for node participating in a DIBADAWN execution (if it is an articulation point) and for edges incident to it (whether they are bridges). The votes are then used by voting rules to deliver the final decision on status of a node and its incident edges. As it is shown in Figure 5.1, the votes may originate directly from the DIBADAWN algorithm (in the first voting round) or the decisions of the first voting round can be forwarded to a new set of voting rules (in the second voting round).

Definition 5.6 *Hypotheses in the bridge/articulation point detection process are associated as follows:*

$$\begin{cases} a & \textit{edge is not a bridge / node is not an articulation point} \\ b & \textit{edge is a bridge / node is an articulation point} \end{cases}$$

In a concrete situation, at a time t, each of the voters in the voter set supports a hypothesis. The function which maps the voters to the set of all possible vote combinations is called a voting profile.

Definition 5.7 *A voting profile* $x = \{x_1, x_2, .., x_n\}$ *is a function that maps a finite set of voters* $\{1, 2, .., n\}$ *to set* $\{-1, 1\}^n$.

Definition 5.8 *A decisive voting rule* f *is a function that maps set of all voting profiles to the set* $\{-1, 1\}$.

A rule is indecisive if for a given set of voters it cannot always select one of hypotheses. For instance, a rule that requires that more than half of votes support a hypothesis in order to select that hypothesis (the majority rule) is indecisive if applied in voter sets consisting of even number of voters – it may happen that voters are divided into two equally sized sets. The same rule is decisive in odd-sized voter sets. The indecisive rules are usually resolved by a tie-breaker, or they are simply not applied in scenarios unsuitable to them.

In this work, the constant ϵ is used as tie-breaker for majority rules. ϵ should be a small positive number if default behavior of rules is to decide against the bridge/articulation point hypothesis (approach that is chosen in this thesis due to higher probability that an edge is not a bridge) or a small negative number if voting rules should decide in favor of bridge/articulation point hypothesis.

Only one of options a or b is correct with regard to the output of detection algorithm applied to ideal global topology knowledge[9]. Due to uncertainty in the system, a voter may support the correct answer or vote against it.

[9] Of course, this is unknown to voters nor to node that manages markings, votes and decisions. The correct markings are known only to the omniscient observer who is completely detached from DIBADAWN.

Definition 5.9 *There exists a random variable y_i associated with voter i such that it takes value 1 if it selects correct option and -1 if it selects incorrect option. The competence of a voter i is probability that it votes for the correct hypothesis: $p_i = p(y_i = 1)$. Vector of abilities is defined as $p = (p_1, p_2, .., p_n)$ and it is associated to the set of voters $\{1, 2, .., n\}$.*

There are numerous strategies for decision making under uncertainty. They start from different assumptions in order to minimize the probability of erroneous decisions. However, minimization of error probability requires deep and precise knowledge of the system, such as the apriori probability of hypotheses, vectors of abilities, constant voter abilities in time and space, etc.

The WMN environment is not as controlled as it is required for direct and accurate application of these known results. Additionally, the required apriori probabilities can only be estimated for an application scenario. If new nodes are added or some of them fail, the apriori probabilities of hypotheses occurrence may be changed. Changes in network's environment may affect the voters' abilities. The shared communication medium invalidates common assumption of voter independence. This is obvious, but it is extremely difficult to capture and quantify these dependencies and their influence on the algorithm.

Due to these uncertainties, a set of rules is proposed both for bridge and articulation point detection. Some of them are taken from the decision theory, and some are extended with the knowledge of algorithm's functionality (the so-called trusted rules). The goal of the whole voting procedure is to maximize the utilization of the available information contained in DIBADAWN's markings and to reduce the error probability.

The voting rules are executed at individual nodes. They do not require inter-node synchronization or any other type of message exchange since their input originates from already finished algorithm executions. Thus, their execution does not put additional communication load on the network, but they may improve accuracy of decisions (as their evaluation in Chapter 9 shows). Finally, their implementation is not complex, so the additional processing load for nodes is very low.

5.6.2. The First Round of Voting

In order to compensate for the uncertainty in the algorithm caused by communication and node faults, decisions are not made based only on one algorithm execution, but on the set of last k algorithm executions.

For bridge detection, voters are equivalent to edge markings. A marking of a node used for articulation point voting is the output of the method **APdetection**.

The voter set is the set of last algorithm executions with size k. The voter set is not constant in time and its composition and vector of abilities changes in time. As a new voter arrives (output of latest execution of DIBADAWN) it replaces the oldest voter in the voter-set. Value of k is a parameter of the protocol and it can be varied. The number of voters in a profile is v. For articulation point voting rules, v is the same as k. For bridge voting it may be larger than k, since an edge may belong to more than one cycle, receiving multiple NOBRIDGE markings in one algorithm execution.

Increasing the value of parameter k improves accuracy of decisions in static topologies. However, it should not be excessive because even in static networks the topology may

change because of node failures and rejuvenation, and the outdated decisions valid in the old topology can influence the decisions in new, changed topology, resulting in prolonged incorrect decisions.

Voting Rules for Bridge Detection

For bridge detection, voters are associated with edge markings. Detailed study of DIBADAWN functionality enables classification of the markings into four categories:

1. Cross-edge detection in method **detect_cycles** can be trusted. The cross-edge and the tree edge over which the cycle was detected are not bridges (e.g., edges EG and EC in Figure 5.5).

2. If a node pairs two markings in the method **forward_msg**, the edges over which they have arrived are not bridges. This decision can be also trusted (e.g., edges BC and BD in Figure 5.5).

3. For all other cases of non-bridge markings, the competence belongs to interval $(0, 1)$.

4. Bridge markings competence belongs to interval $(0, 1)$.

The messages from categories one and two are easily identified and they are used by the trusted rules. Strictly speaking, competence of trusted votes is not always one, since the detected cycle may contain edges with quality below the link acceptance threshold. Still, the evaluation in Chapter 9 will show that trusted rules have highest accuracy of all bridge voting rules, despite these errors. The competence of messages from category three and four can be statistically estimated on a set of known topologies.

The following voting rules have been selected for implementation and evaluation:

1. **Unanimous**: an edge is considered a bridge only if all votes support the bridge hypothesis.

$$f_1(x) = Sign(\sum_{i=1}^{v} x_i + (v - \epsilon)) \tag{5.6}$$

2. **Plain Majority**: votes for bridges and non-bridges are equally valued. The hypothesis with more votes is chosen. If both proposals have the equal number of votes, the edge is declared as a non-bridge (the tie-breaker constant $\epsilon > 0$).

$$f_2(x) = Sign(\sum_{i=1}^{v} x_i + \epsilon) \tag{5.7}$$

3. **Single-for**: this rule is the opposite of the unanimous rule. It is sufficient that a single vote supports the bridge hypothesis to declare the edge as a bridge.

$$f_3(x) = Sign(\sum_{i=1}^{v} x_i - (v - \epsilon)) \tag{5.8}$$

4. **Intelligent Majority**: if none of the votes is a trusted vote, the plain majority rule is applied. If there is a trusted vote, the edge is declared to be a non-bridge.

$$f_4(x) = Sign(\sum_{i=1}^{v} w_i x_i + \epsilon) \tag{5.9}$$

where

$$w_i = \begin{cases} v & \text{if } x_i \text{ is trusted vote} \\ 1 & \text{otherwise} \end{cases}$$

5. **Trusted non-bridge rule**: unless a trusted NOBRIDGE vote exists, the edge is declared to be a bridge.

$$f_5(x) = Sign(\sum_{i=1}^{v} w_i x_i - \epsilon) \tag{5.10}$$

and

$$w_i = \begin{cases} 1 & \text{if } x_i \text{ is trusted vote} \\ 0 & \text{otherwise} \end{cases}$$

6. **Weighted rule**: a weight is assigned to each vote, based on the competence of voters. It is further assumed that the apriori probability α of non-bridge occurrence in a network in known. The competence of votes is a function of number of hops a marking has been forwarded in the network. In order to maximize the probability of making the correct decision, weight of a vote of a decision maker with competence p_i should be $w_i \sim log\frac{p_i}{1-p_i}$ [150]:

$$f_6(x) = Sign(\sum_{i=1}^{n} w_i x_i + \ln \frac{\alpha}{1-\alpha}) \tag{5.11}$$

The results of [150] imply that weight of a single vote is independent of all other votes, which is a consequence of the assumption that voters are independent.

Notes on the weighted rule: Figure 5.12 shows the voter competence as a function of the hop-number. The competencies are derived from simulation results of four different simulation setups, using Rayleigh and Ricean propagation models on topologies that resemble community networks in Berlin and Leipzig, in accordance with the simulation methodology described in Section 9.4. The values shown in the Figure 5.12 are used for evaluation of voting rules in Chapter 9.

The behavior of vote competence has two distinguishable parts. At the very beginning (if hop count is less than two), the influence of incorrect bridge markings results in overall competence of 0.9-0.95. It returns to almost ideal competence if hop count is equal to two and then it slowly decreases as the number of hops increases.

The role of a voter in a weighted rule may vary, depending on the composition of a voting profile. In some voting profiles a voter is dominant (its vote is decisive) while the same voter may have absolutely no impact on voting outcome in another voting profile (it provides the "dummy" vote). For instance, the first voter dominates in a set of three

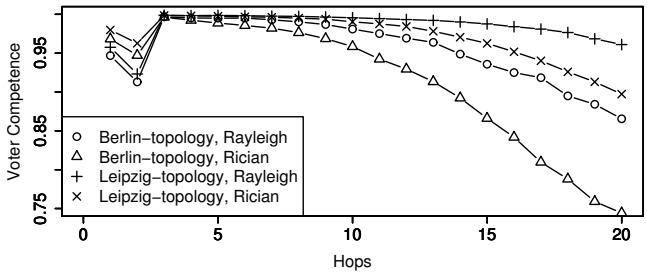


Figure 5.12.: Competence of DIBADAWN markings as a function of number of hops it was forwarded.

Competence	0.6	0.61	0.7	0.71	0.85
Weight	0.4055	0.4473	0.8472	0.8953	1.7346

Table 5.8.: Example of optimal weight assignment based on [150].

voters $\{x_1, x_2, x_3\}$ with the vector of abilities $\{0.7, 0.6, 0.6\}$. But the pair voters x_1 and x_2 are dominated by the voter x_3 if the vector of abilities is $\{0.7, 0.7, 0.85\}$. In both cases the rule is reduced to expert voting – the sum of weights of lower competence voters is lower than the weight of the single expert so they cannot overrule it. However, even a slight increase in competence of a single voter may return control to the whole group. For instance, the ability vectors $\{0.7, 0.61, 0.6\}$ and $\{0.7, 0.71, 0.85\}$ require participation of all voters in order to minimize the probability of decision errors. Similar behavior can be observed in larger voter sets, where a subset of voters may dominate the whole set.

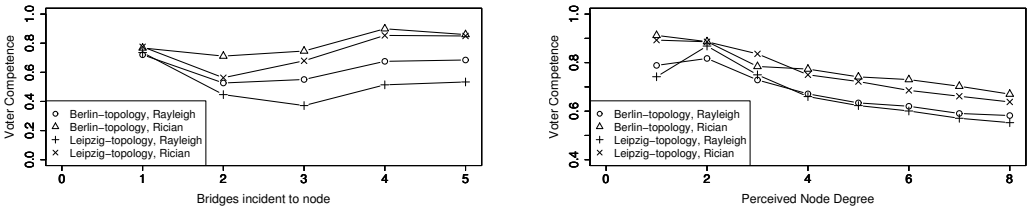
The competence of voters is estimated on a limited set of algorithm executions on a limited set of known topologies (training phase) and they are applied in later algorithm evaluation. The learning phase should be executed for each new deployment scenario: for instance, the competencies obtained from sparse topologies of Freifunk Berlin network may be inappropriate in a dense uniform topology. Additionally, real network is changing and evolving in time, so competence values require constant upkeep since they may be invalidated after several months or years of use. The effects of inaccurate weights on accuracy of weighted decision rules will be evaluated in Section 9.4.3.

Voting Rules for Articulation Point Detection

The **unanimous**, **single-for**, **plain majority**, two **weighted**, and two **trusted** rules are proposed for the articulation point detection.

The first three rules are exactly the same as for the bridge voting. Weighted rules follow the same principles as the weighted rule in Equation 5.11. The weights are no longer differentiated by hop count of markings (it is not applicable in this context) but by:

- Number of bridges incident to a node.
- Node degree.



(a) Competence as a function of number of perceived incident bridges.

(b) Competence as a function of perceived node degree.

Figure 5.13.: Competence of voters for articulation point detection.

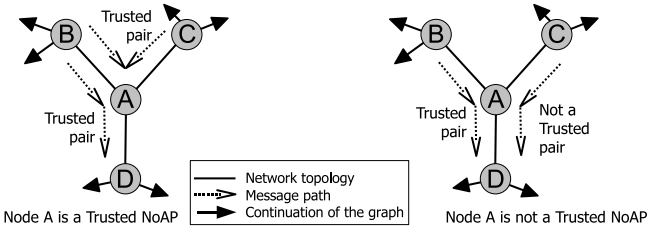


Figure 5.14.: Trusted rule for articulation point detection.

If a node has bridge markings on links incident to it, it is more likely that it is an articulation point. In the deterministic, loss-free detection algorithm a bridge incident to a node rises suspicion that the node is articulation point (pendant nodes are not articulation points although they are incident to a bridge) and a node with two or more incident bridges is an articulation point. Due to packet losses, there is no guarantee that a bridge observed in DIBADAWN is an actual bridge, so this is only a probabilistic metric even if a node observes two or more bridges incident to it.

This competence metric has a two-fold and unexpected behavior (Figure 5.13(a)): competence of voter that observed one bridge is higher than the voter that observed two bridges. If a node observes more than three bridges the competence grows with increase in number of observed bridge markings. The competencies that are assigned to voters may be very low as in simulation scenario where a topology similar to a WMN in Leipzig is used and the Rayleigh propagation model.

The competency of a voter that is represented as a function of perceived node degree has a clear trend – as expected with node degree increase, the probability that a node is an articulation point decreases (Figure 5.13(b)). It is important to notice that for a given environment, shape of network topology does not play an important role in shaping of competence: two simulation cases with Ricean and two with Rayleigh propagation model exhibit surprisingly similar behavior. This is important for later application of the approach – shape and properties of network topology may change during network growth, but the Figure 5.13(b) indicates that changes will have only a limited influence to this competence metric.

The **trusted** rule for articulation point detection performs the equivalence class calculation based only on the trusted NOBRIDGE markings. All other markings are ignored

in the transitive closure. In the left example in Figure 5.14, node A is declared as a trusted non-articulation point: a pair of trusted messages is forwarded from node B to node D over A ($BA \circledcirc AD$) while messages delivered over edges BA and CA were paired at A ($BA \circledcirc CA$). The transitive closure deduces that $AD \circledcirc CA$, thus all edges belong to the same class, and node A is a trusted no-AP (outcome of the trusted rule is `false`).

The counter example is given at the right side of the figure. If standard procedure for calculation of equivalence classes is used, we have $BA \circledcirc AD$, $CA \circledcirc AD \Rightarrow BA \circledcirc CA$ and the node is not an articulation point. However, if only trusted NOBRIDGE markings are considered, $BA \circledcirc AD$ and the link CA belongs to another equivalence class (since it has no trusted markings). Node A is thus declared as an articulation point by the trusted rule.

The rule is extremely conservative and eager to tag a node as an articulation point. In particular if the node is distant from cross-edges and if it is not a HCA (thus it lacks the trusted NOBRIDGE markings like for instance node B in Figure 5.4, at page 80). Because of this, the rule produces numerous false positives and its precision is reduced. It may also fail to detect articulation points if there exists a subset of edges unknown to the node (no markings were delivered over it), thus its recall, although high, is not ideal.

5.6.3. The Second Voting Round

The definition of rules from the first voting round indicates that they have different characteristics: for example, the **single-for** rule reacts promptly to the possibility that a node is an articulation point – it is to expect that it will have high recall and low precision; the **unanimous** rule requires that all votes agree before it decides that an edge is a bridge or a node an articulation point – it is to expect that its precision will be high with a questionable recall. Other rules (e.g., trusted and weighted) have completely different logic and it is difficult to predict their behavior – their characteristics depend on topology shape and environmental factors.

The second voting round is introduced in order to:

- Obtain the balance between precision and recall. Instead of polarized rules with exceptional precision **or** exceptional recall, decisions with good precision **and** good recall are desirable.
- Obtain stable rules. Second voting round rules should provide accurate decisions independent of topology or environment. The behavior of some rules may be extremely variable: in one type of scenario the rule performs well but it under performs in another. For instance, the weighed-bridge rule for articulation point detection has decent characteristics in simulations with Ricean propagation model and very poor if Rayleigh fading is introduced (Appendix D).
- Improve one characteristic while preserving the other. For instance, keep the recall constant and improve the precision.
- If possible, extract additional bits of information from existing markings and decisions, and use them to further improve accuracy of decisions. Even if no major improvements are obtained, every improvement is valuable since it comes "for free" (without additional communication overhead).

In order to achieve these goals, the voting process is repeated, analogously to the first round: a set V_1 (**V**otes from the 1^{st} voting round) is selected and a rule is applied to it. The second voting round rules belong to one of three categories and each category needs progressively more information for its operation:

- The simplest strategy is to use the majority voting on set V_1. No learning phase for the second voting round is required.
- Apply the weighted voting [150] to decisions from the first round. The competence of rules from the set V_1 (Definition 5.9) must be evaluated under controlled conditions (learning phase).
- Apply the Bayes classifier to the set V_1.

Note on the Bayes classifier:
During the learning phase, two probabilities must be estimated for each rule r that belongs to the set V_1: the probability that voter r has selected a (voted 1) if a is the correct hypothesis $P_r(v_r = 1|a)$, and the probability that voter r has selected b (voted -1) if b is the correct hypothesis $P_r(v_r = -1|b)$ (a, b and values associated to them are explained in Definition 5.5). Since a and b are mutually exclusive, the probabilities $P_r(v_r = -1|a)$ and $P_r(v_r = 1|b)$ are also known. The apriori probabilities of hypotheses $P(a)$ and $P(b)$ are estimated in the learning phase.

Voter profile from the first voting round $V_1(t)$ is a function of time. It is updated with new values at the end of each DIBADAWN execution. Each time $V_1(t)$ changes, Bayesian classifier is applied to it. Probabilities $P(a|V_1(t))$ and $P(b|V_1(t))$ are calculated and the more-probable hypothesis is chosen. These probabilities are obtained using the Bayes rule:

$$P(a|V_1(t)) = \frac{P(V_1(t)|a)P(a)}{P(V_1(t))} \qquad P(b|V_1(t)) = \frac{P(V_1(t)|b)P(b)}{P(V_1(t))} \tag{5.12}$$

If we apply approximation that voters are mutually exclusive, the so-called naive Bayes classifier is obtained. In practical applications of Bayes classification this assumption is frequently applied despite the fact that in vast majority of cases the input variables are not independent. It is proven in [104] and [178] that the naive Bayes classifier may provide the optimal decisions (least probability of misclassification) in some cases, and that the dependence among variables may cancel each other, explaining why it often provides surprisingly accurate classification.

The Equation 5.12 is then transformed into:

$$P(a|V_1(t)) = \frac{P(a)\prod_{v_i \in V_1} P(v_i(t)|a)}{P(V_1(t))} \qquad P(b|V_1(t)) = \frac{P(b)\prod_{v_i \in V_1} P(v_i(t)|b)}{P(V_1(t))} \tag{5.13}$$

In order to perform the classification, the node only has to calculate the probabilities $P(a|V_1(t))$ and $P(b|V_1(t))$ and choose the hypothesis with a higher probability. The probability $P(V_1(t))$ is not important for the comparison and it needs not be calculated since it is the divisor in both expressions. Calculating the naive Bayes classifier is not computationally intensive (requires only several floating point operations) and can be used even by nodes in a sensor network.

Composition	Intended purpose
U, M, T, wB, wD	Use input from all distinctive rules to improve overall stability and if possible, F-measure of the rule. Slight preference to precision.
Sf, M, T, wB, wD	Use input from all distinctive rules to improve overall stability and if possible, F-measure of the rule. Slight preference to recall (compared with the previous voter set, the unanimous is replaced by the single-for rule).
Sf, M, wD	Balanced rule, envisioned for use in weighted voting or Bayes classifier. Slight preference to recall improvement.
Sf, T, wD	Balanced rule, envisioned for use in weighted voting or Bayes classifier. Trusted and weighted-degree rules improve precision, single-for balances with its exceptional recall.

Table 5.9.: Rules in the second round of articulation point voting.

Full rule name	Short name	Full rule name	Short name
Unanimous	U	Single-for	Sf
Majority	M	Trusted	T
Weighted-bridge	wB	Weighted-degree	wD

Table 5.10.: Rule legend.

The Bayes classification, as presented here assumes equal penalty for misclassification of both of hypotheses. If it is not the case in application scenario[10], the classifier may be extended to the "Bayes classifier with costs" that assigns misclassification costs to each of hypotheses. The cost is always dependant on the application scenario and cannot be generally assigned. Since the goal of this thesis is the general framework for bridge and articulation detection we have not included this version of the classifier in discussion. ♢

The last, and maybe the most important question is how to form the voter set V_1. Its composition should be chosen to meet the desired goal: if the goal is maximization of precision, the rules with high precision are to be included in V_1. The same principle applies for recall maximization. A mix of rules with heterogenous characteristics provides the best results if the goal is balancing of rule characteristics.

Some combinations of voter sets and voting rules may degenerate into expert voting. An example how a weighted voting rule may degenerate to the expert voting has been already presented in this section (page 101). Analogous cases may be constructed for the Bayesian classification. The second-round majority rules may be affected by this phenomenon as well. For instance, if V_1 comprises of unanimous, single-for and majority votes, decision of second-round majority voter is always equal to decision of the majority voter. Thus, it is important to avoid such membership of the set V_1.

[10] A classical example is the medical diagnostics, where the cost of error of not diagnosing an illness is very high (it endangers a patient), while raising a false alarm results in additional tests (it slightly increases financial costs of the diagnostics).

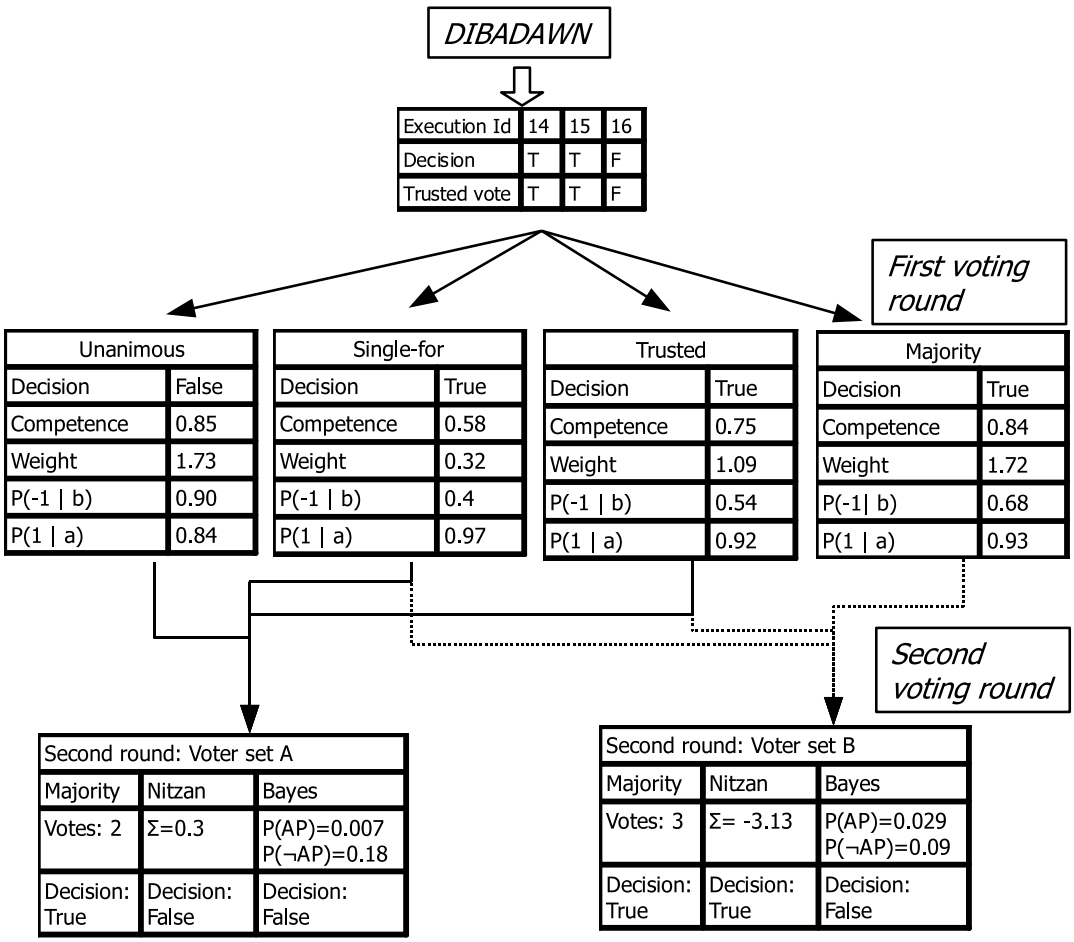


Figure 5.15.: An example of the second voting round.

The general framework supports second-round voting both for bridge and articulation point deciding process. Low gains are expected if the second round voting is applied in bridge detection, since there already exist rules that use information on internal algorithm functionality (trusted and intelligent-majority). Also, there exists insufficient variability of bridge-voting rules: single-for, majority and unanimous are vote-count based, two expert voting strategies (trusted and intelligent-majority), and one weighted rule. Thus, the number of possibilities for improvement is drastically reduced (e.g., there is no point in joining majority and trusted rule, when they are already ”joined” in the intelligent-majority rule). Several second-round voting strategies for bridge detection have been implemented and evaluated but the gains they provide are minor so they are excluded from this work.

Articulation point voting brings more possibilities for vote building so the possible gains obtained by it are more pronounced and worth of further investigation. Table 5.9 shows the rules which bring concrete benefits and which are evaluated in detail. The legend of rule abbreviations is in Table 5.10.

Due to partial unpredictability of outcomes of the second voting round, more input sets have been evaluated than it is shown in Table 5.9. The sets which did not bring

significant improvements in decision accuracy are not included in Table 5.10 nor in Chapter 9.

Figure 5.15 shows an example of the proposed two-round voting scheme for articulation point detection. DIBADAWN delivers a set of last decisions to a queue that (in this example) keeps only the three freshest decisions. Based on these decisions, voting rules from the first round calculate their decisions. It can be seen that the trusted (trusted-flags are part of the vote queue), majority and single-for rule support the hypothesis of articulation point existence, while the unanimous rule votes against it (one `false` vote in the queue).

In the second voting round, output of these four rules is used in two different combinations.

The voter set A in the second voting round consists of the output of the unanimous, single-for and majority rules from the first round. If majority voting is applied to the voter set A, the hypothesis that node is an articulation points is supported. If weighted rule or Bayes classifier are used on it, it is decided against this hypothesis.

The voter set B consists of the output of the trusted, single-for and majority rules from the first round. If majority or weighted voting rules are applied to it, the hypothesis that node is an articulation points is supported. If Bayes classifier is applied to it, it decides against this hypothesis.

5.7. Summary

The existing bridge and articulation point detection algorithms for WMNs [57] [80] [168] require global topology knowledge. They are simple, fast, and efficient if accurate network topology is known. However, the probability of obtaining precise topology information in a wireless multi-hop network is extremely low as it was shown in Section 4.4. Under realistic conditions where packet losses are frequent, the ability of proactive management protocols to deliver topology data of adequate quality for biconnectivity testing degrades sharply, despite the large communication overhead of $O(n^2)$ they create.

DIBADAWN is a distributed bridge and articulation point detection algorithm for wireless networks, where nodes are cooperating through message exchange. It does not depend on the global topology knowledge so it scales better than the proactive approaches and has message complexity of $O(n)$.

The algorithm combines ideas of algorithms from [55], [67], and [158]. It is altered so that it can operate in presence of packet losses and node failures, that are inevitable in reality. These events cause errors in the algorithm and it may produce incorrect decisions. The effects of these faults have been studied and major error cases have been identified. Based on the analysis, it was concluded that the explicit error detection procedures in the algorithm are only partially possible.

A set of two-tier voting rules is proposed as a mechanism for improvement of accuracy of decisions. The rules are applied to a set of latest decisions of DIBADAWN, thus they do not increase the communication overhead. Special care has been taken to create dissimilar voting rules – some of them aim at high precision, such as the unanimous, while others aim at high recall. This provides an exceptional flexibility to the detection approach – without any changes in DIBADAWN and without any additional communication overhead, it is possible to obtain decisions with different characteristics. Since

rules are mutually independent, they can be executed simultaneously, providing their decisions to network, transport or application layer in accordance with their needs. So for instance, the network layer can obtain high-precision decisions, while the application layer simultaneously utilizes high-recall decisions.

The proposed two-round voting mechanism has potential for improvement of accuracy of decisions but it also puts burden of increased parameter space that has to be searched in order to obtain stable and high-quality decisions. Size of sets S_{ap} and S_b, identification of potent rules in the first voting round, combining output results from the first round for the second round must evaluated and compared under various conditions before reaching a conclusion on their capabilities. The evaluation results in Chapter 9 will show that the development and management of voting rules are worth the effort since the voting rules increase accuracy of detection and provide stability of the behavior of the detection approach over various scenarios.

6. Locality in Wireless Multi-hop Networks and Estimation of the Average Cycle Size

Discussion of errors in algorithm caused by message losses in the previous chapter indicates that short cycles are beneficial for the accuracy of the algorithm – in long cycles it is more likely that a marking that travels toward the HCA of the cycle gets lost.

This chapter assesses the average cycle size in random geometry graphs. The theory of random geometrical graphs (RGG) and homogeneous Poisson point processes are cornerstones of evaluation of topological properties in WMNs (e.g., [36] [78] [107]). The RGG theory cannot capture all characteristics of WMNs like packet losses or effects of obstacles on network topology. Yet, with all its drawbacks, it still gives us deeper understanding of underlying topological processes and more general conclusions on studied properties than pure simulation studies, where certain combinations of parameters may drive us to wrong conclusions.

Despite these simplifications, the theory of random geometric graphs is very expressive and may result in unsolvable models that provide only high-level conclusions. Although they are valuable, such results often cannot be applied in practice. For instance, Penrose [136] provides a detailed percolation model of RGGs that can be used for calculation of their connectivity properties. However, the model cannot be solved analytically and it is burdened with redundancies. Even after simplifications that are proposed by Quintanilla and Torquato [142] the number of integrals to solve grows exponentially with number of vertices in a graph[1], so it cannot be applied in practice where networks may have hundreds or thousands of nodes.

A more practical approach to the problem is chosen in this chapter. Instead of unsolvable models, upper bounds of studied characteristics are calculated. The approximations are frequently used in models developed in this chapter: the differences between reality and the starting assumptions of random geometric graphs are obvious, so even if ideal RGG model is developed and solved, it inescapably deviates from values encountered in reality.

Two approaches are used for estimation of the average cycle size in RGGs. The expression for the expected average face size in planarized graphs is the firm upper bound on the average shortest cycle size. The bounds are derived for Gabriel and Relative Neighborhood Graphs in Section 6.1. A more detailed model that accounts for area-boundary effects has been presented in [118]. In this chapter only the unbounded model is presented since its results are comparable to the results of the complex boundary-aware model.

The bounds of the average face size are further improved in Section 6.2. The probability of occurrence of shortest cycles of size three and four is calculated, allowing

[1] Appendix A contains additional details.

estimation of the worst-case average shortest cycle size. The Gabriel graph model is better in sparse graphs while the later model provides stricter bounds in dense graphs.

6.1. Estimation of the Expected Face Size in Gabriel and Relative Neighborhood Graphs

The average face size after planarization of a geometric graph is estimated in this section. Faces are cycles by definition, and since planarization process transforms connectivity graph $G(V, E)$ in $G'(V, E')$ with $E' \subset E$, the average minimum cycle size in the original graph cannot be larger than the average face size in the planarized graph.

Since $RNG \subset GG$ [132] [163], the average face size in Gabriel graphs provides tighter bounds and is of higher importance for this work. The results for RNGs are derived for completeness since the process is analogous to the GGs.

6.1.1. The Ratio of Removed Edges in Planarization Process

Let us observe an arbitrary node C within a homogeneous Poisson point process with intensity λ. Let $P_{neigh}(r, \phi)$ be the probability that there exists a neighbor node D at distance r from the node C under the angle ϕ measured from an arbitrary but fixed axis, and let W be the witness area (as defined in Section 2.2.4). Let $P_W(r, \phi)$ be the probability that there exists a witness node W_{CD} in W.

Edge CD is removed with probability P_{remove} if there exists a neighbor node D on distance r from C and there is a witness node W_{CD} in the area W:

$$P_{remove}(r, \phi) = P_{neigh}(r, \phi) \cdot P_W(r, \phi). \tag{6.1}$$

Integration of this probability over the whole area in which the node C can communicate, provides the expected number of deleted edges $\bar{e_d}$ for a single node:

$$\bar{e_d} = \int_0^R \int_0^{2\pi} P_{drop}(r, \phi) dr d\phi \tag{6.2}$$

In order to calculate the ratio of removed links η at the node C, the number of deleted edges is divided by the expected vertex degree $d(C)$ of the node C:

$$\eta = \frac{\int_0^R \int_0^{2\pi} P_{drop}(r, \phi) dr d\phi}{\int_0^R \int_0^{2\pi} \lambda(r, \phi) dr d\phi} \tag{6.3}$$

The vertex C is arbitrarily chosen and placement process is homogeneous $\lambda(r, \phi) = \lambda = const$, thus the Equation 6.3 is valid for every node in the graph and to the whole graph. Since λ is constant in a homogeneous Poisson Point Process, the equations 6.1, 6.2 and 6.3 do not depend on the angle ϕ. The probability $P_{neigh}(r)$ is in this case:

$$P_{neigh}(r) = 2\pi r \lambda \tag{6.4}$$

In order to calculate the ratio of removed links we need to determine the probability that there is at least one node in the witness area W.

Let us observe the homogeneous placement process of n nodes in an area A. The probability that a node is placed in an area B within the area A is $P_B = \frac{|B|}{|A|}$. The

probability $P_k(B)$ that exactly k nodes are in B is:

$$P_k(B) = \binom{n}{k} P_B^k \cdot (1 - P_B)^{n-k} \tag{6.5}$$

If the probability P_B is small and number of nodes n large, the binomial distribution can be approximated with the Poisson distribution:

$$P_k(B) = \frac{(n \cdot P_B)^k \cdot e^{-n \cdot P_B}}{k!} \tag{6.6}$$

Let the size of area A grow to infinity, keeping the ratio $\lambda = \frac{n}{|A|}$ constant. Then, $n \cdot P_B = \frac{n \cdot |B|}{|A|} = \lambda |B|$. The probability that there exists at least one node in B is:

$$P_W(B) = \sum_{k=1}^{n} P_k(B) = \sum_{k=1}^{\infty} \frac{(\lambda|B|)^k}{k!} \cdot e^{-\lambda|B|} = (e^{\lambda|B|} - 1) \cdot e^{-\lambda|B|} = 1 - e^{-\lambda|B|} \tag{6.7}$$

where we have used the power series of $e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$.

Gabriel Graphs

For Gabriel graph, witness area W is defined as the circle placed between the nodes A and B (Figure 2.5(b), page 25). For $\delta(A, B) = r$, the circle has radius of $\frac{r}{2}$ and the probability that at least one witness node W_{AB} is present within it is:

$$P_W(r) = 1 - e^{-\lambda\pi\frac{r^2}{4}} \tag{6.8}$$

By combining the expressions 6.4 and 6.8 with 6.3 we get:

$$\eta_{GG} = \frac{\int_0^R 2\pi\lambda r(1 - e^{-\lambda\pi\frac{r^2}{4}})dr}{\int_0^R 2\pi\lambda r dr} = \frac{-4 + 4e^{-\lambda\pi\frac{R^2}{4}} + R^2\pi\lambda}{R^2\pi\lambda} \tag{6.9}$$

Relative Neighborhood Graphs

As shown in Figure 2.5(a) the witness area W has the shape of a symmetrical lens defined as the intersection of two circles whose centers are at distance $r = \delta(A, B)$. Area of a lens is equal to the doubled circular segment area (Equation 2.2) with a distance to chord of $\frac{r}{2}$:

$$A_{\text{◊}}(r) = 2(r^2 \arccos\frac{r}{2r} - \frac{r}{4}\sqrt{4r^2 - r^2}) = \frac{2\pi r^2}{3} - \frac{r^2}{2}\sqrt{3} \tag{6.10}$$

knowing that $r \geq 0$ and $\arccos\frac{1}{2} = \frac{\pi}{3}$.

In order to get the ratio of removed edges in Relative Neighborhood Graphs, Equations 6.4 and 6.10 are substituted in 6.3:

$$\eta_{RNG} = \frac{\int_0^R 2\pi\lambda r(1 - e^{-\lambda(\frac{2\pi r^2}{3} - \frac{r^2}{2}\sqrt{3})})dr}{\int_0^R 2\pi\lambda r dr} = 1 + \frac{-6\pi(-1 + e^{-\frac{1}{6}\lambda R^2(3\sqrt{3} - 4\pi)})}{(3\sqrt{3} - 4\pi)R^2\pi\lambda}. \tag{6.11}$$

6.1.2. The Expected Number of Faces and the Expected Face Size

In order to calculate the average face size (number of edges belonging to a face), we start from Euler's Formula for number of faces in a planar graph:

$$f = 1 + c(G) - n + e \tag{6.12}$$

The expected number of edges in the connectivity graph is (we count all neighbors of each node in the graph, thus counting all edges in the graph twice):

$$\bar{e} = \frac{n \cdot d}{2} = \frac{\lambda |A| d}{2} \tag{6.13}$$

where the average degree of a node $d = \lambda R^2 \pi$. The ratio of eliminated edges η is defined in Equation 6.9 for Gabriel Graphs and in Equation 6.11 for Relative Neighborhood Graphs. The expected number of edges in the planarized graph is: $e = \bar{e}(1 - \eta)$.

Roach has proposed in [146] an approximation for the average number of graph components in a unit region of plane:

$$c(G)_{unit} = \frac{\lambda p_0 \ln p_0}{p_0 - 1} \tag{6.14}$$

where $p_0 = e^{-R^2 \pi \lambda}$ is the probability that a node is isolated. If graph is finite, the absolute number of components in it is:

$$c(G) = |A| \frac{\lambda p_0 \ln p_0}{p_0 - 1} + 1 \tag{6.15}$$

The Equations 6.14 and 6.15 are valid for non-planarized graphs. Since the planarization process to GG and RNG does not increase number of graph components, Equations 6.14 and 6.15 are applicable to them as well. Substituting 6.15 in 6.12 we get the expected number of faces in the planarized graph:

$$f = 1 + |A| \frac{\lambda p_0 \ln p_0}{p_0 - 1} + 1 - \lambda |A| + \bar{e}(1 - \eta) = 2 + |A| \lambda \left(\frac{p_0 \ln p_0}{p_0 - 1} + \frac{d}{2}(1 - \eta) - 1 \right). \tag{6.16}$$

The average face size can be approximately calculated as:

$$s = \frac{2 \cdot \bar{e}(1 - \eta)}{f}. \tag{6.17}$$

The expression 6.17 is approximate because it assumes that each edge in the planar graph belongs to two different faces: e.g., the edge PQ in Figure 6.1 belongs to faces $A1$ and $A2$. However a bridge belongs only to one face: e.g., the edge XY in Figure 6.1 belongs to face $A2$ only.

6.1.3. Simulation Results

The analytical expressions are compared with output of a topology simulator that places vertices in an area, creates the underlying connectivity graph and measures the ratio of removed edges and the average face size for Gabriel and Relative Neighborhood Graphs.

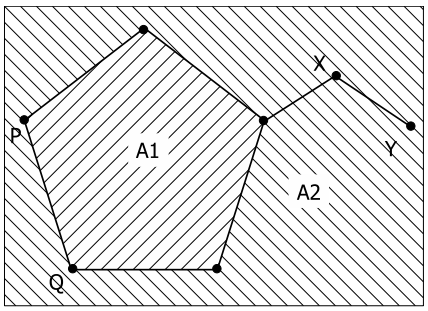
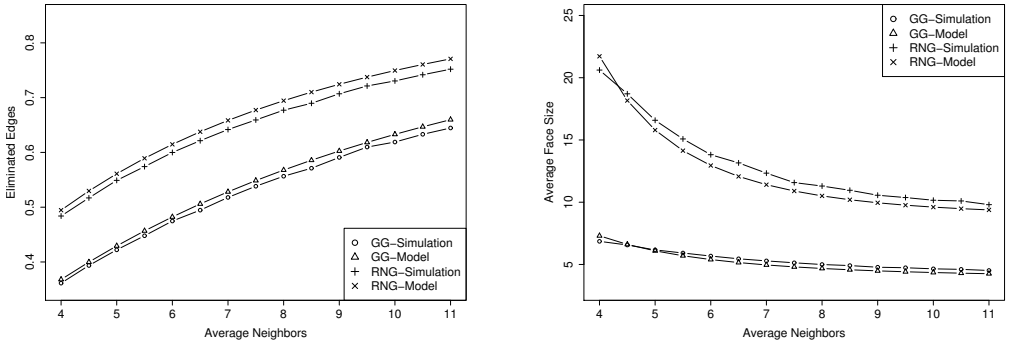


Figure 6.1.: Bridges in a planarized graph.



(a) Ratio of removed edges.

(b) Average face size.

Figure 6.2.: Comparison of graph planarization model and simulation results.

The simulations were run 1000 times per deployment configuration. In all placement scenarios, number of nodes is fixed to 275[2] and they are placed uniformly in a quadratic area. Communication radius is set to 250 units. Size of area is varied to allow different node and graph density (expressed in figures the expected number of neighbors d).

Instead of approximate value for average face size from Equation 6.17, in simulation is used the correct expression that differentiates bridges (b) and ordinary edges (e):

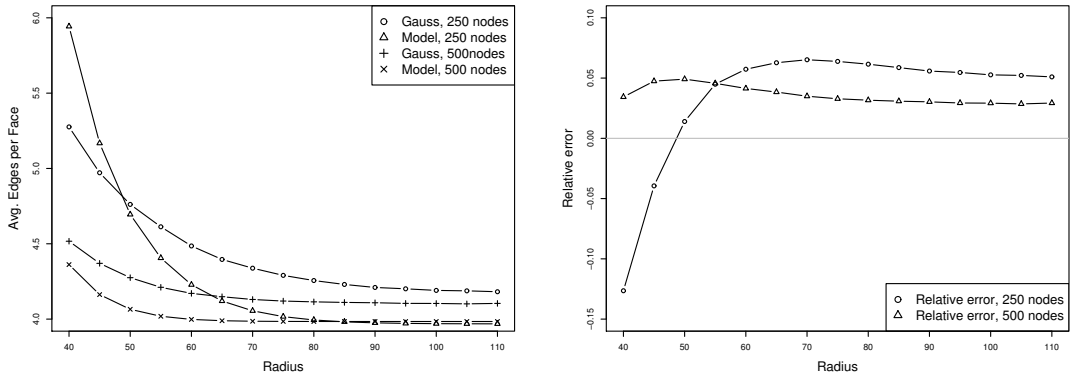
$$s = \frac{2 \cdot e - b}{f}$$

Figure 6.2(a) shows the ratio of removed edges in process of graph planarization for different node densities and different communication radii. The errors of analytical model are rather small – the maximal difference between it and the simulation is approximately 2.5%.

The approximation for the expected face size is acceptable although it ignores the existence of bridges, as it can be seen in Figure 6.2(b). The mathematical prediction follows the shape of the simulation curves, accurately representing the behavior of the average face size.

For a study of behavior of non-homogeneous vertex placement processes vertices are

[2]Average number of nodes present in main component of Berlin's WMN, as explained in Chapter 8.



(a) Average face size.

(b) Relative error of the model.

Figure 6.3.: Comparison of graph planarization model and simulation results for non-uniform node placement.

placed in accordance with normal distribution so that the highest density of nodes is at the central part of the placement area (500x500 units). Communication radius and node count are varied to change the graph density.

The Figure 6.3(a) shows the average face size in non-homogeneous placement process. As expected, the differences between the model and simulation are more pronounced than for the uniform node placement, in particular for very sparse graphs (less than four neighbors on the average). The maximal relative error of model is less than 15% and for most of studied deployment configurations it is close to 5% which is an excellent result taking into consideration that the model has not been developed for such node placement models.

6.1.4. Interpretation of the Results – the Average Face Size in Limit

In this section we determine the behavior of obtained expressions in limit and verify their conformance to results of Devroye [66]. In limit, the communication radius R is increased to infinity. The placement area is infinite as well as the number of nodes but their ratio is kept constant $\lambda = \frac{n}{|A|}$.

As the consequence of communication radius increase, grows the expected number of neighbors - $d = R^2\pi\lambda$ and the overall graph connectivity [36]. It can be safely assumed that the underlying graph is connected ($c = 1$ in Equation 6.12):

$$f = 2 - n + \frac{nR^2\pi\lambda}{2} \cdot (1 - \eta).$$

Division of both sides by $|A|$ and knowing that $n = \lambda|A|$ gives the expected face density $\mathscr{F}$ – the number of faces per surface measurement unit:

$$\mathscr{F} = \frac{f}{|A|} = \frac{2}{|A|} - \lambda + \frac{R^2\pi\lambda^2}{2} \cdot (1 - \eta)$$

Since the observed area A is infinite, the factor $\frac{2}{|A|}$ can be ignored:

$$\mathscr{F} = -\lambda \cdot (1 - \frac{R^2\pi\lambda}{2} \cdot (1 - \eta)) \tag{6.18}$$

Accordingly, based on expression 6.17 the expected face size in the limit is:

$$s^{\infty} = \lim_{R\to\infty} \frac{\lambda|A| \cdot R^2\pi\lambda(1-\eta)}{\mathscr{F}|A|} = \lim_{R\to\infty} \frac{R^2\pi\lambda^2(1-\eta)}{\mathscr{F}} \tag{6.19}$$

Substitution of Equations 6.9 and 6.11 in 6.18 gives the expected face densities for Gabriel and Relative Neighborhood Graphs:

$$\mathscr{F}^{\infty}_{GG} = \lim_{R\to\infty} -\lambda + \frac{R^2\pi\lambda^2}{2} \cdot (1 - \frac{-4 + 4e^{-\lambda\pi\frac{R^2}{4}} + R^2\pi\lambda}{R^2\pi\lambda}) = \lambda \tag{6.20}$$

and

$$\mathscr{F}^{\infty}_{RNG} = \lim_{R\to\infty} \frac{R^2\pi\lambda^2}{2} \cdot \frac{6\pi(-1 + e^{-\frac{1}{6}\lambda R^2(3\sqrt{3}-4\pi)})}{(3\sqrt{3}-4\pi)R^2\pi\lambda} - \lambda = \frac{\pi - 3\sqrt{3}}{3\sqrt{3}-4\pi} \cdot \lambda \approx 0.278\lambda \tag{6.21}$$

Using these expressions in 6.19 and applying once more the Equations 6.9 and 6.11 we get the expected faces' size for Gabriel and Relative Neighborhood Graphs:

$$s^{\infty}_{GG} = \lim_{R\to\infty} \frac{\lambda \cdot R^2\pi\lambda(1 - \frac{-4+4e^{-\lambda\pi\frac{R^2}{4}}+R^2\pi\lambda}{R^2\pi\lambda})}{\lambda} = 4 \tag{6.22}$$

and

$$s^{\infty}_{RNG} = \lim_{R\to\infty} \frac{\lambda \cdot R^2\pi\lambda \frac{6\pi(-1+e^{-\frac{1}{6}\lambda R^2(3\sqrt{3}-4\pi)})}{(3\sqrt{3}-4\pi)R^2\pi\lambda}}{\frac{\pi-3\sqrt{3}}{3\sqrt{3}-4\pi} \cdot \lambda} = \frac{-6\pi}{\pi - 3\sqrt{3}} \approx 9.1745 \tag{6.23}$$

Our results in limit are identical to results from [66] – there exist only minor differences in notation. The added value of our approach is that it provides the model of Gabriel and Relative Neighborhood Graphs' properties not only in limit but for the large set of values.

6.2. Estimation of the Shortest Cycle Size

The average face size in Gabriel graphs in limit converges to four for large complete graphs (Equation 6.22). It is obvious that for complete graphs, the average shortest cycle size is three since every pair of nodes shares a neighbor. Thus, the improvement of the upper bound of the average shortest face size is possible and in this section it is further reduced.

Probability that a node A has a neighbor B on distance r is given by Equation 6.4. Such pair of nodes AB form a 3-cycle[3] if they have a common neighbor, i.e., if there exists a node at a distance smaller than R from both of nodes. Geometrically, such

[3]Cycle consisting of three edges, also called a triangle.

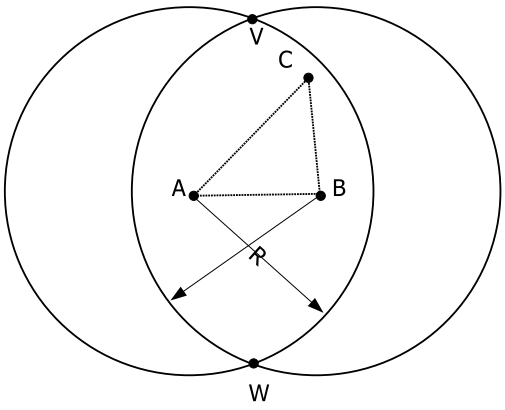


Figure 6.4.: Cycle consisting of three edges in a random geometric graph.

area is defined as intersection of two circles with radii R whose centers are at distance r (Figure 6.4). The probability that there exists at least one node in the area is $1-e^{-\lambda|\lozenge_{AB}|}$.

Thus, the probability P_{3c} of 3-cycle existence is:

$$P_{3c} = \frac{1}{\lambda\pi R^2}\int_0^R 2\lambda\pi r(1 - e^{-\lambda(2(R-\frac{r}{2})^2\cdot arccos\frac{R-\frac{r}{2}}{2R} - \frac{R-\frac{r}{2}}{2}\sqrt{4R^2-(R-\frac{r}{2})^2)})dr \tag{6.24}$$

The integral cannot be analytically solved and has to be numerically calculated.

It is now possible to provide an upper estimate of the average shortest cycle size: If P_{3c} edges form 3-cycles, remaining $(1 - P_{3c})\frac{nd}{2}$ edges are in cycles longer than three. The largest possible average cycle size is obtained if all edges that do not belong to 3-cycles form one large cycle, consisting of $(1 - P_{3c})\frac{nd}{2} + 1$ edges. It follows that the average cycle size cannot exceed the value of $\bar{C}_{3-max}$:

$$\bar{C}_{3-max}(n, d, P_{3c}) = \frac{\frac{3nd\cdot P_{3c}}{2} + ((1 - P_{3c})\bar{e} + 1)\cdot(1 - P_{3c})\bar{e}}{\bar{e}} \approx 3P_{3c} + (1 - P_{3c})^2 \tag{6.25}$$

For brevity, instead of repeating the expression for expected number of edges $\bar{e} = \frac{nd}{2}$ only its symbolic name is used. For large n or d, the additive factor of one can be ignored and the approximating equation is obtained. As expected, with increase in graph's density, average cycle size converges toward three.

Although strict, this upper bound is unlikely to be met, in particular in sparse graphs where the effect of assumed longest-possible cycle is strongest (in a sparse graph, fewer edges belong to 3-cycles than in a dense graph). Such a long cycle is a complex structure, and probability of its formation is very low. Also, a considerable number of edges in sparse graphs are bridges that do not participate in cycle formation.

Unfortunately, we cannot assess the probability of a maximal cycle nor bridges occurrence, so this remains only an intuitive interpretation of Equation 6.25). However, it is possible to account for possible components in the graph and improve Equation 6.25. The components are common in sparse graphs and awareness of their existence should improve the calculated bounds.

In context of estimation of the average length of the shortest cycle, graph components can be treated in two possible ways:

Network is divided into the set of single node components and one connected component, containing all remaining nodes. The single-node components do not have any edges attached to them. Their effect is the reduction of the node count in the connected component $n' = n - c + 1$ (their average number c is calculated by Equation 6.15) and consequently of the number of edges in it $e' = \frac{n'd}{2}$. Now we can substitute new values of n' and e' in Equation 6.25 (the probability P_{3c} does not depend on number of nodes in network/component, so it remains the same as before.)

$$\bar{C}^{1}_{3-max}(n, d, P_{3c}) = \bar{C}_{3-max}(n - c + 1, d, P_{3c}) \tag{6.26}$$

This approach does not compromise the strict bounds used in derivation of Equation 6.25. It includes the effects of disconnected network but it still creates the maximal possible cycle in the connected component of the graph since it does not take edges from the connected component.

Network is divided into c equally sized components. Each of c components has $n'' = \frac{n}{c}$ nodes and $\bar{e''} = \frac{n''d}{2}$. The average shortest cycle size in calculated similarly as in with Equation 6.25:

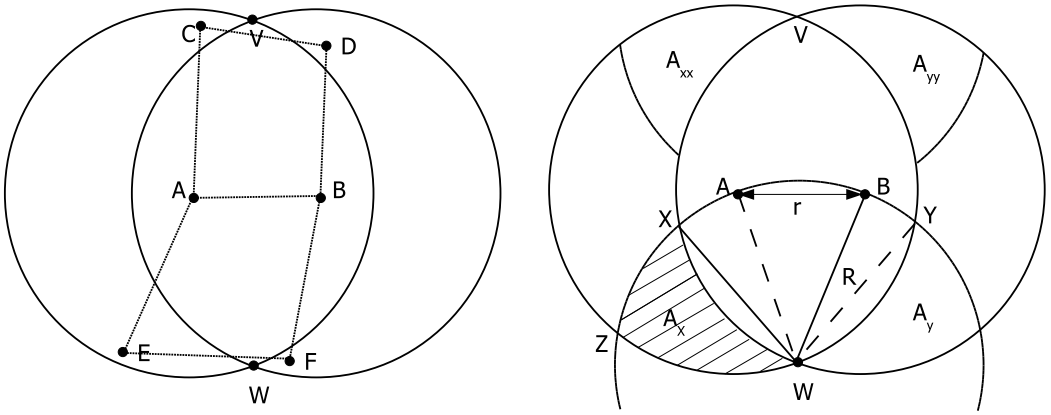
$$\bar{C}^{\frac{n}{c}}_{3-max} = \frac{c(\frac{3n''d \cdot P_{3c}}{2} + ((1 - P_{3c})\bar{e''} + 1) \cdot (1 - P_{3c})\bar{e''})}{c \cdot \bar{e''}} = \bar{C}_{3-max}(\frac{n}{c}, d, P_{3c}) \tag{6.27}$$

The numerator of the fraction in Equation 6.27 sums the lengths of shortest cycle sizes same as in Equation 6.25 over all components (since they are identical, the numerator is simply multiplied by c). The denominator is the total number of edges in the graph - $c \cdot e$". The main difference between 6.26 and 6.27 is that the later calculates the maximal possible cycle in components that are much smaller than the original graph. Smaller components mean smaller maximal possible cycle, which in turn reduces the estimate of the average.

Because of that, estimate from Equation 6.27 provides very low estimate of the average length of shortest cycles in graph. However, its assumption that network is divided in equal components is also its biggest weakness. The assumption of such component composition minimizes the estimate of the average length of shortest cycles, but it also breaks correctness of the bound – if composition of components of a graph is different, the metric will underestimate the average. Thus, it cannot be used as a strict bound, but as an approximate estimate and its value must be interpreted with care.

The upper bound of the average shortest cycle size can be further reduced by calculating probabilities of occurrence of 4-cycles, 5-cycles, etc. Calculating probability of their existence is extremely difficult as node location interplay and determining of the integration limits gets increasingly complicated with each added node, similar as for the connectivity analysis from [136] and [142]. Figure 6.5 demonstrates the existence of two valid configurations in which a 4-cycle exist over an edge: the edge AB belongs to the cycles ABDC and ABFE.

In Figure 6.5(b), A_x marks the largest possible area where a node C can be placed so that it is neighbor with node A, but not a neighbor of node B, and that it is still possible to place another node D within area A_y so that it is neighbor of node B but not a neighbor of node A.



(a) 4-cycle example

(b) Possible location of a node if it belongs to a 4-cycle.

Figure 6.5.: Cycles consisting of four edges in a random geometric graph.

The size of area A_x is calculated by subtraction of the non-shaded part in Figure 6.5(b) from area of lens $\langle\rangle_{ZY}$. The unshaded area consists of two circular segments XW and YW and two overlapping circular sectors XWB and AWY (we subtract them both from lens $\langle\rangle_{ZY}$ and add to it the overlapping area of sector AWB, in accordance with inclusion-exclusion principle):

$$A_x(r) = \langle\rangle_{ZY} - ((\bigtriangledown_{XWB} + \bigtriangledown_{AWY} - \bigtriangledown_{AWB}) + \frown_{XW} + \frown_{YW}) =$$

$$\frac{2R^2\pi}{3} - \frac{\sqrt{3}R^2}{2} - (2 \cdot \frac{R^2\pi}{6} + 2(\frac{\pi}{6} - \frac{\sqrt{3}}{4})R^2 - R^2 arcsin\frac{r}{2R}) = R^2 arcsin\frac{r}{2R} \quad (6.28)$$

(the definitions of lens, circular sector and circular segment are on page 25).

The precise approach would require integration of probability of existence of node C over area A_x and of existence of a node D within area A_y so that distance between C and D is less than R. Such process is highly complex and we employ approximation instead of it – it is assumed that nodes in whole area A_x may communicate with nodes that are placed anywhere in area A_y. So, P^1_{4c} approximates probability of 4-cycle existence:

$$P^1_{4c} \approx \frac{1}{\lambda\pi R^2} \int_0^R 2\lambda\pi r(1 - e^{-\lambda R^2 arcsin\frac{r}{2R}})dr \quad (6.29)$$

The integral in Equation 6.29 is analytically solvable and it is resolved to a complex polynomial, consisting of more that thirty terms. For brevity, it is not shown here, but it can be easily obtained in any analytical solver, such as Mathematica.

This probability accounts only for one possible 4-cycle around point W, but one more may be formed in proximity of point V, with the same probability. For our analysis, it is not important which cycle is formed, but whether there exists at least one. The probability that at least one 4-cycle is formed, in proximity of a pair of neighbor nodes

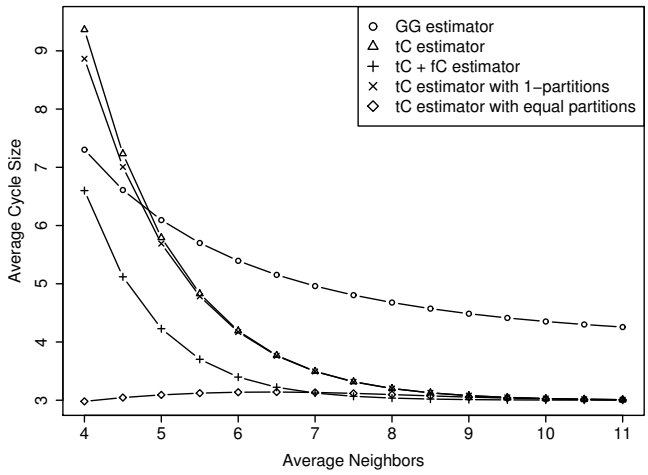


Figure 6.6.: Estimators of the average size of shortest cycles.

is obtained as sum of probabilities that at least one 4-cycle exists:

$$P_{4c} = 2P_{4c}^1(1 - P_{4c}^1) + (P_{4c}^1)^2 = P_{4c}^1(2 - P_{4c}^1) \tag{6.30}$$

A 4-cycle is shortest cycle over an edge if there exists at least a single 4-cycle and there does not exist a 3-cycle on that edge:

$$P_{4c-shortest} = P_{4c}(1 - P_{3c}) \tag{6.31}$$

If approximate probability of occurrence of 4-cycles is included, the upper bound on maximal average shortest cycle size is stricter and converges faster to its limit value. However, in contrary to the Equation 6.25, P_{4c} is not the strict upper limit because of the approximation that is made during calculation of P_{4c}^1. Due to reasons presented in the introduction of this chapter, it is still of considerable practical value and it can be calculated as:

$$\bar{C}_{4-max} = \frac{3P_{3c}\bar{e} + 4P_{4c}(1 - P_{3c})\bar{e} + ((1 - P_{3c} - P_{4c}(1 - P_{3c}))\bar{e})^2}{\bar{e}} \tag{6.32}$$

Since graphs of interest are of considerable size and P_{4c} is already an approximation, the additive factor of one in expression for the largest-possible cycle formed of edges that do not belong to 3- and 4-cycles is ignored.

Figure 6.6 shows the modeled bounds[4]. Estimator 6.25 provides the highest values for sparse graphs. Its improved version, estimator 6.26 provides slightly smaller estimate but the average face size of Gabriel graph provides smaller upper bound and it is better for sparse graphs.

As the density of a graph increases, estimator 6.25 prevails over GG estimator and converges fast toward expected value of three. The change occurs approximately once

[4]tC is the $\bar{C}_{3-max}$ estimator from Equation 6.25, tC+fC is $\bar{C}_{4-max}$ estimator from Equation 6.32, and they are followed by $\bar{C}_{3-max}^1$ and $\bar{C}_{3-max}^{\frac{n}{c}}$ estimators

the average node degree reaches five. It is remarkable that already for graphs with average vertex degree of ten, shortest cycle size is negligibly different from three (such simulation setups are common in the literature as it will be shown in Table 8.1, page 140).

The estimator from Equation 6.32 is tighter than both Gabriel and 3-cycle estimators (Equations 6.25 and 6.26) but as already explained, its predictions must be taken with reserve since it needs not provide the guaranteed bound.

Of all estimators, the estimator in Equation 6.27 gives the smallest values, very close to three, but it is highly approximate as already explained. For extremely sparse graphs ($d<2$) its value even falls below three which is obviously wrong. If the problem of calculation of component size distribution in RGGs (its issues are explained in Appendix A) is resolved, the estimator has potential to be the best of all presented on the whole range of possible node degrees.

6.3. Summary

This chapter has studied the locality properties in RGGs. It has been shown that if a cycle exists in the connectivity graph, it tends to be short.

The analytical results for estimation of the average shortest cycle size have been obtained, using different approximations. All approximations show that the shortest cycle size converges to three already in networks of moderate density. Even in rather sparse networks, the average shortest cycle size will not be over eight.

This information can be used for further reduction of communication overhead in DIBADAWN: instead of full searches that cover whole network, it may be possible to employ localized searches that cover only the k-neighborhood of a node. The performed shortest cycle-size analysis indicates that bridge detection should be accurate even with a small local search radius, but the effects of such partial searches on accuracy of the articulation point detection are unclear and have to be evaluated.

The detailed evaluation of localized searches in DIBADAWN will be performed in Chapter 9. Evaluation is necessary since the RGG model which is used in this chapter introduces considerable simplifications: real networks are not uniformly distributed, the model does not include stochastic behavior of wireless channel, HCA cycles that are discovered by DIBADAWN may not be equally sized/shaped as the shortest cycles.

The applicability of results of this section is not limited only to DIBADAWN, its analysis and proposed changes in its functionality. They also enable better understanding of existing protocols, and easier estimation of performance of new protocols prior to their implementation. For instance:

- Gabriel and Relative Neighborhood Graphs are used in various WMN protocols. The obtained equations on the ratio of removed edges in planarization process and the average face size simplify the theoretical analysis of functionality and performance of such protocols. For instance, Karp and Kung [98] and Bose et al. [44] propose combination of greedy geographical and perimeter routing for guaranteed delivery of packets in a location aware network. A packet is greedily forwarded towards its destination (with regard to its geographical location) until it reaches the destination, or a local minimum of the distance function. To avoid local minimum, the packet is forwarded over the edges of a face of Gabriel Graph

(perimeter routing). It has been observed that the length of path selected by the protocol is close to the optimal (Figure 11 in [98]). The developed face-size model easily explains this behavior: dense networks in simulations in [98] create small faces so the variations introduced by perimeter routing are almost negligible.

- The probabilistic three- and four-cycle analysis in Section 6.2 explains good performance of local route repair protocols [176]. The presented model predicts high probability of having a short alternative path between a pair of nodes and consequently the high success rate of local route repair if a node in a network has more than seven neighbors on the average (as it was defined in simulation scenarios in [176]).

7. Case Study: Measurements from Community Wireless Multi-hop Networks

In this chapter are compared topological properties of community wireless multi-hop networks in Berlin (Berliner Freifunk Netzwerk [7]) and Leipzig (Freifunk Leipzig, [8]) with common theoretical and simulation node placement and mobility models such as uniform, grid and the random waypoint model.

Four topology metrics have been selected for comparison in this chapter: degree distribution, bridge and articulation point count and relative component size after bridge removal. These topological metrics directly influence properties of protocols that are simulated.

The node degree distribution is correlated to the congestion on the wireless channel and probability of packet loss.

Bridges are only communication links between potentially large, 2- (or better) connected network components. If these components are of a considerable size, bridges that connect them tend to get congested, reducing the available throughput per flow and increasing packet latency, thus reducing quality of services deployed in network. In addition to bridge count it is important to capture the size of components that bridges connect.

Articulation points are gateways between different network components. The network gets disconnected if an articulation node is turned off, node's software fails or its energy source is depleted. The articulation points tend to route more traffic than other nodes in the network so they are more prone to energy exhaustion and critical failure that partitions the network. Due to limited operating memory and processing power of wireless nodes, packet buffers at articulation points are more prone to overloading, introducing additional packet losses.

The analysis of these four topological characteristics shows that properties observed in reality are different than in common theoretic models. For this thesis it is of particular importance that articulation points and bridges are much more frequent in reality than in the known theoretical models. The effects imposed by them on a real network and protocols deployed in it are more accentuated than in studies based on purely theoretical models. This fact provides additional motivation for development and application of DIBADAWN.

7.1. Data Sampling and Simulation Methodology

This section explains the methodology used for data sampling from real networks and simulation of artificial topologies.

Scenario	Samples	Avg. Nodes	Min-Max Nodes	Avg. Edges	Min-Max Edges	Area	R	α	σ
Berlin	1465	315.29	199-419	633.79	291 - 951	-	-	-	-
Leipzig	1589	586.66	452-615	1277.91	1006-1396	-	-	-	-
Uniform	1500	400	400-400	1061	945-1188	1000 x 1000	67	4	0
Uniform (S)	1500	400	400-400	1063.33	958-1190	1000 x 1000	40	4	7
RWM	1500	400	400-400	1524.48	1337-1799	1000 x 1000	40	4	7
Grid	-	400	400-400	2560	2560-2560	684 x 684	40	4	7

Table 7.1.: Network characteristics and simulation parameters used for comparison of topological properties.

For data sampling, we were limited by the existing, built-in capabilities of Berlin and Leipzig networks. We had no control over the networks and no knowledge what transpires in the network at the sampling moment: some users might be experimenting with protocol parameters, testing new equipment, or deploying new and possibly incorrect protocol versions. The implication of this uncertainty is that a single measurement sample cannot be trusted. Instead, large series of measurements are needed to reason about topological properties. The same, stochastic methodology is applied to simulation to eliminate effects of outliers in individual simulation executions.

Table 7.1 shows important parameters of measurements and simulation. Number of samples in all cases was approximately 1500. In real networks number of participating nodes varies over time as shown in column with minimal and maximal number of nodes encountered over all samples. All simulation scenarios had 400 nodes. Characteristics of wireless channel σ and α and their meaning have been explained in Section 2.1.3.

7.1.1. Data Sampling Methodology

Data collection in real networks does not allow high intrusion level that would be ideal for sampling purposes: users are unwilling to modify the software running on nodes so that it collects and sends the captured data to a repository. The data had to be extracted from running protocols, as they were.

Networks in Berlin and Leipzig used an extended variant of the Optimized Link State Routing (OLSR) routing protocol [14]. Due to several issues of OLSR in urban environment, discussed on networks' websites ([7], [8]), the protocol used in networks does not comply with the OLSR standard defined in RFC 3626:

- Multipoint relays are not used. Instead, each node is disseminating its local topology knowledge, using plain network flooding.
- In order to reduce the overhead produced by dissemination of TC (topology control) packets, fisheye algorithm is applied for information dissemination: each TC packet has time to live (TTL) field which specifies how many hops a TC packet should be forwarded. Nodes send TC packets every 0.5 to 2 seconds (configurable parameter of the protocol) setting the following values in TTL field: 255, 3, 2, 1, 2, 1, 1, 3, 2, 1, 2, 1, 1.
- In order to improve utilization of network resources, ETX metric (Definition 2.1) is used for packet routing [64] – packets are not routed by minimizing the hop count, but by minimizing the ETX.

```
Table: Topology
Destination IP Last hop IP  LQ   ILQ  ETX
10.14.1.125    10.14.0.200  0.44 0.84 2.71
10.14.1.43     10.14.0.137  0.90 0.96 1.16
10.14.0.189    10.14.0.225  0.00 0.59 0.00
```

Figure 7.1.: A section of topology sample taken at an OLSR node.

- The topology control (TC) packages disseminate the link quality data (ETX), not just the topology information.

The described proactivity of OLSR version 0.4.10 and its differences to OLSR standard were beneficial for sampling purposes. In the meantime, the protocol has been officially differentiated from OLSR and is known as Better Approach to Mobile Ad-Hoc Networking - B.A.T.M.A.N. [5].

TC packets are also used for link quality estimation. A node knows that it should receive a TC packet from a neighbor every 0.5s, with some small jitter. At each node A there exists a sliding window mechanism that counts how many packets should have been sent by node B and how many actually reached node A. Based on these values, node A estimates the link quality w_{BA} as $\frac{\#received}{\#expected}$. Neighbor B estimates the link quality w_{AB} using the same approach. TC packets include link quality information and through their exchange both nodes obtain two directional information on quality of the link that connects them. Although this method has some drawbacks (e.g., it does not capture burstiness of losses, it is unaware of reasons of packet losses) it provides acceptable estimation of link quality and improves network throughput [64].

Frequent topology updates and static nodes allow us to take samples from a single node. The samples are taken every ten minutes in Berlin and every fifteen minutes in Leipzig network. The extracted samples include topology and ETX data. Successive samples differ from one another since nodes are joining and leaving network, and network traffic is changing which in turn generates different interference patterns and different link quality data. Additional changes in link quality are created by interference with network unrelated wireless access points and environment changes.

In Berlin, a node was installed in the network to collect the data. Taking of topology samples from a node running the OLSR daemon, version 0.4.10 is rather simple, since the protocol daemon can output the topology table to a textual file. Figure 7.1 shows a section of the topology sample.

The later analysis of topology samples showed some inconsistencies: name NLQ (neighbor link quality) in documentation is written as ILQ in the output; if a link is not operational but it has not been deleted from the topology table yet, instead of marking ETX as infinite, ETX is set to zero. Since it is impossible to have ETX value less than one (that would mean that for one sent packet more than one are received), it indicated existence of an error in the implementation of the routing protocol. Through inspection of the code it was determined that the "error" (or just an undocumented convention) was in the output routine which wrote 0.00 instead of infinity.

We were unable to install a node that directly participates in the network in Leipzig. The data available from web-site of the network is used instead. The network in Leipzig

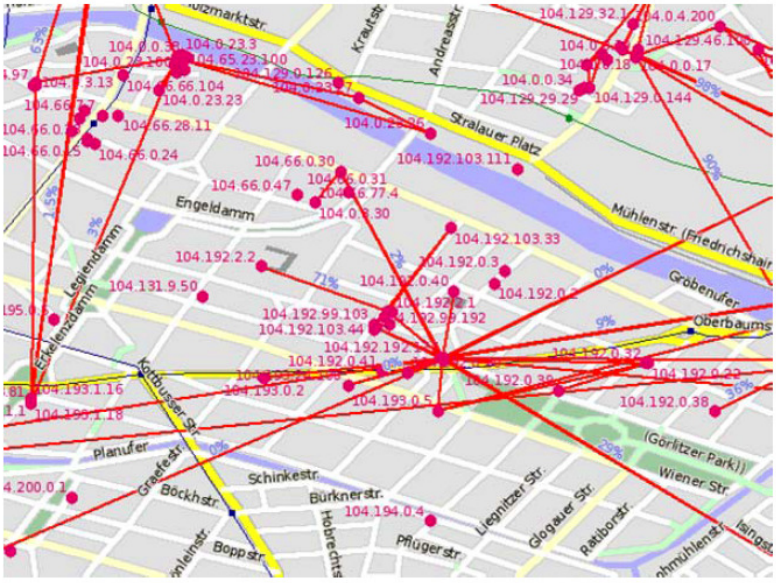


Figure 7.2.: Part of Berlin's network.

has several components. Since it is used for Internet access, each of the components can forward the topology data to the central server that builds the joint topology. The topology data available at the server is in .dot format [75] and each link is weighted with the ETX value. The topology file includes the non-existing links from network components to the central server as well. Such links do not have ETX values associated with them. They are recognized and ignored in the analysis.

The measurement has captured lots of important aspects that are not supported in popular simulators [70] [19] [106]. The Figure 7.2[1] shows a small portion of the network in Berlin. It can be seen that between nodes are buildings, streets, parks, even a river, creating a very complex propagation environment. Nodes are placed in buildings of different heights, some of them are on roofs and some are inside apartments. Small part of network is planned and uses custom-built, high gain, carefully directed antennas that provide high quality links over large distances (several kilometers) while other nodes are end-user (their location is determined by users), using off-the-shell components. All this influences the signal propagation and resulting network topology as it can be seen in the Figure 7.2 where some of the links are very short, while some reach over long distances.

The topological and link quality data were parsed and placed in a relational database [13]. The data processing (e.g., calculating which links in the network are bridges) was performed in a custom Java application. Statistical processing has been performed in R-Environment [143].

[1]The Figure does not show all the nodes and links in the area since coordinates of approximately $\frac{1}{3}$ of nodes in Berlin's network are unknown and cannot be shown on the map. Due to it, it seems that network is substantially sparser than it really is.

7.1.2. Validity of Measurements

It must be noted that the presented analysis is not perfect. For instance, the existence of "tunnels"[2] in the network is confirmed. They can establish connections between otherwise disconnected network components and alter perceived topology. All such systematic errors which have been recognized have been removed, but it is not possible to guarantee that all of them were covered since networks are community based and we had no control over tunnels.

Additionally, the results of Chapter 4 send a clear message that the link existence detection is not a trivial task. The HLD detectors have a considerable probability of errors in link detection, in particular if they are not configured properly. As it is predicted by the developed HLD error models, the developers in Freifunk community experienced numerous issues with the hysteresis mechanism[3] for local link detection because of the numerous false positives and negatives.

The OLSR version deployed in Freifunk networks does not use hysteresis-based neighbor detection as in the original OLSR. Instead of HLDs, a heartbeat mechanism which estimates link quality is used.

The link quality estimator $LQE(l)$ operates on sequence of l heartbeats. $LQE(l)$ uses the sliding window for heartbeat management: the oldest entry in the window is replaced with the newest outcome of heartbeat transmission (success or omission). It receives r heartbeats and observes o omissions during this sequence ($r + o = l$) and estimates the link quality $p_{est} = r/l$. The hypothesis of link existence is accepted if estimated link quality is greater or equal than the link acceptance threshold t and rejected if it is smaller than the threshold.

$LQE(l)$ is applied to a link of quality p, $p \in [0, 1]$. The estimation of link quality asymptotically approaches the quality p if the observation sequence has an infinite length. Within a sequence of finite length l, stochastic behavior of communication channel causes the variations in estimation and deviation from the correct values. Additional errors are introduced by $LQE(l)$ since for finite l, it can assign link quality only in steps of fixed size $\frac{1}{l}$.

In order to evaluate probability of errors in link detection process in community networks of Berlin and Leipzig, the overall evaluation methodology of HLDs from Chapter 4 is utilized. According to the methodology, the first step is to calculate the probability of link acceptance $P_1(l, t, p)$ and rejection $P_0(l, t, p)$.

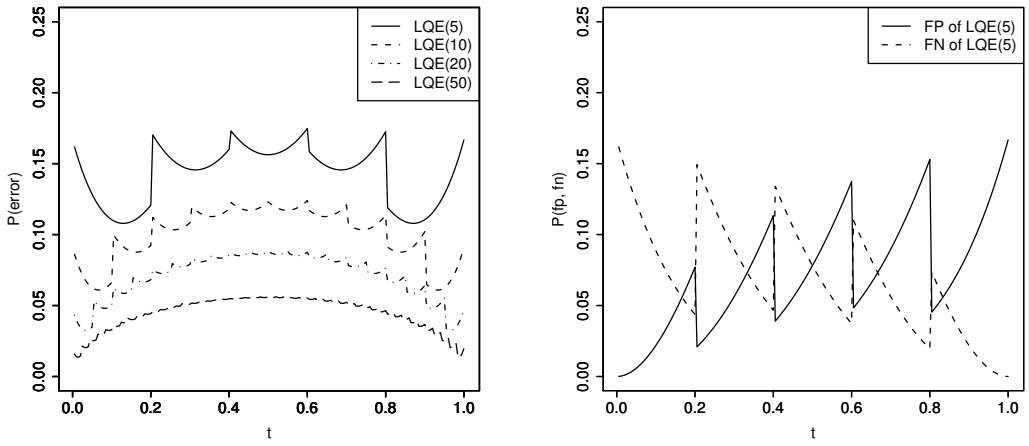
It can be safely assumed that heartbeat transmissions are mutually independent. The period between successive heartbeats is measured in seconds so it is considerably longer than the correlated changes of signal strength of communication channel with fading (measured in tens of milliseconds).

Thus, the probability of receiving r heartbeats on a sequence of l attempts can be modeled by the binomial distribution:

$$P_{LQE}(r, l, p) = \binom{l}{r} p^r (1 - p)^{l-r} \tag{7.1}$$

[2] wired communication channels that forward wireless traffic

[3] Discussed in user and developer forums, such as https://lists.openmesh.net/pipermail/b.a.t.m.a.n/2008-June/001999.html



(a) Probability of error.

(b) Probability of false positives and false negatives for $LQE(5)$.

Figure 7.3.: Behavior of $LQE(l)$ link detectors for the uniform distribution of link quality.

The link is accepted if at least $\lceil t * l \rceil$ heartbeats are received. So the probability of declaring the link with quality p as active is:

$$P_1(l,t,p) = \sum_{i=\lceil t*l \rceil}^{l} \binom{l}{i} p^i (1-p)^{l-i} \tag{7.2}$$

and

$$P_0(l,t,p) = 1 - P_1(l,t) \tag{7.3}$$

The probability of an erroneous link estimation is the sum of probabilities of a false positive and of a false negative (they are mutually exclusive, same as in Equation 4.1, page 47):

$$P_E(l,t,p) = \int_0^t P_1(l,t,p) f_p(p) dp + \int_t^1 P_0(l,t,p) f_p(p) dp \tag{7.4}$$

The probabilities of errors, false positives and false negatives can be seen in Figure 7.3 for the uniform distribution of link quality. The effects of step-wise increase in link quality estimation of LQE detectors are clearly visible and, as expected, more pronounced for shorter observation sequences.

The error probability has a distinguishable shape, with series of local minima that are located within intervals $[\frac{i}{l}, \frac{i+1}{l}], i \in \{0..(l-1)\}$ and local maxima at $\frac{i}{l}, i \in \{0..l\}$.

The reasons of such peculiar shape of the error probability are clearer if we observe the components of the error probability in Figure 7.3(b). As a threshold t is approaching from left (grows toward) the points $\frac{i}{l}$, the probability of false positives increases: it is more difficult for $LQE(l)$ to distinguish whether a link is above or under the threshold.

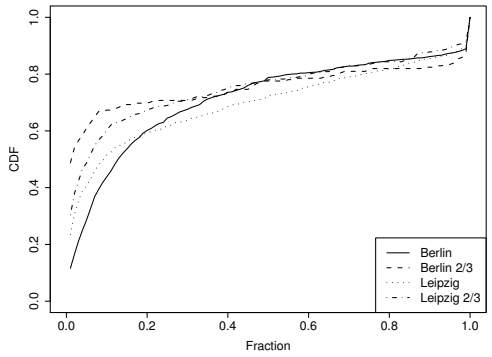


Figure 7.4.: Probability that an link in a network is repeatedly a bridge.

For example, if we observe $LQE(5)$ and the threshold t is set to 0.39, in order to correctly reject links with quality below 0.39, $LQE(5)$ should have less than two heartbeat receptions for all links in network that are in interval $[0, 0.39)$. The probability of having two or more receptions out of five attempts is considerable for links just below the threshold, which results in increased number of false positives. If threshold is set just a little bit higher, for instance to 0.41, the probability of false positives drops significantly: in order to create false positives, links with quality below 0.41 need to observe event "more than three heartbeat receptions out of five" which is an event with lower probability than the event "at least two receptions out of five" on interval $[0, 0.39)$. If the threshold is increased toward 0.6 the probability of false positives increases, due to the same reasons explained for threshold of 0.39.

Probability of a false negative follows the reversed logic: if threshold is just below step $\frac{i}{l}$ of a $LQE(l)$, in order to produce a false negative $LQE(l)$ has to observe less than i heartbeats for all links above the threshold which is not very likely (in example of $LQE(5)$, links with quality over 0.39 have to observe zero or one heartbeat receptions). As threshold is reduced, probability of a false negative increases: it is more probable that links just above threshold of 0.21 experience zero or one heartbeat reception.

In Freifunk networks, the local link timeout value is set to 20 seconds, during which one heartbeat message per second is exchanged (default configuration parameters), so it uses the $LQE(20)$. By Equation 7.4 $LQE(20)$ has the error probability of 0.049 for threshold set to 0.1. Compared with the error probability of $LQE(20)$, the error probability of OLSR and its combination of $HLD(2, 2)$ and $HLD(3, 2)$ is 0.401 and 0.307 respectively, for the same threshold. The error probability of $LQE(20)$ link detection strategy in static networks is acceptable for performed measurements. Additionally, we believe that the measurements can be trusted because:

- Timeout values for link removal from global topology tables in Freifunk version of OLSR are considerably larger than in the default OLSR. It can be said that Freifunk version of OLSR is even prone to create false negatives in context of bridge and articulation point detection since links in global topology table are held for 30 seconds. For comparison, the default OLSR removes links after only six seconds. It was possible to choose such conservative link removal times in Freifunk version of the OLSR since the operating conditions of this version were

		t=0.1	t=0.2	t=0.3	t=0.4	t=0.5	t=0.6	t=0.7
HLD(a,r), $a, r \in \{1..5\}$	(a,r)	(1,5)	**(1,5)**	(2,5)	(4,5)	(5,5)	(5,4)	(5,2)
	P_E	0.11	0.12	0.11	0.1	0.08	0.1	0.11
LQE(5)	P_E	0.11	0.12	0.146	0.16	0.156	0.175	0.146
HLD(a,r), $a, r \in \{1..10\}$	(a,r)	(1,10)	(2,10)	(4,10)	(6,10)	(10,10)	(10,6)	(10,4)
	P_E	0.06	0.07	0.06	0.05	0.04	0.05	0.06
LQE(10)	P_E	0.066	0.092	0.108	0.118	0.12	0.124	0.121

Table 7.2.: Comparison of optimal HLDs and LQEs for same length of observed sequences. Link quality $f_p(p)$ is uniformly distributed.

known (a static mesh network). The default parameters of OLSR [59] are selected so that it can be used in mobile networks.

- Analysis of repeatability of bridge existence shows that certain links have clear preference to be a bridge, while others rarely belong to the bridge set. The Figure 7.4 shows the cumulative distribution function of event E_b "a link is observed in a topology and it is a bridge". It can be seen that a subset of links is permanently in the bridge set (sharp raise at the end of the distribution). A large subset of links rarely belongs to the bridge set (sharp raise at the start of the distribution) which can be attributed to transient changes in network topology: as some of nodes move, fail or rejuvenate they may alter the status of a small subset of links in the network. Such behavior is even more pronounced if we observe only links that occur in at least two thirds of sampled topologies. In presence of a considerable systematic error caused by improper data sampling, distribution of the event E_b would be more uniformly distributed (consequently having a more linear cumulative distribution function).

Note on LQE and HLD link detectors:
Comparison of the optimal HLD detectors in static networks and their error probabilities from Tables 4.4 and 4.5 with the behavior of LQE detectors on the same length of observation sequence (i.e., $HLD(a,r), a, r \in \{1..l\}$ is compared with $LQE(l)$) reveals a surprising and partially counterintuitive property: HLDs which do not measure link quality have lower error probability than LQEs which explicitly measure link quality. This is explained by two facts:

- A HLD provides less information than a LQE and it is less flexible than a LQE – in order to obtain low error probability in link detection, HLD has to be configured especially for a single threshold. LQE delivers its estimation on link quality which is then interpreted by other participants in communication. This is particularly important if multiple applications are operated in a network, each with a different demand on acceptable link quality.

- The HLDs presented in Table 7.2 are optimized for a given threshold. If network application needs a different threshold value, the same HLD may produce error probabilities for order of magnitude higher (e.g., example of $HLD(1,8)$ at page 55) which is not the case for LQEs. Changes in error probability of LQEs exist, but the variation is considerably smaller as it can be seen in Figure 7.3(a) – even for highly variable $LQE(5)$ they are confined in interval $(0.11, 0.175)$.

7.1.3. Evaluation Methodology of Artificial Topologies

A custom-built application is used for analysis of artificial node placement algorithms, specialized for topology creation and analysis. The following node placement and mobility scenarios are compared to the measurements:

- **Uniform**: nodes are placed uniformly in the area. The path-loss propagation model is used.
- **Uniform (S)**: nodes are placed uniformly but there exists shadowing on the channel.
- **RWM**: nodes are initially uniformly placed and then they move in accordance with the RWM. Minimum speed is 0.5 m/s, maximum is 10 m/s, pause time is zero. Nodes are initially uniformly placed and then they move in accordance with the RWM. Topology snapshot is taken after 2000s of movement which is enough to reach the steady state node distribution [39].
- **Grid**: nodes are placed in a quadratically shaped grid, 20 by 20 nodes. Internode distance is slightly smaller (36m) than the nominal communication radius (40m). Because of the shadowing and the large internode distance, usable links exist only with direct neighbors in the grid. Due to simplicity of its regular shape, the grid placement is not simulated but properties of interest are calculated.

Except in the first scenario with the path-loss propagation model, link qualities are calculated based on the shadowing model (Section 2.1.3). The path loss exponent for shadowing model is four and standard deviation is seven which is in accordance with measurements in urban areas, for low height antennas [25].

For the sake of simplified presentation, instead of defining the threshold a_t and the sending power p_t, the nominal communication radius of a node R is used in Table 7.1 - communication radius that would exist in presence of path loss attenuation only ($\sigma = 0$). Simulation of shadowing imposes the log-normal variations to this nominal communication radius in order to determine the existence and quality of a link between two nodes.

We can calculate the nominal communication radius in the following manner: It is known that the attenuation of the wireless signal depends on the distance which the signal travels d and the path loss coefficient α: $a_{PL} \sim \frac{1}{d^\alpha}$. The threshold attenuation value a_t is then $a_t = 10log\frac{p_r}{p_t}[dB] = 10log\frac{p_r}{p_r \cdot R_0^\alpha}[dB]$. Finally, we get $R_0 = 10^{-\frac{a_t}{10 \cdot \alpha}}$.

In order to obtain referent topology for metric calculation, all links with ETX value higher than 100 (link quality below 0.1) are removed from the connectivity graph. The same threshold is applied to measurement samples, in order to eliminate the links that are not functional but the routing protocol has not realized it yet (a node on a link has left the network, but the link still remains in the topology table).

In order to create a connected network for the uniform node placement model, the communication radius of nodes has to be rather large, increasing the average node degree throughout the graph. The results of [36] and [107] show that in order to have a connected network with high probability, the average degree of nodes should be more than ten. This increases the number of disjoint paths and decreases the number of

	Freifunk Berlin	Freifunk Leipzig	Uniform (S)	Uniform	RWM
Node degree	4.0204	4.3565	5.3167	5.3049	7.6222
Bridge share	0.1506	0.07943	0.0212	0.0212	0.01249
Art. points	75.9338	93.3285	32.7673	32.8905	21.3662
Bridge ETX	5.8824	4.0795	28.4579	~ 1	27.3775
No-bridge ETX	17.5832	18.2664	20.4782	~ 1	20.8674
P_{bt}	0.743	0.468	0.352	0.35	0.127

Table 7.3.: Mean values of selected characteristics in measured and artificial topologies.

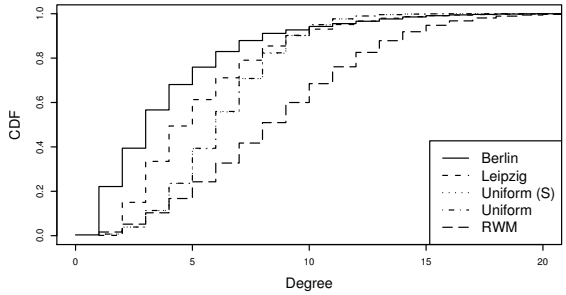


Figure 7.5.: Cumulative node degree distribution in real networks and artificial placement models.

bridges and articulation points in the network. We have selected simulation parameters for both uniform scenarios such that the average node degree is approximately 5.5. This does not guarantee the connectivity and there exist isolated nodes or smaller groups of nodes. The number of nodes that do not belong to the main partition is small and it its impact on presented results is minor. By trial and error it has been observed that further reduction of the average node degree is not acceptable. It increases the bridge count but also completely disconnects the network, creating partitions consisting of less than ten nodes. The same parameters as for the uniform scenario with shadowing are used in RWM scenario for configuration of nodes and signal propagation.

7.2. Data Analysis

In this section, the measured data is analysed and compared with the artificially produced topologies. The summary of measurement and simulation results can be found in Table 7.3. Even from mean values it is clear that properties between reality and common artificial models differ significantly. The following sections analyze the differences in more detail and provide insight in distributions of the studied metrics.

7.2.1. Node Degree Distributions

This section analyses the degree distribution of nodes in a network. The node degree distribution is correlated to the congestion on the wireless channel and probability of packet loss – higher density of nodes increases the contention, lower density reduces it. The node degree distribution also affects connectivity of a network [36].

The cumulative node degree distribution function is shown in Figure 7.5. It can be seen that artificial placement models have substantially fewer low degree nodes than real networks. The tail of the real node degree distributions is more pronounced (nodes with large number of neighbors are not uncommon) than in the uniform and grid scenarios. The RWM distribution has a very heavy tail, thus protocols simulated with use of this model are more prone to suffer from contention issues than protocols simulated with other models or deployed in reality. So, neither of artificial models possesses characteristics that are aligned with characteristics of real networks.

Grid, as the non-randomized placement model has the most peculiar properties. For comparison purposes, let us assume that the node and wireless propagation parameters are set so that the topology of the grid reassembles the one shown in Figure 2.1 (page 13). Let the nodes be organized in 20 by 20 grid. The grid is square-shaped with $n = 400$ nodes and $a = \sqrt{n} = 20$ nodes form the edge of the square. The grid has four corners, so the share of nodes with degree 3 is $P_3 = \frac{4}{n} = 1\%$. Nodes belonging to the outside edge but not in the corners have degree 5 and their share is $P_5 = \frac{4\sqrt{n}-8}{n} = 18\%$. Remaining nodes have degree 8 and their share is $P_8 = \frac{n-4\sqrt{n}+4}{n} = 81\%$. The average node degree is: $8 - \frac{12\sqrt{n}-4}{n} = 7.46$. Additionally, the grid guarantees connectivity (the presented example is 3-connected because of diagonal links) and it is resilient to node failures. Grid simultaneously eliminates high degree nodes – the high contention points in the network, reducing the issues created by it.

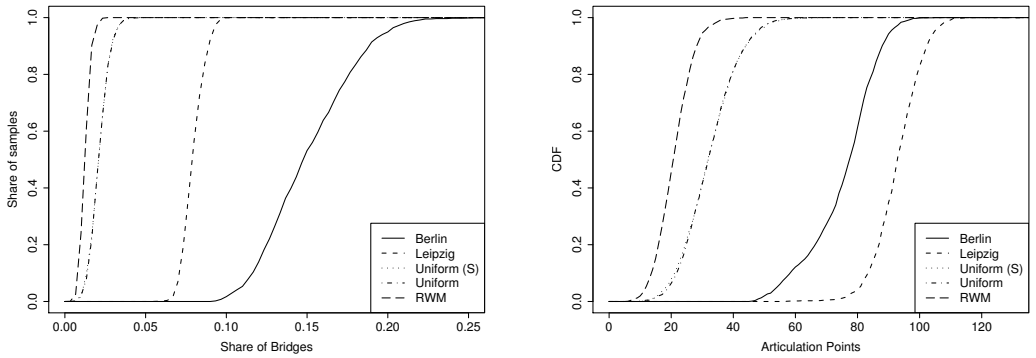
Furthermore, in literature grid placement model is frequently combined with path-loss models where communication range is slightly larger than the internode distance. That further reduces the experienced contention since node degrees may be two, three or four. In example of 20 by 20 grid, it would result in probabilities of $P_2 = 1\%, P_3 = 18\%, P_4 = 81\%$ and the average node degree of 3.8.

7.2.2. Bridges and Articulation Points Analysis

This section analyses the frequency of bridge and articulation point occurrence in different types of networks. Figure 7.6(b) shows the distribution of articulation points' count. Community networks have considerably more articulation points than the artificial placement scenarios. The network in Leipzig has more nodes than simulation scenarios, but the increase in number of articulation points is disproportional to increase in the total node count. If we observe it relatively to the network size, Berlin's network has the highest share of articulation nodes - 23.8% on the average.

Fraction of bridge edges in the network is shown in Figure 7.6(a). Again, real networks exhibit poorer connectivity and their bridge fraction is considerably larger than in the artificial placement models. Berlin's network has the largest bridge fraction: most of the samples from Berlin have between 12% and 20% of bridges. Although the fraction of bridges in Leipzig network is half the share of Berlin's network, it is still 3.8 times larger than for the uniform and 6.5 times than for the RWM scenarios. The grid placement model produces neither bridges nor articulation points.

The node degree distribution in Figure 7.5 indicates that real topologies have high share of pendant nodes - resulting in reduced connectivity on borders of the network. However, bridges are also present in central parts of the network where their impact on network functionality is even more emphasized.



(a) Bridge share.

(b) Articulation points.

Figure 7.6.: Cumulative distributions of bridge fraction and articulation point count.

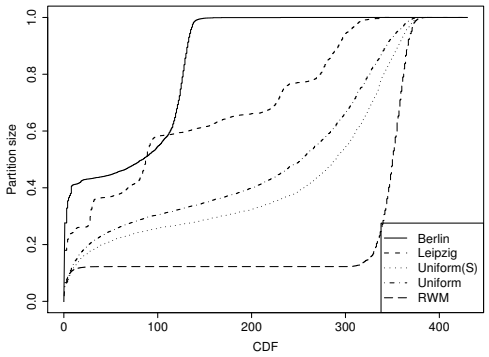


Figure 7.7.: Weighted distribution of network components obtained by bridge removal.

For Berlin's network, the probability of bridge traversal (P_{bt}, defined in Section 2.4, page 27) is particularly high – almost 0.75 (Table 7.3). The probability in Leipzig is smaller than in Berlin but larger than in the artificial scenarios and it indicates the existence of bridges in the central parts of network.

In order to study the size and relation of subnetworks that are connected over bridges, the connectivity graph is divided through removal of bridges in it, thus obtaining disconnected graph components. Each of components obtained in such a way is either at least 2-connected or it consists of a single node.

The vast majority of components has size smaller than five and distribution directly obtained from component count would be difficult to present. To offset this effect and get clearer picture of their size and count, component count is weighted by its size, relatively to the network size: $C_{rel} = \frac{C_{count} \cdot |C|}{n}$. For instance, if a network has 100 nodes and 12 of them are in three node components, weighted impact of three-node components is: $C_{rel}(3) = \frac{3 \cdot 12}{100} = 0.36$.

The distribution of relative component size can be seen in Figure 7.7. RWM distribution is skewed to the right because of its property to group nodes in central part of

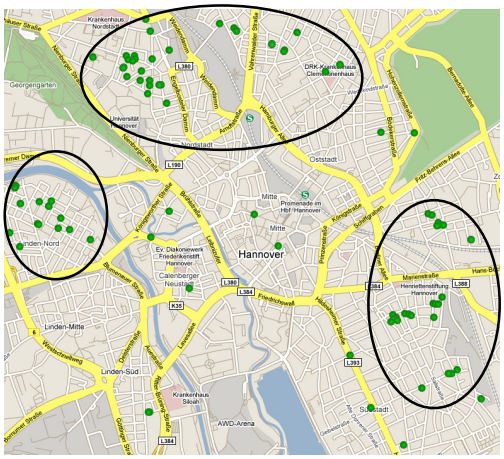


Figure 7.8.: Map of Hannover Freifunk.

the placement area. So not only that it has few bridges, but they are all placed on network borders. The uniform placement scenarios have rather uniformly distributed component size, with a slight preference for very small (below 20) and large (300 to 350) components.

Berlin network's distribution is bimodal: one part of distribution weight is placed around 110 and most of the remaining distribution weight belongs to the small components, composed of 1 to 5 nodes. Since samples from Berlin have more than 300 nodes on the average this means that for most of the time the network has two well connected components connected over one or more bridges.

The distribution of Leipzig samples has to be taken with caution. Since network in Leipzig is disconnected from the start (elimination of traffic-tunnel links), this distribution cannot tell us much except for sizes of these components (three components with sizes of approximately 80, 210 and 270). However, as the network in Leipzig grows, it is to expect that it will get connected and that the internal component structure will not change substantially, creating a structure similar to the structure of network in Berlin.

The similar properties are encountered in Hannover Freifunk [10] although it is smaller than networks in Berlin and Leipzig. We did not make measurements of it, but the topology visualization on web site of the network clearly shows clustering of nodes as well as the weak connectivity among three clusters (Figure 7.8).

Based on all the observations made in this study, it can be concluded that only parts of real networks provide multiple disjoint paths between nodes. Such network components are mutually connected over bridges and articulation points resulting in loosely connected structure. The structure is in particular contrast with grid placement scenarios that does not have a single bridge and with RWM which has few but only on the network periphery.

7.2.3. Link Quality Analysis

The Figure 7.9 shows cumulative distribution of ETX values for bridges and ordinary links. The ETX distributions for ordinary links in Berlin and Leipzig are almost identical, while bridge ETX distributions are slightly different.

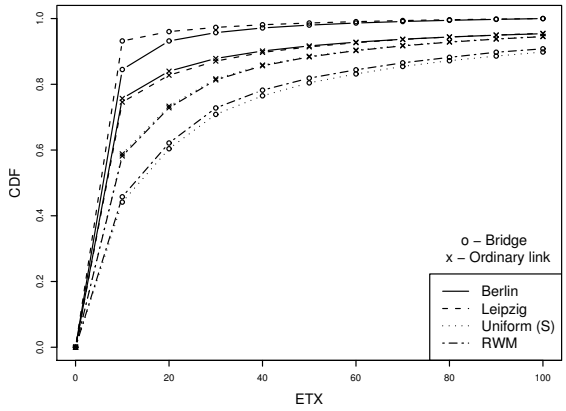


Figure 7.9.: ETX cumulative distributions.

The distributions from real networks incline to lower ETX values (which means higher link quality) than the artificial placement models. Also, in real networks, bridges have higher quality than ordinary links. In artificial placement models the situation is opposite - ordinary links have better quality. This goes in hand with typical theoretical assumption that bridges are always long, stretched communication links and due to their length, their quality suffers. In an obstacle-free model, it is more likely that a bridge existence implies larger distance between nodes that create it. But in reality, if a user is bound to use a bridge to access the network, unless the link provides certain quality of service (e.g., throughput, delay), the user will not participate in the network at all. This way, users prune bridge links with low quality at network periphery. Additionally, existence of obstacles influences communication link establishment so an edge may be a bridge despite existence of nodes in its vicinity (an example can be found in Figure 8.12 at page 151).

Important characteristics of samples is that non-negligible number of links have high ETX values, larger than 40. The probability of successful packet transmission over such links is very low. To make situation worse, some of these links are bridges.

This network characteristic may have negative influence on optimistic routing and broadcasting schemes which rely on the presence of multiple nodes in vicinity. Once they receive a packet they are supposed to forward it further so even if some packets are lost in intermediate steps, the message will be eventually delivered to destination over different path. In case of bridge traversal, it is impossible to avoid its usage since it is the only link connecting two subnetworks. If the message is not delivered over the bridge, it is lost for the adjacent subnetwork.

This similarity establishes trust in accuracy of measurements: as already described in Section 7.1.1, users are able to alter node configuration so that the node reports user-defined ETX value instead of the measured. Big differences in the ETX distribution would indicate that such user behavior is prevalent, and consequently disqualify the ETX measurements. The similarity shows that fair nodes are dominant and false reports on link quality rare.

7.3. Summary

This section has presented measurement results from community networks in Berlin and Leipzig and compared them with typical node placement models found in literature. The presented analysis is of particular importance from topological perspective since node placement and behavior is not pre-determined like in simulation and testbeds, but user initiated and controlled.

The measured topological and link quality characteristics can be attributed to a combination of sociological and technical factors that should be valid in other user-initiated multi-hop networks:

- New participants in the network are more likely to join the network in areas where the connectivity is already good. Improved connectivity (several links to the network) increases resilience to individual node failures and provides higher uptime of a connection to a user.
- Information about the network existence is spread through word of mouth, attracting new participants in areas where network is already established.
- The stochastic nature of the uniform, RWM and other random models makes occurrence of isolated nodes possible and sometimes even common. In real world a typical user is not interested to operate an isolated node without connectivity to other parts of the network.
- A participant in the network expects to have at least a single communication link to the remainder of the network.
- A pendant node may become a seed for a new, larger and well connected subnetwork.

The real networks exhibit high inhomogeneity – there exist exceptionally dense sections of network but also very sparse parts. The degree distribution observed in reality is different than the degree distribution of placement models which are widely used in theory and simulation of WMNs. Bridges and articulation points are much more numerous in reality than in artificial models.

The observed topological inhomogeneity is of high relevance for development of protocols that are to be deployed in general purpose WMNs, and in networks whose topological shape cannot be predetermined or fully controlled. Because of inhomogeneity of real networks, it can be particularly dangerous to adjust the protocol functionality for a certain node density in a network. Such protocol will operate very well in network sections that fit its assumptions, but it may face serious drawbacks elsewhere. Thus, a truly universal protocol must be able to operate in highly heterogeneous networks.

The analysis of measurements has shown that the artificial node placement models should not be used exclusively in evaluation of network protocols because of large discrepancies between properties observed in reality and characteristics of artificial topologies. The observed differences should inspire measurements in other types of WMNs and verification to measurements of other common modeling and simulation assumptions. Measurements are the only possibility to determine to which extent can we trust the artificial models, and learn the limits of their applicability.

8. NPART - Node Placement Algorithm for Realistic Topologies in Wireless Multi-hop Network Simulation

The main approach in natural sciences such as physics or biology is to observe reality and create a model that reflects it. Most of the node placement models for WMNs do not belong to this class: they have not been inspired by reality nor verified by the measurements. The disparity between existing node placement models and field measurements has been confirmed through the topological analysis performed in the previous chapter. In order to correct this fundamental issue, the Node Placement Algorithm for Realistic Topologies (NPART) has been developed.

The detailed model of a WMN is complex and it consists of six submodels: node, radio, signal propagation, node placement and mobility, packet loss, and traffic models (Chapter 2, page 11). Most of these WMN sub-models are based on real data measurements (e.g., wireless signal propagation [25], traffic models [167]) but the topology generators/node placement models are artificial and somewhat arbitrary. Thus, it was not very surprising that despite a considerable number of existing topology generation algorithms for simulation of wireless multi-hop networks it was not possible to find a single algorithm that creates output with properties similar to those observed in real networks.

The differences between characteristics of topologies of simulation setups found in literature and properties observed in measurement are illustrated in Table 8.1[1]. The predominant uniform placement model has issues in reproducing reality: in order to produce connected topologies, node density must be increased. High density may create higher channel contention in simulation studies than it will be encountered in protocol deployments. Simultaneously, the higher node density also reduces the number of bridges and articulation points in a network (or completely eliminates them), improving network's reliability and providing multiple independent paths between traffic sources and destinations.

If a networking protocol is developed for such dense environment and tested in simulation with same characteristics, its weaknesses may remain hidden from its developers and the protocol can face serious issues once it is deployed in reality. In order to perform realistic simulation, all aspects of the simulation model should agree with the observations made in real systems. NPART fills the existing gap in realistic modeling of WMN topologies.

[1] R is communication radius, n number of nodes, $\bar{d}$ is the average node degree, $\bar{b}$ and $\bar{ap}$ are average number of bridges and articulation points. The average number of network partitions is $\bar{c}$.

	n	Area	R	$\bar{d}$	$\bar{b}$	$\bar{ap}$	$\bar{c}$
Berlin	315	/	/	4.02	93.59	75.93	1
Leipzig	586	/	/	4.35	101.39	93.32	1
Ngai et al. [128]	100	200x200	40	10.34	0.24	0.43	1.09
Zhu et al. [181]	100	1500	250	7.54	1.54	2.38	1.39
	150	x1500		11.19	0.35	0.49	1.08
Wu and Li [175]	100	100x100	25	15.49	0.07	0.09	1
	300			46.67	0	0	1

Table 8.1.: Comparison of network characteristics in representative simulation setups to community networks in Berlin and Leipzig.

8.1. Algorithm Description

The goal is to develop a node placement / topology generating algorithm with the following characteristics:

- **Flexible** – it is capable to create more than one type of node placement.
- **Realistic** – if algorithm receives input based on measurements from a real network, the topologies that it produces should have similar properties as the original networks.
- **Random** – the algorithm does not merely re-create a sampled topology from measured node locations, wireless device parameters (power, receiving threshold), signal to noise ratio. It is capable to create new, random topologies while preserving its flexibility and realism.

The starting point for NPART development are the following sociological and technological factors that shape topologies of real networks (described in Chapter 7):

- It is more likely that new participants join the network in areas where connectivity is already good.
- A participant in the network expects to have at least a single communication link to the remainder of the network.
- A pendant node may become a seed for a new, larger and well connected subnetwork.
- It is the network that specifies the area it occupies, not the other way around.

The core idea behind the algorithm is that the network should be allowed to grow. Node by node should be added to the network, imitating user behavior. Instead of defining the node placement area like in most of the existing placement algorithms, nodes can expand it as long as they have connectivity with already established network topology.

The algorithm which follows these ideas is presented in Figure 8.1. As input parameters, it accepts the number of vertices to be placed n, and the communication radius R.

```
NPART(nodes n, comm.radius R, candidates to evaluate in iteration retries):
  placedNodes = place first node arbitrarily at (x,y)
  minX=maxX=x
  minY=maxY=y
  repeat
    minMetric=∞
    repeat
      repeat
        x-coordinate=U(minX-r, maxX+r)
        y-coordinate=U(minY-r, maxY+r)
        create node candidateN from coordinates
      until (candidateN ∪ placedNodes is connected)
      m=apply metric on placedNodes ∪ candidateN
      if(m < minMetric)
        bestCandidate = candidateN
        minMetric = m
      endif
    until(retries different candidates are evaluated)
    if required, update minX, maxX, minY, maxY based on bestCandidate
    placedNodes = placedNodes ∪ bestCandidate
  until(all n nodes are placed)
```

Figure 8.1.: NPART pseudo-code.

In the first iteration of the algorithm, the first vertex is placed at an arbitrary point (x,y) in two-dimensional space. The variables $minX$ and $maxX$ are initialized to x, $minY$ and $maxY$ to y. Values of these variables from iteration I_k are used to determine the placement area of nodes in the next iteration: in iteration I_{k+1}, x coordinate of candidate nodes is uniformly sampled from ($minX - R$, $maxX + R$), y coordinate is chosen from ($minY - R$, $maxY + R$). This enables the network to grow, without need to predetermine its placement area.

It is possible that in an iteration I_k a vertex placed in rectangle (($minX - R$, $minY - R$), ($maxX + R$, $maxY + R$)) is not connected to topology from iteration I_{k-1}. For instance, in the iteration I_4, the Vertex 0 in Figure 8.2 is disconnected from nodes placed in the iteration I_3. Such vertices are ignored and a new candidate vertex is generated. This ensures connectivity of produced topology (functionality is implemented in the innermost loop of the algorithm presented in Figure 8.1).

Once the candidate vertex is connected with the existing topology, a user-defined metric is applied to it. Section 8.2 describes four metrics that we have implemented and tested. If the candidate vertex has lower metric than previous candidates, it is stored as the *bestCandidate* and minimal metric value is updated. After evaluation of *retries* connected candidates, the best candidate is added to the topology, and if needed the variables $minX$, $maxX$, $minY$, $maxY$ are updated. For example, if Vertex 3 is the best candidate out of three candidate nodes in Figure 8.2, the variables $maxX$ and $minY$ must be updated. Number of evaluated candidates per algorithm iteration *retries* is a parameter of the algorithm. As the number of evaluated candidates grows, the chance

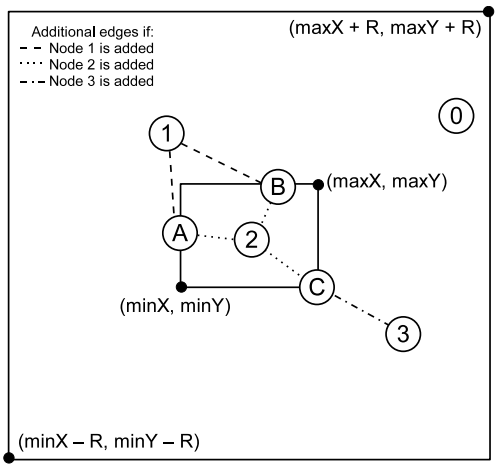


Figure 8.2.: Placement area and the candidate vertices in NPART.

that the produced topology is closer to the predefined goal is increased.

After placement of all n nodes their locations can be translated so that they are in rectangle $((0,0), (|maxX - minX|, |maxY - minY|))$. This step is optional and does not influence the functionality of the algorithm nor the properties of obtained topology.

Note: The algorithm calculates topologies based on the path-loss model. The user should specify the additional propagation models in the simulation (shadowing and Rayleigh fading [25]) to create realistic simulation results. Also, users can also customize the algorithm by specifying appropriate metrics that take propagation models in account (e.g., number of edges in the graph with expected packet loss higher than 0.5 in presence of shadowing on the wireless channel).

8.2. Topology Quality Metrics

A metric that evaluates quality of topology candidates is as important as the algorithm itself. An inappropriate metric results in unsatisfactory topologies. Unfortunately, there does not exist the universal metric. User must define them and perform tests to check whether the algorithm and the metric produce desired topologies. The desired topology characteristics for a user may be arbitrary, but our goal are topologies that have properties observed in real, user-initiated networks.

In the process selection of the input for algorithm and its metrics, it was obvious from experience with existing placement models that generic parameters such as the average node degree do not capture sufficient level of detail. Realistic topologies can be produced only with input parameters that originate from measurements.

Capturing spatial node distribution and link quality metrics (e.g., signal to noise ratio, bit error ratio, packet loss probability) in real network would be an excellent input for a vertex placement algorithm but impossible to implement. For instance, due to privacy concerns not all participants in a real network are willing to disclose their geographical locations. Even if location data is collected and its usage allowed at a research institution, dissemination of such data to wider research audience (prerequisite for cross-validation of results) would not be possible.

Additionally, quality of antennas cannot be automatically collected, signal propagation environment is heterogenous and its impact on link quality cannot be accurately measured with of-the-shelf components that are commonly used. In rare cases when it is possible to take samples from user initiated networks, typically only the topological information is available, without node location data.

We have implemented several NPART metrics based on the node degree frequencies.

Definition 8.1 *Given an undirected graph, a degree sequence is a monotonic nonincreasing sequence of the vertex degrees (valencies) of its graph vertices. A degree set is a set of integers that make up a degree sequence.*

Definition 8.2 *The frequency of an event i is the number n_i of times the event occurred in the experiment. The frequency can be absolute, when the counts n_i are given and relative, when counts are normalized by the total number of events.*

The degree frequency is a compromise between detail level, data anonymity and feasibility of sampling. It is easily extracted from networks, regardless of the routing protocol type (proactive or reactive) and it is anonymous by its definition. In proactive protocols, it is trivial to calculate it. In case of reactive routing protocols where no global topology view exists, node degrees can be easily obtained assuming that nodes in the network are cooperative: each node samples its degree and shares it with the central repository. Additionally, small errors in sampling (e.g., a node erroneously measures its degree) are hidden by the larger set of correct data.

The implemented metrics are described in Figure 8.3. As the input for *distance* and *adaptive* metrics, the relative node degree frequency is calculated from degree sequences from measurements. The relative node degree frequency of real network is multiplied by number of nodes that should be placed by the algorithm, creating absolute vertex degree frequency of the target topology *target*. For each evaluated candidate vertex, absolute degree frequency *candidate* of topology that it creates with already placed nodes is calculated and compared with the target frequency.

The simplest, *distance* metric is a variation of the Manhattan metric [101]:

$$\sum_{degrees}^{d} (1_{target_d - candidate_d > 0} \cdot (target_d - candidate_d) + 1_{target_d - candidate_d < 0} \cdot p \cdot (candidate_d - target_d)) \quad (8.1)$$

where $1_{A(x)}$ is indicator function, returning one if $x \in A$, zero otherwise. The metric sums the differences between proposed and target vertex frequency if the difference is positive. If it is negative (candidate topology has more nodes of certain degree than the target topology), the absolute value of difference is multiplied by a penalty factor p. The penalty factor reflects user's tolerance for overloading of degrees: with decrease in tolerance, user increases the factor p. If $p = 1$, the *distance* metric is identical to the Manhattan metric.

The drawback of the *distance* metric is its impossibility to detect stronger need for creation of vertices with certain degree. Some degrees are more frequent in the target degree frequency, so topologies that produce vertices with such degrees should obtain greater reward. For instance, if the algorithm should create 20 vertices with degree

distance metric(target frequency *target*, candidate frequency *candidate*, penalty *p*):
  $metric = 0$;
  for (i in degrees of *target*)
    if(target[i]-candidate[i]<0)
      $metric = metric + |target[i] - candidate[i]| \cdot p$
    else $metric = metric + target[i] - candidate[i]$
  return *metric*

adaptive metric(target frequency *target*, candidate frequency *candidate*,
frequency of already placed *placed*, penalty *p*):
  $weights = normalize\_to\_one(|target - placed|)$
  $metric = 0$;
  for (i in degrees of *target*)
    if(target[i]-candidate[i] < 0)
      $metric = metric + |target[i] - candidate[i]| \cdot p$
    else $metric = metric + (target[i] - candidate[i]) \cdot weights[i]$
  return *metric*

secondary-distribution metric(candidate graph topology *topology*,
secondary degree relative frequency *secondary_target*)
  Initiate list of empty absolute degree frequencies *list*
  for each node *n* in *topology*
    select frequency based on current node degree
    $current\_freq = list[degree(n)]$
    for each neighbor *ng* of *n*
      $current\_freq[degree(ng)] + +$
    endfor
  endfor
  list=calculate relative frequencies from absolute frequencies
  $metric = 0$
  for i in 1:length(secondary_target)
    $metric = metric + |secondary\_target[i] - list[i]|$
  return *metric*

combined(target frequency *target*, candidate frequency *candidate*,
frequency of already placed *placed*, penalty *p*, secondary weight *s*,
secondary degree relative frequency *secondary_target*, candidate graph topology *topology*)
  return **adaptive metric**(target, candidate, placed, p) +
secondary-distribution metric(target,topology) $\cdot s$

quality(target frequency *target*, generated frequency *generated*)
  $quality = 0$
  for (i in degrees of *target*)
    $quality = quality + |target[i] - generated[i]|$
  return *quality*

Figure 8.3.: Implemented metrics.

two and three vertices with degree four, the metric should give greater reward (smaller metric value) to topologies that increase the number of vertices with the degree of two in early iterations of the algorithm. The *adaptive* metric resolves this issue through introduction of weights that capture this generation need:

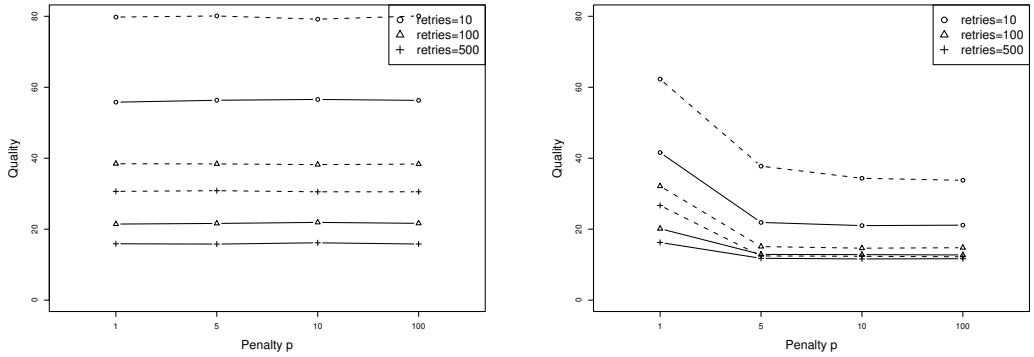
$$\sum_{degrees}^{d} (1_{target_d - candidate_d > 0} \cdot (target_d - candidate_d) \cdot w_d$$
$$+1_{target_d - candidate_d < 0} \cdot p \cdot (candidate_d - target_d)) \tag{8.2}$$

and

$$w_d = \frac{|target_d - placed_d|}{\sum_{degrees}^{d} |target_d - placed_d|} \tag{8.3}$$

Degrees	1	2	3	4	5	Distance metric	Adaptive metric
Absolute Target degree frequency	2	5	3	2	1	0	0
Absolute Placed degree frequency	0	3	0	0	0	10	2
Weights w_d	0.2	0.2	0.3	0.2	0.1		
Candidate 1	0	2	2	0	0	9	1.8
Candidate 2	0	0	4	0	0	15	6.9
Candidate 3	1	2	1	0	0	9	1.9

Table 8.2.: Metric values for example in Figure 8.2. Penalty is set to five.



(a) Distance metric.

(b) Adaptive metric.

Figure 8.4.: The average Manhattan distance to target degree distribution of topologies produced by *distance* and *adaptive* metrics of NPART (lower value is better).

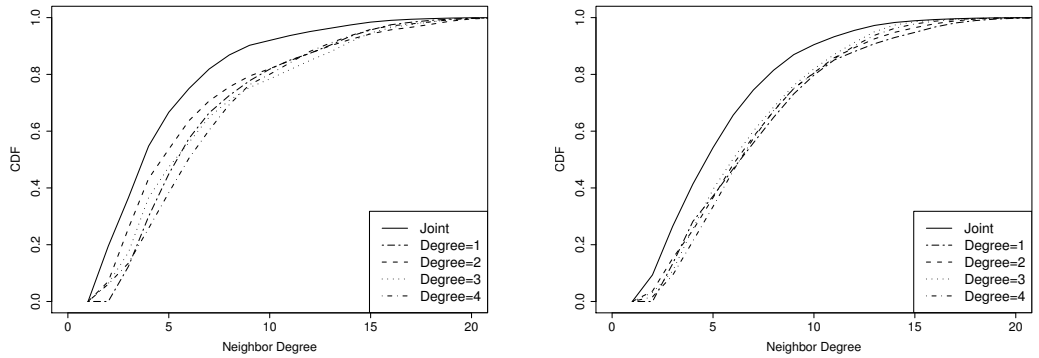
where *placed* is the absolute degree frequency of vertices that are already placed by the algorithm.

Table 8.2 illustrates the application of *distance* and *adaptive* metrics to the example from Figure 8.2. Nodes A, B and C are already placed. Topology obtained in I_3 defines the weights w_d for the I_4. In fourth iteration three candidate vertices are successively evaluated.

For each candidate vertex we calculate metric values by Equations 8.1 and 8.2. The *distance* metric has the equal value for candidate vertices 1 and 3, so either of them can be selected as the best candidate vertex. Adaptive metric correctly chooses candidate vertex 1 as better, since it satisfies greater need to create node of degree three (after I_3, three more nodes with degree three are required) than to create node of degree one like the candidate vertex 3 (after I_3 two more nodes of degree one are needed to reach the absolute target degree frequency).

Figure 8.4[2] compares the behavior of the placement algorithm if it is used with the *distance* and *adaptive* metrics, for different combinations of parameters p and *retries*. As the input of the algorithm, the degree frequencies from Freifunk Berlin and Freifunk Leipzig networks are used. The Manhattan metric between targeted and produced absolute degree frequencies is used as the quality measure of the produced topologies (lower value is better).

[2]Solid line represents Leipzig-like topologies while dashed is used for topologies similar to the network in Berlin.



(a) Berlin. (b) Leipzig.

Figure 8.5.: Conditional degree distributions.

The topologies produced with the *adaptive* metric are considerably better (have lower distance to target) than the topologies produced with the *distance* metric. The *distance* metric is invariant of the penalty parameter. The *adaptive* metric is sensitive to the penalty parameter, but if penalty parameter is equal to or larger than five, the quality of produced topologies stabilizes. As expected, an increase in number of retries improves quality of produced topologies for both metrics. However, the quality is not drastically improved (Manhattan distance to target distribution reduced) if number of retries is increased from 100 to 500. Based on this evaluation, we conclude that *adaptive* metric is better, and it will be analyzed in more detail in Section 8.3.

It is possible to refine the degree data distributions and extract some additional data from topologies. Let us observe relative degree frequency of neighbors of a node A under condition that degree of A is i. As Figure 8.5 shows, the conditional relative frequencies differ among themselves and to the joint relative frequency.

The metric *secondary-distribution* utilizes these differences. First, it calculates set of conditional relative degree frequencies for candidate topology and then compares them (using Manhattan metric) with the target conditional relative degree frequency.

The *combined* metric is a linear combination of the *secondary-distribution* and *adaptive* metrics. It allows user to vary the penalty factor p in the *adaptive* metric and the weight s for the *secondary-distribution* metric.

8.3. Evaluation of Characteristics of Topologies Created by NPART

This section compares properties of topologies produced by the NPART algorithm with properties of real networks. The quality of the algorithm is demonstrated through similarity of its topologies and measurements. Different combinations of algorithm parameters are tested, in order to find those that produce topologies similar to reality.

The data that is used to provide input degree distributions for the algorithm and for later comparison is taken from Berlin and Leipzig networks. There is a small change in datasets if compared with data presented in Chapter 7: only the main partition (largest

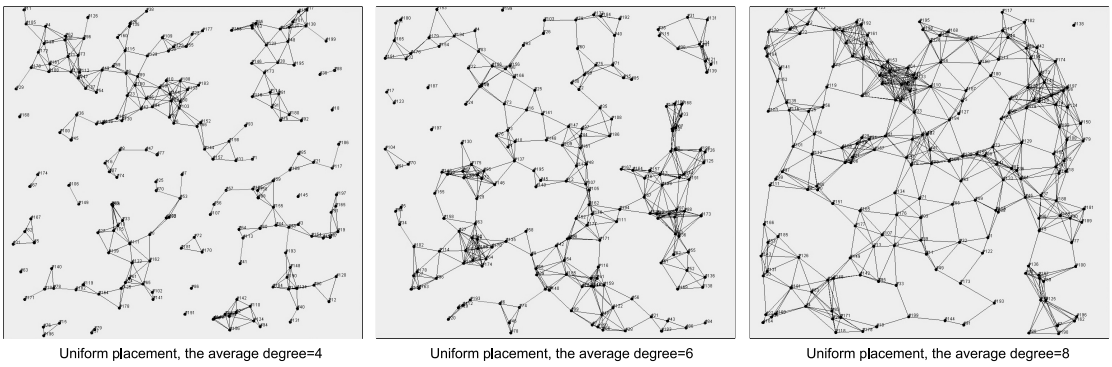


Figure 8.6.: Visual comparison of topologies created by the uniform node placement model.

maximally connected subnetwork) is considered for algorithm's degree data input and later result comparison. The main partition in Berlin has 275 and in Leipzig 346 nodes on the average.

To illustrate the improvements brought by the NPART, the analysis includes characteristics of topologies created by the uniform placement model. The uniform placement algorithm is set to create topologies with the average node degree of six. The average node degree is substantially lower than proposed in [36] and [107] for networks that are to be connected with a high probability so it is possible that such a graph is partitioned. Increasing the average node degree above six improves connectivity but creates even greater discrepancy with measurement results (e.g. bridges do not exist in generated topologies), while decreasing it creates highly partitioned graphs (Figure 8.6). So, the selected average node density provides a compromise between these two opposing trends[3].

Since the size of Berlin's and Leipzig's main partition differs there are also two uniform placement scenarios with 275 and 346 nodes respectively. The data for comparison is collected in 500 executions of each scenario.

The NPART is run with two basic setups: 275 vertices and degree data input from Berlin's network (NPART/Berlin), and 346 vertices and data input from Leipzig's network (NPART/Leipzig). The parameter *retry* is set to 150 while parameters for penalty p and secondary metric weight are varied to take values from set $\{0, 1, 5\} \times \{0, 1, 5\}$. Not all results are presented since some parameter combinations do not create reasonable results: as soon as the weight of *secondary-distribution* s is higher than the penalty p, the algorithm becomes unstable and creates almost fully connected graphs. We conclude that the *secondary-distribution* metric is excellent for refinement of the *adaptive* metric, but it should not be used on its own.

Figures 8.6 and 8.7 informally illustrate differences between topologies created by the uniform placement model, sample real topology and topologies created by NPART. None of uniform placements resemble the shape of real topology from Figure 8.7. Although visual representation of topologies in Figures 8.6 and 8.7 gives valuable insight in shape

[3]In the previous chapter, a slightly sparser uniform node placement has been used with the average node degree of 5.5 but there was no major increase in bridge or articulation point count. Only the number of graph components was slightly increased.

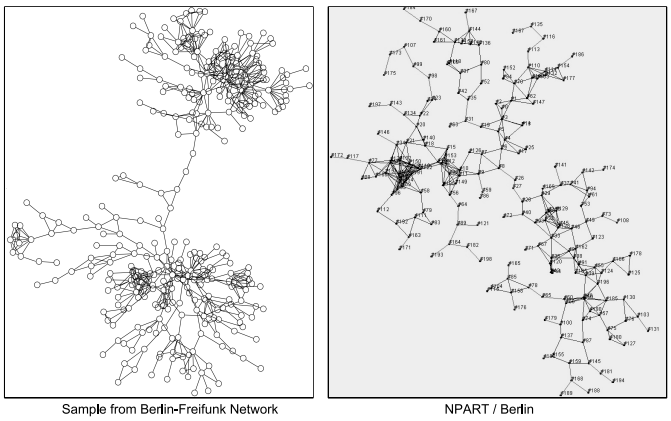
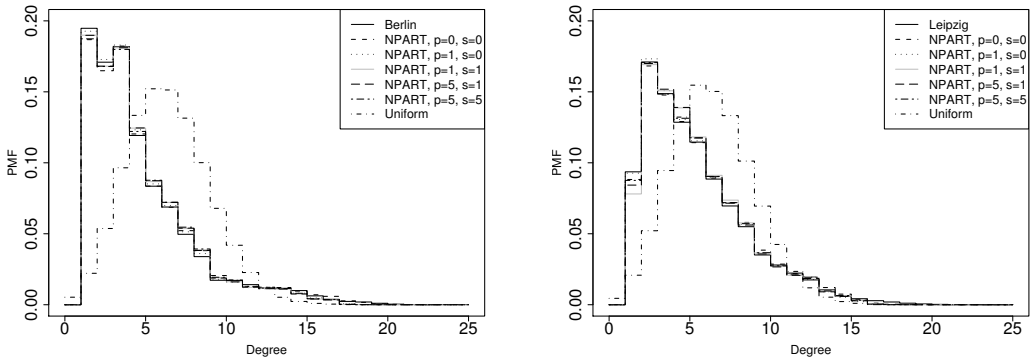


Figure 8.7.: Visual comparison of a real and a NPART generated topology.



(a) NPART/Berlin

(b) NPART/Leipzig

Figure 8.8.: Comparison of node degree distributions.

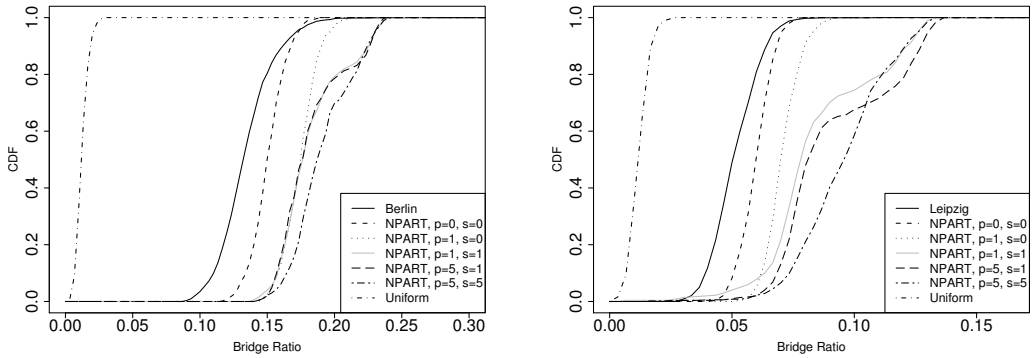
and characteristics of generated topologies, such informal comparison is not sufficient. The next section statistically compares the properties of uniform, NPART generated and real topologies.

8.3.1. Properties of Generated Topologies

For comparison of real, NPART, and uniform node placement model topologies we use the metrics from Chapter 7: degree distribution, articulation point count, share of bridge edges and relative component size after bridge removal.

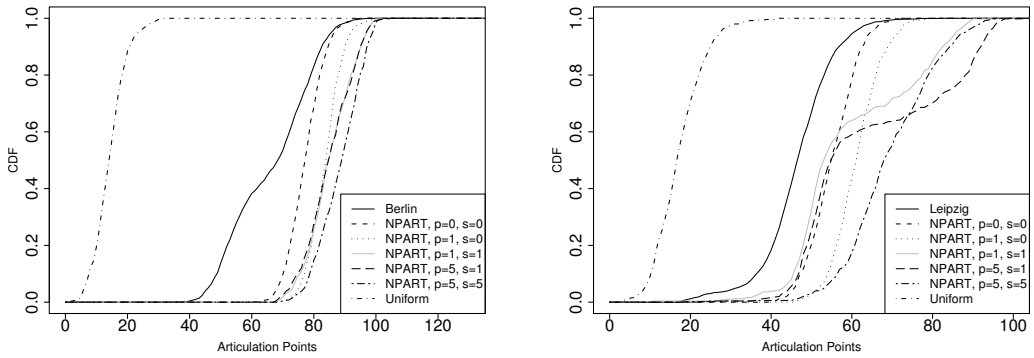
These metrics are easily quantified and they relate to characteristic of networks (e.g., network with more articulation point is more likely to disconnect if a node fails) and protocols deployed in them (e.g., bridges get easily congested with traffic).

Figure 8.8 shows the vertex degree probability mass function (PMF). As it can be seen, for all parameter combinations, the degree distribution of topologies created by NPART precisely follows the distribution of measurements. The algorithm adapts with ease to both distributions. The uniform distribution has its own shape that is considerably



(a) NPART/Berlin (b) NPART/Leipzig

Figure 8.9.: Cumulative distributions of bridge to edge ratio for real samples and NPART generated topologies.



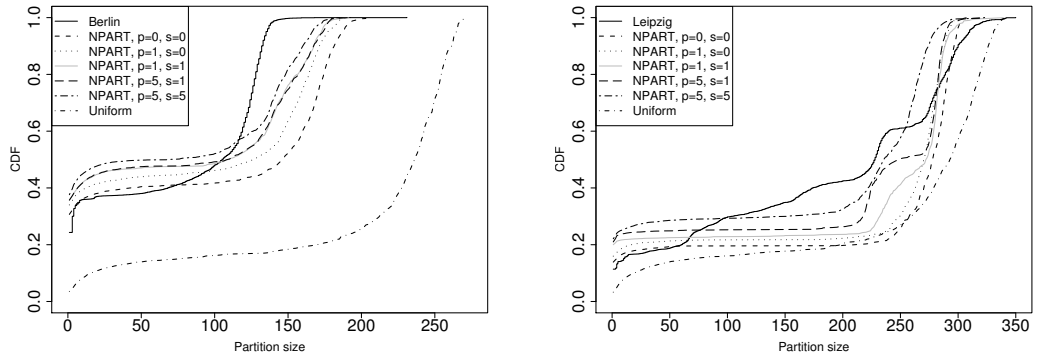
(a) NPART/Berlin (b) NPART/Leipzig

Figure 8.10.: Cumulative distributions of articulation point count for real samples and NPART generated topologies.

different from real distributions. It also has zero-degree nodes, indicating existence of partitioned topologies.

The bridge to edge ratio (Figure 8.9) and articulation point count (Figure 8.10) show that NPART topologies follow the properties of real networks. However, the proposed algorithm creates slightly more bridges and articulation points than it should. The uniform placement model is unable to adapt nor to represent the reality: its topologies have less than 1% of bridges and a few articulation points.

Figure 8.11 shows the cumulative distribution of the relative component size obtained by removal of bridges. NPART is again considerably better than the uniform placement model, in particular for the Berlin's network. Topologies produced with the assistance of *secondary-distribution* metric have distributions more aligned with real measurements than samples produced exclusively by the *adaptive* metric, both for Leipzig and Berlin distributions.



(a) NPART/Berlin

(b) NPART/Leipzig

Figure 8.11.: Cumulative distribution of relative size of network components obtained by bridge removal.

8.3.2. Offsetting the Imprecision Brought by Simplified Environment and Signal Propagation Modeling

Although considerably better than existing topology generators, the algorithm can be further improved since it creates more bridges and articulation points than it should.

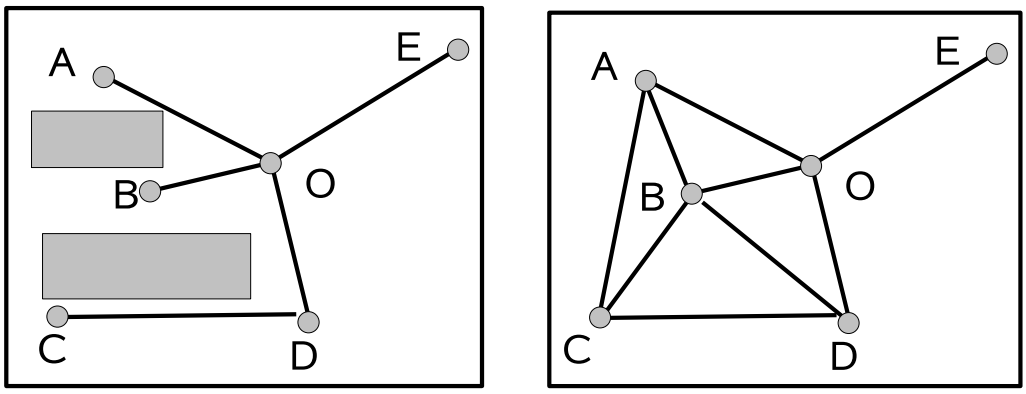
We have carefully investigated the input data and the original topology samples. It has been revealed that some nodes with high degree have several pendant nodes attached to them. In reality it is possible because of:

- The correlated shadowing on wireless channel.
- Heterogeneous node equipment.
- Directional antennas and partial network planing.

The correlated shadowing on wireless channel may cause that nodes which are physically close cannot communicate (as shown in Figure 8.12(a), because of obstacles, links AB and AC do not exists despite the small distance between them). Correlated shadowing model exists for single-wireless-hop analysis [82] but it is not supported in discrete event simulators for multi-hop networks. The propagation models supported by simulators create links as a function of internode distance (Figure 8.12(b)). As the consequence, there are less pendant nodes in simulator than in reality for identical placement of nodes.

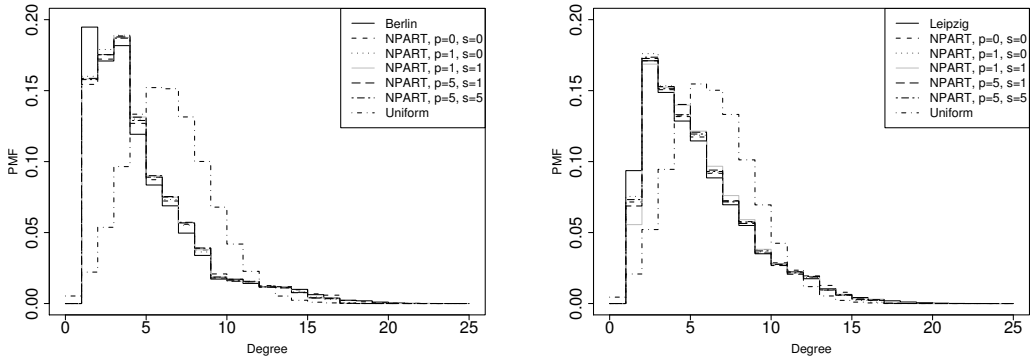
If high number of pendants is requested from placement generator, it cannot fulfil this request without affecting other characteristics of generated topology: it creates the requested number of pendant nodes, but it also increases number of bridges and articulation points more than it should.

To overcome this issue, there exists an option in NPART tool, so that a user may reduce the number of generated pendant nodes. The user has freedom to decide which topology characteristic is more important for him/her: accurate degree distribution, or better fit with bridge and articulation point distributions. We have performed numerous



(a) Correlated shadowing. (b) No correlated shadowing.

Figure 8.12.: Effects of correlated shadowing on network topology.



(a) NPART/Berlin (b) NPART/Leipzig

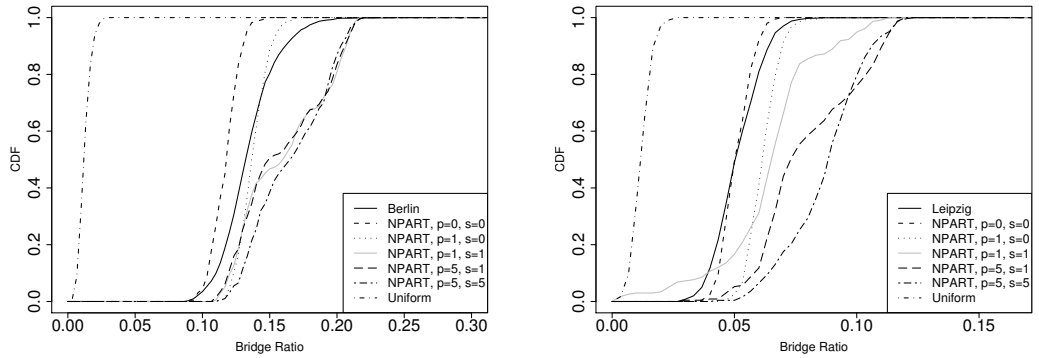
Figure 8.13.: Comparison of node degree distributions after reduction of pendant node count by 20%.

tests and empirically found that the 20% reduction ratio of pendant node count provides good results.

The effects of reduction of bridge and articulation point count are to be seen in Figures 8.14 and 8.15 since topology characteristics are closer to real distributions than in Figures 8.9 and 8.10. The degree distribution (Figure 8.13) follows closely the real distribution, except of course for nodes of degree one.

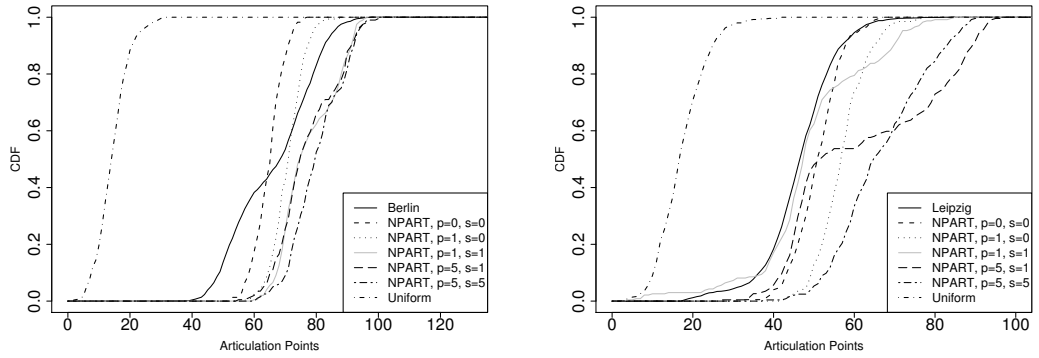
The relative component size distribution in Figure 8.16 retains good fit with reality as for the original distribution. It also demonstrates the importance of *secondary-distribution* metric: in Figure 8.16(a) the topologies created without it have poorer alignment with reality than in Figure 8.11(a), while topologies that have used the secondary metric remain as good as they were.

In order to summarize and quantify the differences between uniform placement model, NPART algorithm and reality, we have calculated Mean Square Error (MSE) between characteristics of the measurement results and the generated topologies. The value of



(a) NPART/Berlin (b) NPART/Leipzig

Figure 8.14.: Cumulative distributions of bridge to edge ratio for real samples and generated topologies after reduction of pendant node count by 20%.

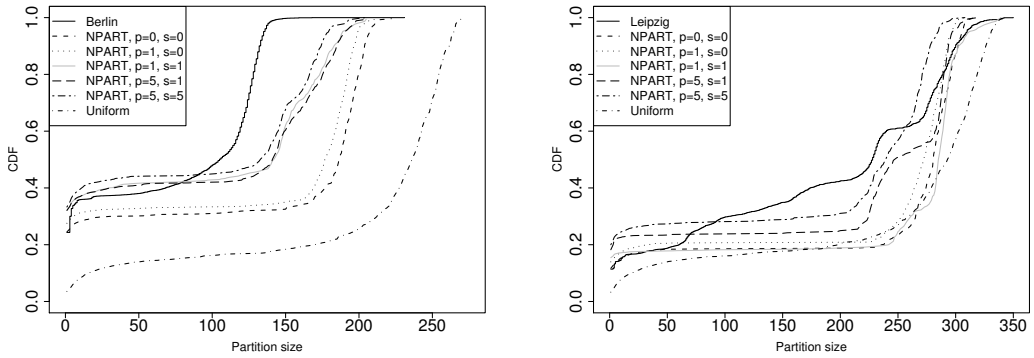


(a) NPART/Berlin (b) NPART/Leipzig

Figure 8.15.: Cumulative distributions of articulation point count for real samples and generated topologies after reduction of pendant node count by 20%.

the mean square error of a single approach is difficult to interpret: although it is known that a better approach has smaller value of MSE, it is not possible to determine a MSE threshold that guarantees acceptable approach. However, MSE is an excellent metric for mutual comparison of multiple approaches since it provides their relative ordering when compared to measurements. Tables 8.3 and 8.4 show MSE values for distributions of node degree, bridges and articulation points with and without reduction in pendant vertex count. The advantage of NPART is clear as it has by order of magnitude smaller MSE values than the uniform node placement model.

Based on MSE values and Figures 8.11 and 8.16 we conclude that the parameter combination of $p = 5, s = 1$ provides the best compromise between bridge share and articulation point count (fit with reality decreases with increase in s) and relative component size (fit with reality improves with increase in s).



(a) NPART/Berlin

(b) NPART/Leipzig

Figure 8.16.: Cumulative distribution of relative size of network components obtained by bridge removal after reduction of pendant node count by 20%..

	Degree Berlin	Bridges Berlin	Art.Points Berlin	Degree Leipzig	Bridges Leipzig	Art.Points Leipzig
NPART, p=0, s=0	7.254e-06	0.001295	0.000924	4.042e-06	0.001185	0.001108
NPART, p=1, s=0	2.779e-06	0.002929	0.001768	2.307e-06	0.002959	0.002368
NPART, p=1, s=1	7.370e-06	0.002221	0.001311	2.084e-05	0.002156	0.000625
NPART, p=5, s=1	7.558e-06	0.002298	0.001155	1.233e-05	0.002436	0.000761
NPART, p=5, s=5	9.028e-06	0.002363	0.001650	4.334e-06	0.002334	0.002004
Uniform	3.85e-03	0.007200	0.003224	1.775e-03	0.005367	0.003199

Table 8.3.: Comparison of mean square errors.

	Degree Berlin	Bridges Berlin	Art.Points Berlin	Degree Leipzig	Bridges Leipzig	Art.Points Leipzig
NPART, p=0, s=0	9.187e-05	0.001191	0.002057	2.805e-05	0.000416	0.000461
NPART, p=1, s=0	7.701e-05	0.000580	0.001250	2.281e-05	0.001812	0.001607
NPART, p=1, s=1	8.531e-05	0.000647	0.000865	8.906e-05	0.001258	0.000251
NPART, p=5, s=1	8.422e-05	0.000547	0.000720	4.342e-05	0.001622	0.000629
NPART, p=5, s=5	8.491e-05	0.000828	0.000811	2.580e-05	0.002210	0.001641
Uniform	3.85e-03	0.007200	0.003224	1.775e-03	0.005367	0.003199

Table 8.4.: Comparison of mean square errors after pendant node count reduction.

8.3.3. Analysis of Algorithm's Execution Time

The complexity of the algorithm is difficult to calculate because it includes a stochastic deciding process in it: the innermost loop is repeated until a connected graph is produced. The probability of generating a connected topology depends not only on number of nodes, but on metric type, user preferences and choices from previous algorithm iterations. Still, it is necessary to evaluate the time algorithm needs to create a topology in order to demonstrate the algorithm behavior that can be experienced by a user.

For evaluation we measure the time needed to generate a topology. The test server has 32GB memory and 8 Dual-core AMD Opteron processors working at 2.6GHz. The algorithm is implemented in Java programming language and run in Sun's Java 2 Runtime Environment Standard Edition (build 1.5.0_06-b05). However, the algorithm imple-

mentation uses only one processor core at a time and up to 256MB of memory[4]. The measurements in this section are based on 500 executions of the algorithm.

As a reference point for the execution time of the algorithm, we have also implemented the uniform placement algorithm and run it on the same test computer under identical conditions. The task was to produce a connected graph consisting of 275 vertices with the average vertex degree of 4.17 (the values are equal to the average degree and the average number of nodes in the main partition of Berlin's network). The time required to place nodes and to create connectivity graph is measured. Connectivity testing is not included in the measured time. At first, the time required for fully connected graph, where all 275 vertices belong to the same graph component, was measured. However, after several hours of attempts, the uniform placement algorithm did not produce even a single fully connected topology. The connectivity condition was then weakened, and the time required for creation of topologies whose biggest partition contains at least 97.5% of placed nodes was measured.

With the weaker condition the uniform placement algorithm needs 489.8 seconds per topology on the average. The uniform placement algorithm is substantially faster for a single node placement execution, but for low-degree networks its topologies are partitioned, and in our tests the average 66623 retries were required to meet the 97.5% connectivity condition. These numerous retries increase the total execution time of the uniform node placement algorithm.

NPART is considerably faster for such sparse graphs. If only *adaptive* metric is used, topology is created in only 2.84 seconds on the average. The *combined* metric is more demanding and its average execution time grows to 288.56 seconds. Still, that is 41.1% faster than the uniform placement algorithm with the 97.5% connectivity constraint. For sparser graphs or tighter connectivity conditions it is to expect that difference increases in favor of NPART. Also, if user needs a connected topology produced by the uniform node placement generator, the connectivity testing would have to be performed, increasing total execution time (connectivity testing has $O(n^2)$ complexity), and further increasing the execution time difference in favor of NPART.

If some user prefers to create topologies faster, we have implemented a version of the algorithm that does not necessarily execute all retries: Once the best-candidate is added to current topology in iteration I_k, its metric m_k is memorized and transferred in next algorithm iteration I_{k+1}. Instead of testing *retries* candidates in iteration I_{k+1}, the process is broken as soon as the current candidate has lower metric value than m_k. This provides a speed-up of four to five times but the quality of produced topology is reduced by the same order.

8.3.4. Effects of Network Topology on Simulation Results

This section demonstrates the effects of node placement algorithm to simulation results. For this purpose, we have used ns2 simulator [70], version 2.29 with Rayleigh-Ricean fading extension [140]. Nominal communication range of nodes is set to 250m. There are six distinct simulation setups:

- Grid node placement and path-loss propagation model
- Grid node placement and Rayleigh propagation model

[4]It is determined by the parameter `Xmx` of the Java virtual machine.

- Uniform node placement and path-loss propagation model
- Uniform node placement and Rayleigh propagation model
- NPART/Berlin node placement and path-loss propagation model
- NPART/Berlin node placement and Rayleigh propagation model

Grid consists of 272 nodes, put in 16 rows and 17 columns. Internode distance is 200m. For uniform node placement, 275 nodes are placed in 2700x2700m area, producing average node degree of 7.4. Such parameter selection creates network that is not too dense but connected with rather high probability. NPART algorithm generates 275-node topologies, using data input from Berlin's network and combined metric (s=1, p=5). The radius for NPART was also set to 250m. Routing protocol is AODV [137]. Its default parameters have not been changed except increase in the network diameter (the parameter NET_DIAMETER in [137]).

The average throughput per flow is the observed characteristic. TCP flows are created between randomly selected pairs of nodes. The number of TCP flows is varied (4,6,8,10) in each of simulation setups in order to test protocol's behavior under different network loads. In order to avoid counting of unsuccessful flows (their average throughput is always zero) that are caused by a partitioned network, uniform placement topologies are tested for connectivity before they are accepted for simulation. Simulations are performed on 50 different topologies except for grid where all topologies are identical.

Warm up phase is set to 30s and simulation is executed for 250s. Warm up phase of 30s is sufficient because AODV is a reactive routing protocol with aggressive purging of inactive routes. Prolonging the warm up time cannot change results: nodes that are required to maintain the local connectivity have enough time to execute neighbor detection process (Section 6.10 in [137]) because of frequent heartbeats. Inactive routes are deleted from route cache in less time than the duration of the warm up phase (depending on implementation, route deletion from a cache may vary between three and six seconds as proposed Section 10 in [137], to ten as it is set in ns2, up to 30 seconds as it is set in Jist/SWANS) so the extension of the warm up phase cannot change the routing efficiency. Runtime of 250s is sufficient for TCP (that is used as transport protocol for FTP traffic) to leave initial slow-start phase: the total simulation time is much longer than the packet round-trip times in networks that were simulated. Increasing this value to larger values provides consistent results: we have performed a test, increasing simulation run time to 500s for uniform placement scenario with 10 flows under two-ray-ground propagation model. Average throughput with 95% confidence intervals was 5708.2 [4932.3, 6484.0] which is very close to 5657.1429 [4980.2, 6334.1] obtained for 250s runs.

As expected and already shown in research studies [126] [157], there exists a considerable difference in simulation results between over-simplistic, over-optimistic path-loss propagation model and realistic Rayleigh model: obtained throughput is considerably higher and number of unsuccessful flows is considerably lower for the path-loss model. Also, within the same propagation model there exist differences in throughput between grid, uniform node placement, and NPART produced topologies: in NPART topologies throughput is lower than in both artificial placements.

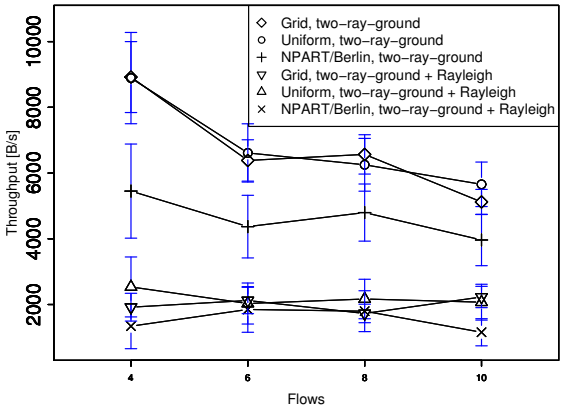


Figure 8.17.: The average throughput per flow for different topology types and signal propagation models.

8.4. Summary

NPART - Node Placement Algorithm for Realistic Topologies has been developed and evaluated in this chapter. Statistical analysis has been applied in order to compare NPART-produced topologies with the uniform placement algorithm and with real networks in Berlin and Leipzig. The properties of topologies produced by NPART fit closely to measurements of real networks. Based on the detailed literature review performed in Section 2.1.1 (page 12) it can be claimed that this is the first node placement algorithm for wireless multi-hop networks capable of creating topologies that have properties observed in user initiated networks.

The importance of accurate WMN models has been shown in literature (e.g., signal propagation models in [126] [157]) but the impact of topology generators on simulation outcome is often overlooked. In order to demonstrate importance of node placement models in simulation, the average throughput per flow in topologies produced by the grid, uniform placement model and NPART algorithm under simplistic path-loss and Rayleigh signal propagation models has been compared. Simulation results show that node placement model plays as important role in simulation outcome as the wireless signal propagation model.

Implementation of NPART is available at project's webpage[5]. User can specify type of input data, number of retries, penalty p, weight for the *secondary-distribution* metric, and output format. The tool currently supports ns2 and .dot output formats. An importer of ns2 topologies for Jist/SWANS simulator is also available for download.

Hopefully, the presented analysis and the developed tool will encourage the research community to use realistic modeling in all segments of simulation setup, thus increasing the simulation quality and narrowing the gap between simulation and reality.

[5] www.rok.informatik.hu-berlin.de/rok/npart

9. Implementation and Verification of the Approach

The correctness of DIBADAWN was proven under assumption that no packets are lost in network. However, reality of wireless communication teaches us that message losses and node failures are imminent. Packet losses cause faults in the detection process. The algorithm was tailored in Chapter 5 in order to make it operable in such environment. The changes guarantee termination, accurate order of execution of algorithm's steps, and recognize some faults, but they cannot eliminate all decision errors.

The effects of losses on algorithm's accuracy have been discussed in Section 5.5. In Section 5.6 a two-round voting scheme as a mechanism for reduction of effects of packet losses has been proposed. This chapter evaluates the proposed approach in presence of packet losses, improvements brought by the proposed voting rules, and compares DIBADAWN with solutions from related work.

The chapter is organized as follows: Section 9.1 describes overall evaluation methodology. Section 9.2 analyzes existing approaches that use proactive topology management for biconnectivity testing [57] [80] [168] on example of the OLSR routing protocol. The results of proactive approach are used as a reference point for DIBADAWN's results.

DIBADAWN and voting strategies are verified in experiments on Motelab testbed (Section 9.3) and in Jist/swans simulator (Section 9.4).

In simulator, DIBADAWN is evaluated in different static topologies and environments in Section 9.4.2, while Section 9.4.3 analyses sensitivity of approach with regard to mobility of nodes, link acceptance threshold, and use of inaccurate weights in the weighted voting rules. In Section 9.5 DIBADAWN is applied only locally (forward search radius of the algorithm is limited), in accordance with the analytical results derived in Chapter 6 and the accuracy of algorithm is evaluated. Section 9.7 provides a sample application of DIBADAWN for success rate improvement of route searches of reactive routing protocols. Overview of voting rules and summary of their characteristics is presented in Section 9.8.

9.1. Overview of the Evaluation Methodology

The proposed approach is evaluated by simulation and through experiments. A common misconception in literature is to confuse simulation with experiments (in particular, to claim that a simulation outcome is an experiment), or to create even more nebulous expression: "simulation experiment". In this work, clear distinction between them is made, and their natural definitions [11] are followed:

Definition 9.1 *Experiment is an operation or procedure carried out under controlled conditions in order to discover an unknown effect or law, to test or establish a hypothesis, or to illustrate a known law.*

Definition 9.2 *Simulation is the imitative representation of the functioning of one system or process by means of the functioning of another.*

Both simulation and experiments are used in the evaluation in order to obtain good coverage of the modeling space and to avoid over-fitting of the approach to a particular testing environment:

- DIBADAWN is verified in Motelab wireless testbed. In a testbed, there are no hidden assumptions with regard to signal propagation, node, and radio performance. Even the small details which are essentially irrelevant for algorithm's verification may influence experiment results[1]. The issues of validation in testbed are its limited size and fixed position of participating nodes. The topology shape is rather stable so it may systematically improve or degrade the measured accuracy.

- Jist/swans simulator enables testing of the algorithm in considerably larger networks and in various topologies. In simulator it is easy to study the effects of parameter selection to algorithm performance and accuracy. Also, node mobility can be introduced (impossible in testbeds with remote access). The main weakness of Jist/swans simulator is homogeneity of its signal propagation model (the same issue applies to other simulators such as ns2 or OPNET). As explained in Chapter 7, large networks are deployed in highly inhomogeneous environment where propagation parameters vary. In Jist/swans one type of propagation environment has to be chosen and it applies to all participating nodes. For instance, Rayleigh fading[2] correctly models signal propagation without line of sight between sender and receiver. However, in a large network, consisting of several hundreds of nodes it is unlikely that there does not exist even a single pair of nodes in line-of-sight setup.

Both evaluation environments have their good sides and weaknesses. Together they form comprehensive testing suite that covers various situations in which the devised approach may be deployed and applied. Agreement among their results ensures us that the algorithm is independent of particular simulation or testbed setup, and that pitfall of over-optimization for a certain use-case is avoided. The results of experimental evaluation are presented before simulation studies, since the experimental results are used as a reference point for simulation, allowing selection of realistic simulation parameters.

All voting rules presented in Section 5.6 are implemented and evaluated. However, not all of them are presented in detail. One of the main purposes of the evaluation process is to select the promising voting rules so rules that underperform are not analyzed in detail. For brevity, in this section can be found only a digest of complete evaluation results which can be found in Appendix D.

For each experiment/simulation setup three different result types are presented: precision, recall, and F-measure (the detailed description of metrics can be found in Section 2.4, page 27).

[1] For instance, memory management must be implemented with care since a node that depletes its complete memory due to mismanagement, cannot allocate it for DIBADAWN message manipulation in the following executions of the algorithm.

[2] In simulation, Ricean or Rayleigh fading are superimposed on the two-ray-ground propagation model, creating a realistic propagation model. For brevity, in text is written only Ricean or Rayleigh, instead of "two-ray ground with Ricean fading" and "two-ray ground with Rayleigh fading".

Precision, recall and F-measure of a rule in a simulation setup are calculated using Equations 2.3, 2.4 and 2.6. They are jointly calculated for all nodes/edges in a network. For instance, precision of a rule for articulation point detection is calculated as:

$$\frac{\sum_{l=1}^{samples}\sum_{j=1}^{decisions}\sum_{i=1}^{nodes} TP(i,j,l)}{\sum_{l=1}^{samples}\sum_{j=1}^{decisions}\sum_{i=1}^{nodes} TP(i,j,l) + \sum_{l=1}^{samples}\sum_{j=1}^{decisions}\sum_{i=1}^{nodes} FP(i,j,l)}. \quad (9.1)$$

So, the value of precision is jointly calculated for all decision rounds made in a network ($i = 1..decisions$) over all nodes ($j = 1..nodes$) over all experiments/simulations ($l = 1..samples$).

The reward metric helps us quantify the relative position of a bridge or an articulation point within a network, since for applications of the detection approach it is not the same whether it successfully detects bridges and articulation points only at network periphery or it is capable of detecting them in central parts of a network. For a given simulation setup, voting rule and value of parameter k, the average value of reward is calculated over all decisions that are made on all bridges/articulation points (see page 29).

Typically, recall and reward metrics are correlated: e.g., an approach that detects more bridges has higher recall, and its average reward should also grow. However, there are exceptions as it will be demonstrated later. On some occasions, recall of majority rule for bridge detection is higher than of the trusted rule, but its reward is smaller. In that scenario, majority rule failed to detect bridges in central parts of network, so despite larger number of bridges that are detected, its reward was limited.

For brevity, the reward metric is presented in this section only when there exists a notable difference between it and the recall. Detailed graphs of reward for all analyzed scenarios can be found in Appendix D.

Plots are used as one of technical means for presentation of the precision, recall, and F-measure. Each of the points in a plot is jointly calculated from all decisions as it was shown on example of precision in Equation 9.1.

Still, the possible differences between individual samples within a simulation scenario cannot be ignored. In order to assess these differences, confidence intervals have been calculated. Instead of iterating through all samples of a given scenario as in Equation 9.1, precision is calculated for each individual sample and the average value with 95% confidence intervals is calculated from this set. Table 9.1 provides an excerpt of this evaluation. It can be seen that confidence intervals are tight and that there are no notable deviations from the average value. Interesting property of the approach is that recall is considerably less variable than the precision, regardless of value, rule, or detection purpose.

The link acceptance thresholds were set to $t = 0.1$ and to 0.316. The link acceptance threshold of $t = 0.1$ (equivalent to ETX=100) demonstrates network behavior in extreme conditions when network protocols are forced to use links even if they are of very low quality. Such scenarios may occur in sensor networks after long periods of activity, after some nodes have failed without replacement. The acceptance threshold of 0.1 may be inappropriate for some applications if they cannot tolerate so frequent packet and message losses. Thus, the tests are repeated for a more strict link acceptance threshold $t = 0.316$ (equivalent to ETX=10).

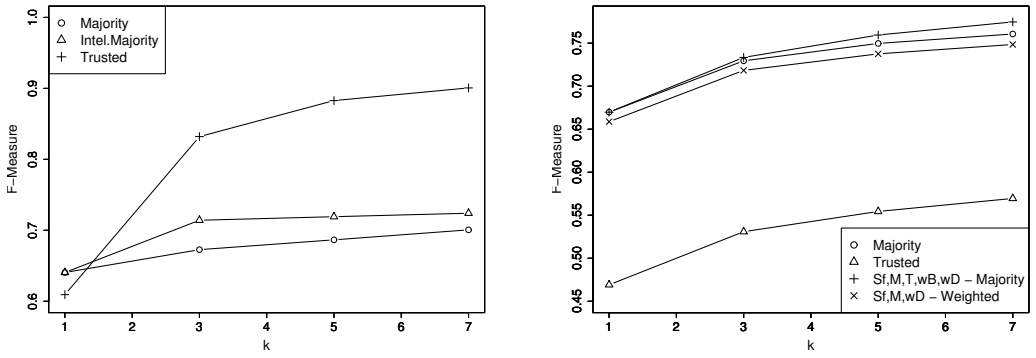
Bridge detection

Rule name	Precision	Recall
Unanimous	0.904154 [0.8809450, 0.92736297]	0.805756 [0.79398500, 0.81752699]
Single-for	0.563678 [0.5205635, 0.60679241]	0.996890 [0.99527644, 0.99850355]
Majority	0.753996 [0.7100893, 0.79790265]	0.979602 [0.97410130, 0.98510269]
Trusted	0.816026 [0.7735757, 0.85847630]	0.933764 [0.92894478, 0.93858322]

Articulation point detection

Rule name	Precision	Recall
Unanimous	0.904190 [0.88635794, 0.92202205]	0.554758 [0.54218586, 0.56733014]
Single-for	0.408792 [0.39374432, 0.42383967]	0.971802 [0.96851312, 0.97509087]
Majority	0.682968 [0.65743871, 0.70849729]	0.851512 [0.84208421, 0.86093978]
Trusted	0.551382 [0.52528226, 0.57748174]	0.853888 [0.84148349, 0.86629251]

Table 9.1.: 95% confidence intervals of precision and recall of selected rules. Simulation setup: NPART/Berlin node placement, Ricean propagation, link acceptance threshold $t = 0.1$.



(a) Bridge detection.

(b) Articulation point detection.

Figure 9.1.: Effects of parameter k on detection accuracy of voting rules (Motelab experiments example).

The rules can be applied on various sizes k of the set of latest decisions. In this chapter are presented the evaluation results for $k = 5$. It has been noticed in all evaluation scenarios that voting rules reach the saturation point of F-measure rather early, so already at $k = 5$ the rules are close to their limits. This characteristics can be clearly seen in Figure 9.1 which shows evaluation results of a subset of results obtained in the Motelab experiments. There exists evident increase in accuracy if k is increased from one to five, but if k grows above five, the improvements are minor. The complete evaluation results, that include other values of parameter k are presented in Appendix D.

9.2. Issues of Proactive Topology Management for Bridge and Articulation Point Detection

The distributed approach is not the only possibility for bridge and articulation point detection. Section 3.1 (page 32) presents numerous contributions in literature which

propose proactive collection of topology data or reuse of the topology data of a proactive routing protocol in order to run Tarjan's DFS detection algorithm [158] on the collected topology. The crucial assumption in the literature is that proactive topology dissemination provides sufficiently good topology information so that it can be used for bridge and articulation point detection.

The simulation evaluation presented in this section demonstrates that belief in accurate topology knowledge was created mostly by the imprecise simulator configuration: the exclusive use of the two-ray ground propagation model. In reality and in simulation with improved propagation models, it is experienced that network topology view at individual nodes can be different from the accurate network topology, resulting in faulty decisions in bridge and articulation point detection.

Two main reasons for imprecise topology view at individual nodes have been identified:

- **Impossibility of precise detection of communication link existence.** As explained in Section 3.3, heartbeat messages are commonly used for link existence detection in proactive topology management protocols. The analysis performed in Chapter 4 demonstrated the issues of link detection if communication channel is unreliable. It was shown that even with the optimal link detector parameters, errors in link detection process are notable.

  A false negative in link detection may exhibit itself in occurrence of false positives in context of bridge and articulation point detection – the topology seems to be sparser than it is and some links may be declared as bridges and some nodes as articulation points although they are not. A false positives in link detection may result in false negatives for bridge and articulation point detection (since the perceived topology seems to be denser than it actually is).

- **Issues of accurate and timely dissemination of the local topology information through the network.** The second component of proactive topology management is the dissemination of local topological information through the network. It also faces challenges in presence of channel fading. Flooding is one of simplest yet widely used approaches in WMNs for dissemination of local data. Some studies [125] [174] consider it highly redundant with huge unnecessary overhead and propose various means of redundancy reduction [129] [174]. Such claims are true in very dense networks or if exceptional link quality is assumed, but in reality neither of these claims is true.

  As the analysis in Chapter 7 shows, real networks are neither very dense as there exists considerable number of bridges, nor all links in them are of high quality (in Berlin, 5.3% of bridges have packet delivery ratio less than 0.1 and 22.6% of them less than 0.5). Correlated packet losses and bridge presence may seriously affect flooding's ability to deliver local data to all nodes in the network.

  The coverage of flooding was analysed in [120]. It was demonstrated that the flooding does not reach large parts of the network. Successive retries improve its delivery ratio but they also increase the delay in message delivery, in particular for nodes that are distant from a node that reports the changes.

  The effects of the dissemination issues may be reduced through higher frequency of topology updates. However, each dissemination cycle is costly in terms of scarce

resources such as node energy and channel bandwidth so the update frequency increase must be limited in order to prevent bandwidth starvation of other services in the network, and node failures due to energy depletion.

Interplay between issues in local link detection and global topology dissemination may be particularly devastating: if a false information reaches wide audience and it is followed by correct but short-lived dissemination cycles, the erroneous global topology views at multiple nodes in the network exist for long time periods. Thus, there exist differences between topology *perceived* by a node and *accurate* topology that is seen by an omniscient observer.

In order to assess the effects of unreliable communication channel on proactive topology management protocols and their impact on bridge and articulation point detection approaches which were proposed in the literature, ns2 simulations of OLSR routing protocol was performed. The parameter TC_REDUNDANCY of OLSR was set to 2, so that complete local link information is shared throughout the network. Other protocol parameters have been set as defined in its RFC [59].

Traffic is deliberately excluded from simulations – in presence of traffic, the topology management issues can only escalate due to more frequent collisions between user-generated traffic packets and topology dissemination packets. Two sets of referent topologies are used for simulation with link acceptance thresholds of 0.1 and 0.316. Simulations are performed for two-ray ground, Rayleigh, and Ricean fading propagation models.

After warm-up phase that lasts for 200 seconds, every 30 seconds in the next 360 seconds, nodes calculate articulation points and bridges in the network from the topology information that is available to them. This information is then compared with the actual bridges and articulation points, calculated from precise topology (known only to the omniscient observer) and precision, recall and F-measure are derived.

The decisions are divided in two classes:

- **Decisions at node level** (local decisions) include only decisions of a node on links incident to it and whether the node declares itself as an articulation point.
- **Decisions at network level** (global decisions) are derived by a node for all links and nodes of the network.

Tables 9.2 and 9.3 show the simulation outcomes. If two-ray ground model is exclusively used for signal propagation, almost complete topology is available at each node, as it was assumed in literature [57] [80] [168]. It is not surprising that the precision and recall are exceptional, both for local and global decisions. The two-ray ground propagation model completely eliminates the false positives in link detection process and it can only result in false negatives (omission of some links from the topology). Thus, detection recall is always one (if node/edge exists in perceived topology) while precision may be slightly lower.

If fading is introduced, the issues of proactive topology detection and dissemination become clearly visible: the perception of topology at individual nodes is significantly different from the topology visible to the omniscient observer. Table 9.4 presents the average ratio between number of perceived edges at a node and edges in the accurate topology. As it can be seen, even with false positives in link detection that are included

Simulation scenario	Local Precision	Local Recall	Global Precision	Global Recall
Uniform, two-ray ground	1	1	0.9516	1
NPART/Berlin, two-ray ground	0.9992	1	0.9663	1
NPART/Leipzig, two-ray ground	0.9996	1	0.9772	1
NPART/Berlin, Ricean(7), $t = 0.1$	0.602	0.79	0.14	0.258
NPART/Berlin, Ricean(7), $t = 0.316$	0.903	0.397	0.307	0.187
NPART/Berlin, Rayleigh, $t = 0.1$	0.348	0.565	0.054	0.035
NPART/Berlin, Rayleigh, $t = 0.316$	0.824	0.31	0.1832	0.028
NPART/Leipzig, Ricean(7), $t = 0.1$	0.822	0.775	0.247	0.335
NPART/Leipzig, Ricean(7), $t = 0.316$	0.951	0.495	0.347	0.256
NPART/Leipzig, Rayleigh, $t = 0.1$	0.504	0.678	0.025	0.084
NPART/Leipzig, Rayleigh, $t = 0.316$	0.897	0.324	0.082	0.073

Table 9.2.: Precision and recall for bridge detection under proactive topology management.

Simulation scenario	Local Precision	Local Recall	Global Precision	Global Recall
Uniform, two-ray ground	1	1	0.9653	0.9737
NPART/Berlin, two-ray ground	1	0.9972	0.9743	0.9254
NPART/Leipzig, two-ray ground	1	0.9987	0.9802	0.9473
NPART/Berlin, Ricean(7), $t = 0.1$	0.671	0.79	0.254	0.249
NPART/Berlin, Ricean(7), $t = 0.316$	0.931	0.461	0.504	0.208
NPART/Berlin, Rayleigh, $t = 0.1$	0.404	0.486	0.114	0.037
NPART/Berlin, Rayleigh, $t = 0.316$	0.807	0.286	0.384	0.04
NPART/Leipzig, Ricean(7), $t = 0.1$	0.87	0.771	0.412	0.336
NPART/Leipzig, Ricean(7), $t = 0.316$	0.97	0.57	0.542	0.292
NPART/Leipzig, Rayleigh, $t = 0.1$	0.713	0.634	0.063	0.072
NPART/Leipzig, Rayleigh, $t = 0.316$	0.958	0.342	0.162	0.075

Table 9.3.: Precision and recall for articulation point detection under proactive topology management.

	$t = 0.1$	$t = 0.316$
NPART Berlin, Rayleigh	0.234	0.361
NPART Berlin, Ricean (7)	0.627	0.774
NPART Berlin, two-ray ground	0.979	
NPART Leipzig, Rayleigh	0.484	0.764
NPART Leipzig, Ricean (7)	0.789	0.98
NPART Leipzig, two-ray ground	0.986	

Table 9.4.: The average ratio of number of captured edges (including false positives) to accurate number of edges by OLSR protocol.

in samples with fading, topology dissemination under fading captures only a portion of the total number of edges. This trend is particularly notable for scenarios with Rayleigh fading.

Obviously, the introduction of Ricean and Rayleigh fading influences the accuracy and availability of topology information, and the quality of detection results is reduced. Dissemination of local topology knowledge through network is particularly difficult. Nodes receive infrequent and possibly inaccurate updates from distant nodes which drastically reduces the availability of topology information on distant parts of the network resulting in very inaccurate biconnectivity testing decisions at the network level.

As explained in Chapter 4, HLDs used by OLSR are prone to create false positives in link detection for higher link acceptance thresholds. More false positive links results in higher false negative ratio in bridge and articulation point detection and reduction in recall (for $t = 0.316$). Since false negatives are less probable, higher link acceptance threshold results in higher precision.

Lower link acceptance threshold ($t = 0.1$) reduces number of false positives and increases false negatives in link detection, which results in reduction of precision and improvement in recall (relative to detection outcomes if $t = 0.316$). These processes are easily observable in Tables 9.2 and 9.3.

Rayleigh fading:

Local decisions: The accuracy of detection is far from the impressive achievements obtained with the unrealistic two-ray ground model. F-measure of bridge detection in NPART/Berlin topology is around 0.48, while F-measure of articulation point detection is between 0.41 and 0.43. Detection in NPART/Leipzig topologies achieves a bit better results, in particular for the lower link acceptance threshold. Results are of practical use with F-measure of 0.578 for bridge detection but far from idealized values observed with two-ray ground propagation model. If link acceptance threshold is increased, recall drops sharply and reduces F-measure to 0.476.

Global decisions: Large parts of the global network topology are unknown at decision-making nodes resulting in numerous false negatives. If a decision-making node is unaware of a node/edge in topology it cannot mark it as an articulation point/bridge, so it is interpreted as a false negative. The known parts of topology are perceived as sparse and partially disconnected. The first issue reduces recall and the second precision of the detection, so the obtained accuracy may be very low – for NPART/Berlin topology and $t = 0.1$, F-measure of bridge detection is only 0.042, and the highest obtained F-measure for Rayleigh fading and NPART/Berlin topologies is 0.0723 (for articulation point detection and $t = 0.316$). Results in NPART/Leipzig topologies are a bit better, but the improvement is still insufficient: F-measure of bridge detection is up to 0.77, and for articulation point detection it is up to 0.102.

Ricean fading:

Ricean fading generally provides better communication in the network due to smaller variations in signal strength. This results in better accuracy of the detection than in Rayleigh-fading scenarios.

Local decisions: Quality of local decisions has considerably improved, in particular for the lower link acceptance threshold. Bridge detection in NPART/Berlin topologies has high F-measure of 0.683 for $t = 0.1$ which is increase of 58% to the scenario with the same topology type, link acceptance threshold and Rayleigh fading. In NPART/Leipzig topologies improvements are also considerable – F-measure is up to 0.82 which is an

improvement of 38.6% over Rayleigh scenarios. Articulation point detection is particularly accurate in NPART/Leipzig topologies for $t = 0.1$ where its F-measure is 0.871, but unfortunately detection is not particularly stable. If link acceptance threshold is increased to $t = 0.316$, F-measure of articulation point detection falls to 0.71 and of bridge detection to 0.65. Important characteristics of the proactive approach is its excellent precision but low recall for this higher link acceptance threshold.

Global decisions: The quality of perceived network topology at nodes is improved and larger part of network is captured. For biconnectivity testing this results in better accuracy than for Rayleigh fading, but the global detection results are still of very limited value.

In NPART/Berlin scenarios, F-measure of global bridge detection is between 0.18 and 0.23. So, for the more accurate case, only 18.7% of bridges are detected and out of all bridge-markings only 30.7% are correct. Such uncertainty would confuse most (if not all) of the higher-layer algorithms that are to utilize this information, such are the approaches for reinforcement of bridges by mobile nodes [80]. In NPART/Leipzig topologies, results are better, but the F-measure does not even reach 0.3.

Articulation point detection results are also better in both topologies compared with Rayleigh scenarios, with higher improvement for NPART/Leipzig, where F-measure reaches 0.379 for $t = 0.316$, but even in this best case more than two thirds of articulation points remain undiscovered.

Summary

The approach which is described in related work [57] [80] [168] was directly applied in this section: topology information is taken from a proactive routing protocol and it is used for biconnectivity testing. Related work assumed the two-ray ground propagation model. With such propagation model, OLSR routing protocol indeed provides excellent quality of captured topologies, and consequently the detection results are also exceptional. Realistic channel modeling introduces topology inaccuracies that cause numerous faulty decisions in bridge and articulation point detection process.

It can be concluded that combination of difficulties encountered in reality in link detection and topology dissemination processes obstructs the fulfillment of high requirements for topology accuracy for 2-connectivity testing and invalidates the assumed absolute correctness of proactive approaches [57] [80] [168]. The extensive overhead that is produced for the topology dissemination brings meager and highly varying (highly dependant on the link acceptance threshold) detection results at node level and practically unusable results at the network level.

The local accuracy of biconnectivity testing based on the proactive topology management approaches may be improved by more careful selection of HLD parameters, but the issues of local link information dissemination remain. In practice (community networks presented in Chapter 7), the issues of link detection have been partially resolved through increase in heartbeat frequency and the observation period of a link is prolonged, making such protocol more stable in static but rather difficult to use in mobile and semi-mobile networks.

The major concern in open communities is the protocol robustness, while provision of optimal parameter selection is left to researchers. Unfortunately, some models in research are still based on unrealistic assumptions, completely ignoring the practical issues. For instance, in [125] optimal parameters for proactive protocols are derived for the unrealistic two-ray ground propagation model. So, the heartbeat frequency is

reduced, which indeed reduces communication overhead of the protocol, but the authors did not perform additional tests to verify if a protocol with such configuration can function at all if deployed in environment with signal fading.

Note on OLSR, proactive topology management and detection approaches described in related work. The parameters of the OLSR protocol have been taken from its RFC as it is envisioned for use by their authors. The analysis of heartbeat link detectors in Chapter 4 indicates the suboptimal default parameter combination of OLSR, but the parameters have not been changed in this evaluation for two reasons:

- The outcome of simulations is so surprising and reduction in accuracy so notable that it would not be clear if such behavior of OLSR is consequence of the new parameter set or of topology management issues introduced by fading.

- The evaluation results as they are presented confirm the finding of Chapter 4 on issues of improperly parameterized HLDs.

Finally, this study does not disqualify the proactive routing protocols. Amount of topology information that is needed for routing is considerably lower than for biconnectivity testing. Also, even if a suboptimal route has been initially envisioned by a node that initiates the traffic, packets may get rerouted by nodes which are closer to destination and have more accurate information on current network state at their disposal.

9.3. Implementation and Evaluation of DIBADAWN in a Wireless Sensor Testbed

The simulations are a useful tool for development of protocols and tuning of various parameters but the ultimate verification test of every network protocol is a real network. It exposes the protocol to various limitations that are not supported by simulator or that have been inadvertently omitted from it. Experiments may also expose weaknesses and hidden assumptions of protocols such as high memory overhead, intensive computation, algorithm and protocol instabilities caused by environmental changes, etc.

This section explains experiences from deployment and execution of the DIBADAWN approach in a wireless testbed. Four tasks have been necessary for verification of the approach:

- **Finding a suitable testbed:** due to high development and maintenance costs of establishing a new testbed, it was decided to use one of existing testbeds. Out of numerous possibilities, Motelab testbed at Harvard University [12] was selected due to its size, accessibility, deployment environment and control facilities.

- **Implementation of the algorithm:** the DIBADAWN algorithm has been customized for deployment on TMoteSky nodes that are used in Motelab testbed. The nodes operate the TinyOS event-based operating environment. TinyOS applications are written in nesC [77], a dialect of the C programming language optimized for the memory and processing limitations of wireless sensor nodes.

- **Setup of the testbed:** Adequate setup had to be determined so that it creates topologies with sufficient number of bridges and articulation points located both in central parts and periphery of the network.

- **Data collection, processing and evaluation of results:** data is collected from nodes and sent to a central repository server. Low processing power of nodes and their confined memory limits data that can be collected and sent from them. Motelab nodes send data from the first phase of DIBADAWN (building a tree and detection of cross-edges) to the server. This information is used later by the omniscient observer to determine referent topology. Edge markings from the second phase of DIBADAWN are sent to the server for evaluation of algorithm's accuracy. Postprocessing of collected data is performed offline in a custom Java application, derived from Jist/swans simulator code.

9.3.1. Overview of Existing Testbeds and Selection Criteria

The testbed in which the algorithm is to be deployed and verified has to be of considerable size. Small testbeds, consisting of several nodes cannot create sufficiently complex topologies, the algorithm's accuracy would be exceptionally high and verification results would be doubtful. Option of building a large testbed at the Institute for Informatics has been rejected due to high financial costs and resource overhead required for its setup and maintenance. Instead, it was decided to use one of numerous existing wireless testbeds, used for scientific purposes. They vary in size, capabilities, accessability, and support. The following criteria has been applied in selection process:

- **Testbed size:** large testbeds are preferable. Testbeds with less than 20 nodes have not been considered. As explained later, in order to produce adequate[3] topologies some nodes in the selected testbed had to be excluded from the evaluation process, further reducing its effective size.

- **Topology and environment:** testbeds with regular node placement patterns (e.g., grids or chains) are undesirable because their topologies are not diverse enough. Signal propagation in the testbed should be combination of line-of-sight situations and propagation through obstacles. Three-dimensional node placement (i.e., building) is desirable while open-space and large hall environments are undesirable.

- **Remote access and control:** The access rights to the testbed during experiments must be exclusive, in order to have controllable experiments. Setup of experiments, their execution and data collection must be performed without in-site human assistance. The implementation and testing is a lengthy process and any reliance on testbed's maintenance crew can considerably prolong it.

Nodes in a testbed have two interfaces: one wireless for actual tests, and one wired for control, upload of software and download of collected data. It is necessary that a node

[3]Topology with significant number of bridges and articulation points, but also with dense network sections.

Testbed	Motelab [172]	Kansei [69]	TWIST [84]
Declared Nodes	190 TMote Sky	210 Pairs of XSM and XSS	102 TMote Sky, 102 eyesIFX
Active Nodes	121	77	unknown
Remote Access	++++	+++	++
Topology and Prop. Environment	+	-	+

Table 9.5.: Overview of testbed characteristics.

Node	TMote Sky	XSM - Extreme Scale Mote	XSS - Extreme Scale Stargate	eyesIFX
Processor Frequency	TI MSP430F1611 8MHz	Atmel AT-mega128L 4MHz	XScale PXA55 400MHz	TI MSP430F1611 8MHz
Memory	48kB ROM, 10 kB RAM, 1MB Flash	128kB ROM, 4kB RAM, 512kB Flash	32MB ROM, 64MB RAM	48kB ROM, 10 kB RAM, 512kB Flash
Radio Frequency	Chipcon CC2420, 2.4GHz	Chipcon CC1000, 916MHz	SMC2532W-B, 802.11b WLAN	TDA5250, 870MHz
Development	TinyOS	TinyOS	Linux	TinyOS

Table 9.6.: Characteristics of nodes.

possess the wired interface, since wireless transmission of control data and collected results would interfere with the measurements.

It is common that there exists at least one more supernode in the testbed that actuates and instruments nodes, controls software that is executed on nodes, and acts as central repository for data collection. It should be possible to upload an application for nodes to it, and the rest of node software update process is automatized by the supernode. At the end of execution of an experiment, the data collected from nodes can be accessed and downloaded from it.

Not all evaluated testbeds implement these control mechanisms in full extent and some that did, imposed special constrains on them. For instance, Sensornets[4] project from University of California at Berkeley (UCB) developed three testbeds, using different technologies, but it is accessible only to UCB affiliates.

Table 9.5 lists characteristics of evaluated testbeds and Table 9.6 of nodes that are used in them. The data has been collected at the beginning of year 2008 during the search for a suitable testbed.

Kansei testbed [69] from University of Utah provides two node types: processing-weak XSM nodes and XSS as nodes with highest processing potential of all evaluated. It supposed to be the largest testbed of all, but out of 210 nominal nodes, only 77 were functional. The information which is available about this testbed is confusing. The data in Table 9.5 originates from project's homepage, while its online help system claims presence of 112 XSM/XSS nodes, additional 432 Tmote nodes and 104 Intel Imote2 nodes. The main issue of the testbed is that nodes form a dense grid in a large hall – both topology and propagation environment are regular.

TWIST (TKN Wireless Indoor Sensor network Testbed) [84] is located on Technical University (TU) Berlin. The network is considerable in size, has two types of nodes and supports TinyOS. Unfortunately, at the moment of testbed selection, control interfaces were still under development (or at least not accessible for general public) so deeper understanding of its capabilities and overview of topology has not been possible.

[4] https://www.millennium.berkeley.edu/sensornets/

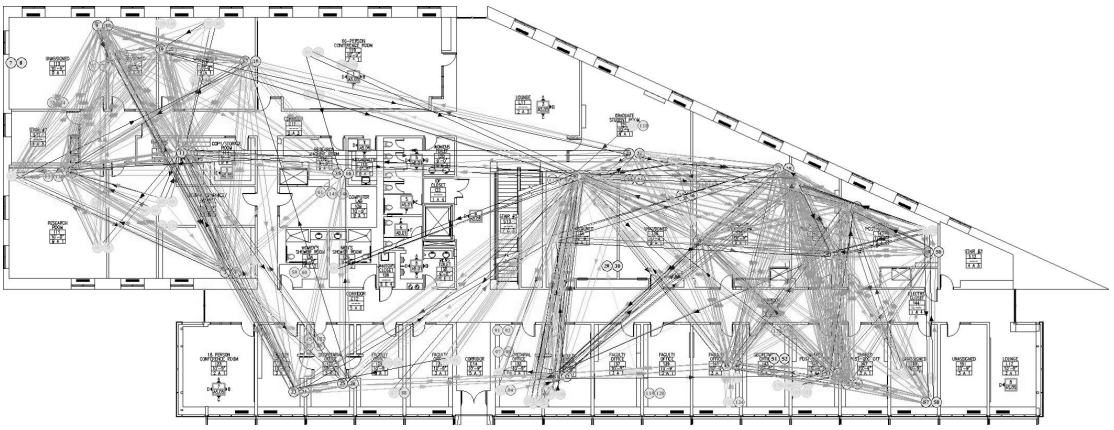


Figure 9.2.: A sample Motelab topology, first floor of the building.

Emulab[5] offers interesting combination of simulation, emulation and real networking. It has 25 wireless nodes in an unclear formation. Similar as in Kansei testbed, information offered through website is unclear and often contradicting: on some web pages are photographs showing regular node placement, while other pages claim semi-random placement in an office building. Due to uncertainty on its structure and small size of the network, this testbed has not been evaluated in detail like others.

Motelab [12] [172] is a testbed located at the Harvard university. It nominally consists of 190 nodes, located on three floors of an office building. Most of nodes can communicate only within a floor. A small subset of fortunately placed nodes provide connectivity between floors. Figure 9.2 shows snapshot of topology from the first floor of building where Motelab is located. The propagation environment is dynamic and heterogenous since humans working in it create constant and notable changes in the environment. It is sufficient that several doors are opened or closed to get different conditions in the testbed. People also act as signal reflectors and scatterers. TMoteSky nodes use ISM 2.4GHz band, that is shared with other devices that use the same band, such as WLAN, so the effects of interference with wireless terminals that do not belong to the testbed are also present in the experiments.

The testbed is open without fees for all researchers. Its usage is allocated in slots of 30 minutes. Usability of its management software is excellent: account management, slot allocation, upload and download of data are performed through a web-interface. Its biggest issue is the mandatory use of TinyOS version 1.x [6].

Since it offers best compromise between the three selection criteria, Motelab testbed has been chosen for experiments.

9.3.2. TinyOS and TOSSIM

Nodes in Motelab use TinyOS. TinyOS is an event based operating environment designed for use with embedded networked sensors. Although its name suggests otherwise, it is not a traditional operating system, but a component oriented, programming framework.

[5] www.emulab.net

[6] Testbed is TinyOS 2.x compliant since February 2009.

It supports rudimentary concurrency that is required in embedded networked systems with limited hardware resources.

TinyOS programs are built of components. Components are connected through interfaces. Interfaces and components abstract both hardware (e.g., packet communication, sensing, actuation) and software resources (e.g., routing, data storage). The interface of a component is the functional specification of a service provided by the component to its users. Interface is obligatory for both sides: the service provider must implement all specified commands and the user must implement the event handlers defined in the interface. Specification of non-functional requirements is not supported by the TinyOS.

Components interact through commands and events. A command is a request to a component to perform a service. An event is component's confirmation that the service has been performed. Commands and events are not blocking. They provide the concurrent behavior of TinyOS, in particular if a component provides an interface to a hardware resource. For instance, reading of a sensor value may be a lengthy operation, so the main program invokes reading command and continues its execution until an event originating from hardware is triggered, indicating the completion of read operation.

Component can also specify tasks – functions that are scheduled for execution at a later time. Tasks improve responsiveness of programs by postponing time and resource intensive tasks for later execution. The TinyOS uses a non-preemptive, FIFO scheduler for management of tasks. Hardware interrupts are treated separately so they have preemptive behavior.

Components are written in nesC, scale-down derivative of C. In nesC it is not possible to use function pointers or global variables. Components can be only statically allocated (at compile time). The nesC compiler transforms nesC code to a standard C code that can be afterwards compiled to executable code using the tools for a target hardware platform.

TinyOS component model allows easy switchover between real, hardware resources and its software emulators. This characteristic is utilized by TOSSIM – a simulator for TinyOS-based wireless sensor networks. The hardware interrupts are emulated by simulator events that are forwarded through simulator's event queue to target components. The official claim of TOSSIM's developers is that except for these hardware components, remainder of code stays unchanged. However, TOSSIM's interrupts are not preemptive which in turn means that all tasks are atomic in simulation. If implementation depends on atomicity, this can lead to serious and difficult to understand problems once a protocol is deployed in real networks.

TOSSIM's abstraction of network communication is probably its weakest point: the network is a directed graph, where each edge has a bit error probability. The error probability of a link can be randomized or loaded from measurement studies but there does not exist explicit support for correlated error rates between links. The main purpose of TOSSIM is to accelerate development process and enable easier debugging of TinyOS applications so such network model is usually sufficient – the developed code is eventually deployed to a wireless network where it is tested under real conditions.

9.3.3. Testbed Setup and Data Collection

Motelab testbed is managed through a Web interface. It is used to upload the software and download the experiment results. Experimental data is saved in a local database.

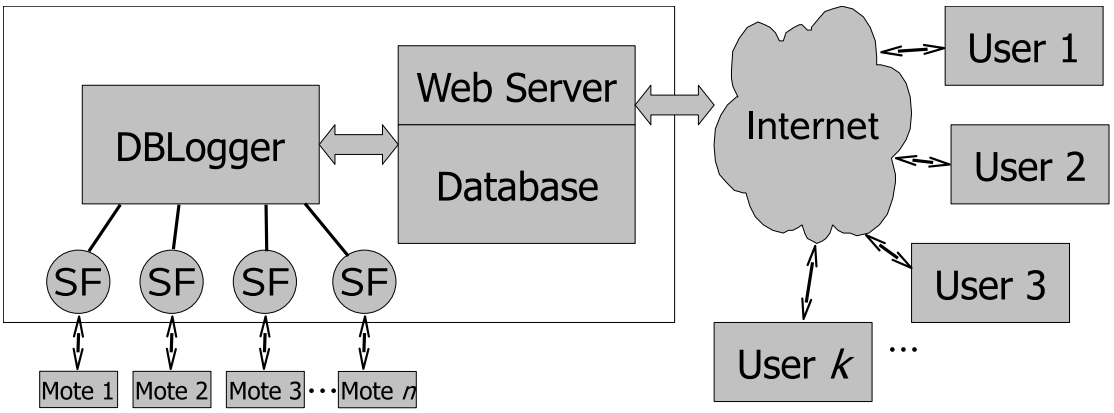


Figure 9.3.: Motelab infrastructure.

Multiple users are supported, as it can be seen in Figure 9.3. A user has exclusive access to the testbed during its time slot.

Nodes in the testbed use only their radios for experiments. Limited node memory prevents storing of measurements on a node during an experiment. Instead, the measurements are exported during experiments to the database through serial port that exists on every node. A dedicated component of TinyOS – the "Serial Forwarder" (SF in Figure 9.3) acts as a gateway between the serial port of a node and a TCP/IP port of the central server. DBLogger is a software component at server which collects data from TCP/IP ports and writes it to the database.

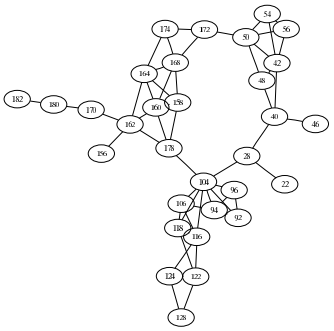
During message write to the serial port, a node is unable to send data over radio. Fortunately this issue does not affect the reception of messages – the radio operates autonomously and once the packet is ready, the radio generates an interrupt to the processor.

This issue required careful selection of data that is sampled by a node for later analysis – intensive data collection combined with computational weakness of nodes may impair their capability to timely execute algorithm steps. This is particularly dangerous for later analysis of algorithm's efficiency and detection accuracy – such events may be misinterpreted as algorithm's high computational overhead, or even worse as its incapability of accurate bridge and articulation point detection. To offset these effects, processing time per tree level in the second phase of the algorithm has been increased to allow execution of both native detection code and data collection facilities.

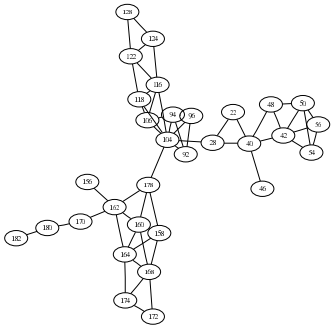
After the technical issues of data collection have been resolved, appropriate topology had to be selected. This includes selection of a node subset that operates the algorithm and setting their transmission power.

First step was removal of pairs of nodes. As it can be seen in Figure 9.2 nodes in the testbed are deployed in pairs, probably to reduce the wiring (needed for serial port communication) and to increase network's resiliency to node failures. Their presence eliminates bridges and articulation points in the network. To circumvent this issue, only one node in each pair has been selected for use in experiments.

Ideally, the topology in which DIBADAWN is tested has combination of dense and sparse parts, with bridges located both on network periphery and in its central parts. In order to achieve this, topology probes have been taken from the testbed for differ-



(a) Topology that increases DIBADAWN accuracy.



(b) Topology chosen for testing – there is no advantage for DIBADAWN accuracy.

Figure 9.4.: Various topology types encountered in Motelab in preparation of experiments.

Edges	Bridges	Ratio
50.9811 [50.0118, 51.9503]	6.6603 [6.1592, 7.1615]	0.1306
Nodes	Art. points	Ratio
31.6981 [31.3174, 32.0787]	7.0188 [6.7060, 7.3316]	0.22

Table 9.7.: Properties of Motelab topologies for link acceptance threshold $t = 0.1$.

ent transmission power of nodes. TinyOS allows easy manipulation of it through constant CC2420_DEF_RFPOWER that can be set to values between one (equivalent to -25dBm) when only node pairs can communicate, and 31 (equal to 0 dBm) when testbed is extremely dense: without pair removal, nodes have between 20 and 30 neighbors. Through trials and errors it has been determined that if CC2420_DEF_RFPOWER is set to eight, the satisfying topology is obtained.

If node pairs are eliminated and CC2420_DEF_RFPOWER is set to eight, the topology has few bridges and articulation points, almost exclusively on network outskirts (Figure 9.4(a)). Testing in such topology would produce favorable conditions for the detection algorithm: it is rather easy to detect articulation points and bridges incident to pendant nodes so algorithm's precision and recall rates would be unrealistically high. To avoid this unfairness, additional nodes have been removed so that bridges and articulation points are placed both on network outskirts and in its central parts, creating topologies similar to one shown in Figure 9.4(b).

The algorithm tests lasted for several days and due to unknown reasons some nodes were occasionally unavailable, resulting in varying topologies but the overall structure of the testbed remained qualitatively the same with bridges and articulation points located both in central part of network and on its periphery.

Table 9.7 shows characteristics of Motelab topologies used for the experiments. It can be seen that the studied Motelab topologies have similar properties as the topologies of Berlin's and Leipzig's open networks.

9.3.4. Implementation of the DIBADAWN Algorithm and the Evaluation Procedure

In order to evaluate the algorithm, two parallel processes had to be supported. One implements the detection of bridges and articulation points. The edge markings are collected and send to server for postprocessing. The second process gathers information from the forward phase of DIBADAWN execution. This is necessary for later evaluation of results – the reference topology of network has to be determined in order to calculate precision, recall, F-measure and reward metrics. Reference topology detection had to be performed simultaneously with the execution of DIBADAWN because of the unpredictable behavior of environment where Motelab is located.

The reference topology is created using the following steps: node set of the reference (omniscient) topology comprises all nodes that have participated in DIBADAWN. Messages that were exchanged in the forward phase of DIBADAWN were preserved by sending and receiving nodes. If a message was successfully received by a node, a link to the sender of the message is added. Number of messages that successfully traversed link is counted and divided by the number of transmissions that were initiated on that link. Thus, the link traversal probabilities are obtained. Finally, the topology is pruned. Only links which exceed the acceptance threshold remain in the reference topology. Tarjan's DFS is executed on this referent topology and information on cut edges and vertices is preserved for later accuracy evaluation and comparison with decision provided by DIBADAWN.

Each network node is emulated in a Java application so that it can process the markings which were delivered to, or sent by its counterpart in testbed (log files provide this information). The emulated node fully implements the decision making in DIBADAWN and the voting rules with the only difference that markings originate from log file, not from a network interface. Emulated node has a confusion matrix (Table 2.3, page 27) where it keeps track of accuracy of its decisions.

The timestamped markings are loaded from log files and forwarded to corresponding emulated nodes. An emulated node recreates individual algorithm executions, derives decisions, and executes voting rules. The output of rules is compared with the correct decisions (provided by omniscient observer) and confusion matrices of a node and links incident to it are updated. After processing of all markings, at all nodes, the confusion matrices are saved in database.

The implementation demonstrates that DIBADAWN is not demanding with regard to memory consumption and processing resources. Nodes that executed it have 10KB of memory that was shared between DIBADAWN, its voting rules (a set of voting rules was also implemented and deployed in testbed), and the serial forwarder. DIBADAWN was able to operate on 8MHz processors and the voting procedure does not produce considerable memory nor computational overhead.

9.3.5. Detection Results

This section evaluates the accuracy of DIBADAWN and voting rules from experiments in Motelab testbed, for different link acceptance thresholds ($t = 0.316$ and $t = 0.1$). Unfortunately, we were unable to find Motelab-ready implementation of a proactive routing protocol capable of complete-topology delivery. Instead, the results of evaluation

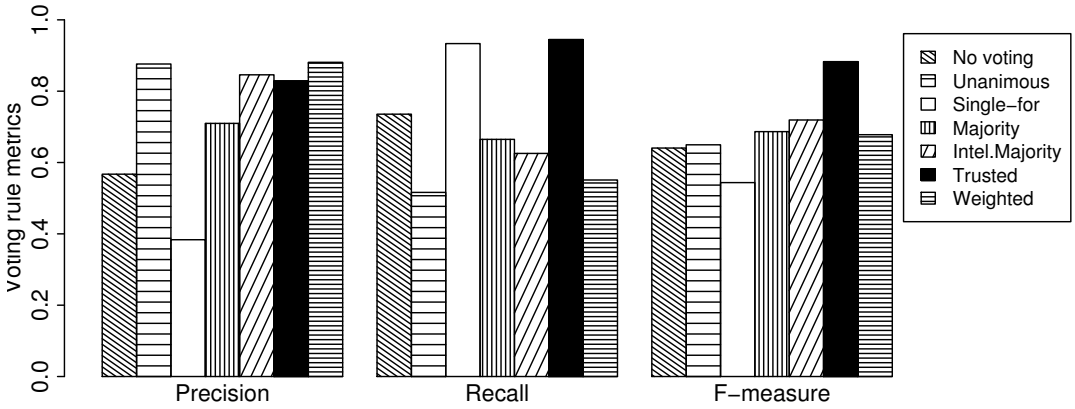


Figure 9.5.: Voting rules for bridge detection. Motelab experiments, link acceptance threshold $t = 0.1$.

of the OLSR approach (Section 9.2) are used as reference for comparison of Motelab experiment results. Due to considerable differences between the evaluation procedures, the behavior comparison is more important than the exact values of accuracy metrics.

Bridge Detection

The evaluation results are shown in Figures 9.5 and 9.6. The biggest disappointment in evaluation was the weighted voting rule, since it brings limited benefits despite its requirements for the learning phase. Its precision is almost identical to the unanimous rule with slightly improved recall, and F-measure.

The trusted and the intelligent majority rule have precision that is comparable to precision of the weighted rule and much higher recall and F-measure. Since trusted and intelligent majority rules do not require learning phase, their advantage over weighted is even greater. Thus, they are preferred over other rules unless precision is of utter importance when weighted is to be selected. The trusted rule performs better than the intelligent majority in Motelab experiments, regardless of the link acceptance threshold (in other scenarios, this is not always the case as it will be shown in Section 9.4.2).

Compared with the original (single-vote) decision making, introduced voting rules bring improvement and rise in F-measure, in particular the trusted voting rule which improves F-measure by 37.8% for $t = 0.1$ and 26% for $t = 0.316$. Other rules (except the Single-for rule) improve precision by increase in voter set size, but they simultaneously experience reduction in recall which results in similar levels of F-measure for higher link acceptance threshold ($t = 0.316$) and slight improvements for the lower (0.1) link acceptance threshold.

Behavior of DIBADAWN and voting rules is independent of the link acceptance threshold – the changes in accuracy are in range of several percent, which is an excellent characteristics of the proposed voting rules. Such behavior is opposite to the behavior of the proactive approaches. They are very sensitive to choice of link acceptance threshold and as it was shown on example of OLSR-based detection, lower link

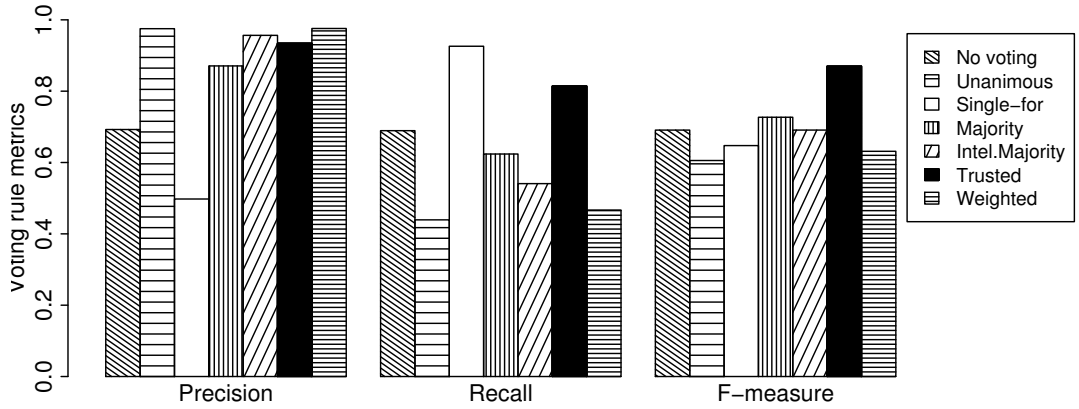


Figure 9.6.: Voting rules for bridge detection. Motelab experiments, link acceptance threshold $t = 0.316$.

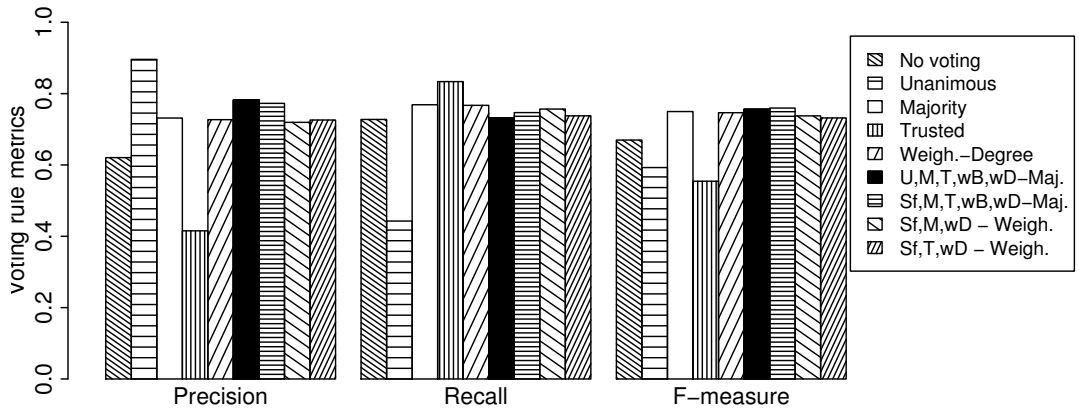


Figure 9.7.: Voting rules for articulation point detection. Motelab experiments, link acceptance threshold $t = 0.1$.

acceptance thresholds result in acceptable precision and good recall while the higher link acceptance thresholds result in excellent precision but and very low recall. The F-measure of rules in Motelab is higher than in OLSR based detection – F-measure of bridge detection in Motelab is 0.883 while the best OLSR achievement has F-measure of 0.82 and it goes as low as 0.43 for simulations with Rayleigh fading.

Articulation Point Detection

Experiment results are shown in Figures 9.7 for $t = 0.1$ and 9.8 for $t = 0.316$. The unanimous rule offers the best precision but its recall is by far the worst of all. It can be seen in the Appendix D that its recall is monotonously reducing with increase of parameter k. The reduction of recall cannot be compensated by increase in precision,

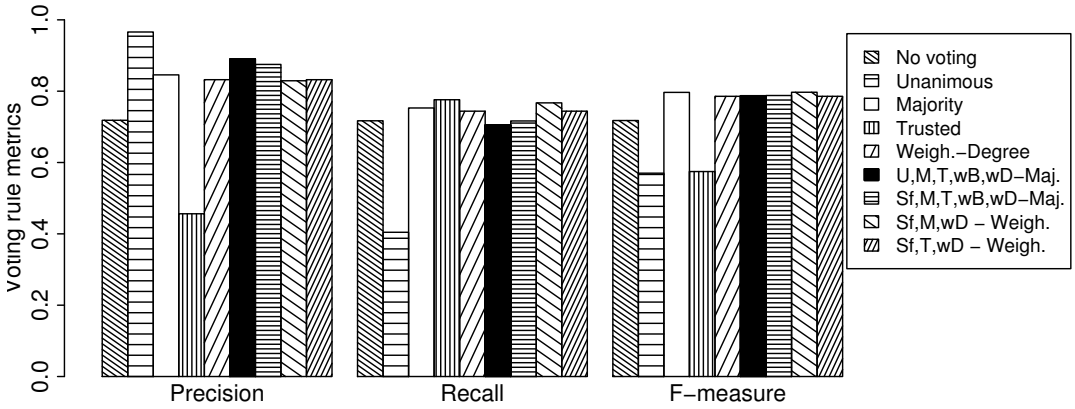


Figure 9.8.: Voting rules for articulation point detection. Motelab experiments, link acceptance threshold $t = 0.316$.

so its F-measure also reduces with growth of k (Figure D.6, page 229).

Other rules provide rather balanced accuracy, without outstanding winners: the difference in F-measure is up to 0.05. Common tradeoff between precision and recall is also present – rules with higher precision have smaller recall and vice versa.

Majority voting is particularly effective in the experiments, but in other evaluation scenarios (shown later) its accuracy is reduced, in particular its precision. The trusted rule has best recall of all rules that are evaluated in this section, but it also has the worst precision, that barely overcomes 0.4 mark. This is in contrast to trusted rule which was used for bridge detection, and which was clearly the best rule of all evaluated. Recall of the trusted rule is bested by the single-for rule. The single-for rule is not shown in figures since it behaves very similarly to single-for voting in bridge decision process – it has excellent recall, but its precision falls sharply if parameter k is increased.

Same as for bridge detection, voting brings improvements. Compared with a single round execution of detection algorithm, it can be seen that voting rules provide increase in F-measure of up to 13.4%. Most of them provide increase both in precision and recall.

Compared with the OLSR-based biconnectivity testing, DIBADAWN and voting rules are much more stable. The differences in precision, recall and F-measure between experiments with different link acceptance thresholds are measured in percents, while OLSR completely changes its behavior. For instance, recall of the $Sf, M, T, wB, wD - Majority$ rule and $t = 0.1$ is 0.747 and it changes to 0.744 if $t = 0.316$. In OLSR-based detection, the best detection results are achieved in NPART/Leipzig scenario with Ricean fading but even there the change in the threshold lowers recall from impressive 0.771 to meager 0.57. If Rayleigh fading is used in same topology type, the changes are even larger – recall is almost halved and it falls from 0.634 to 0.342.

Table 9.8 provides more detail on voting rule behavior. It shows that rules from the first voting round are good choice if either excellent precision or recall are needed. The second voting round balances them and provides some improvements in F-measure. The rule $Sf, M, T, wB, wD - Majority$ has the best F-measure if link acceptance threshold is set to 0.1. The $Sf, M, wD - Weighted$ rule that has best F-measure if link acceptance

Acceptance link threshold $t = 0.1$					
Best in	Rule	Precision	Recall	F-measure	Reward
Precision	Unanimous	0.89	0.44	0.59	10
Recall	Single-for	0.43	0.94	0.6	27.2
F-measure	Sf,M,T,wB,wD-Majority	0.77	0.74	0.76	19.9

Acceptance link threshold $t = 0.316$					
Best in	Rule	Precision	Recall	F-measure	Reward
Precision	Unanimous	0.96	0.4	0.57	9.29
Recall	Single-for	0.52	0.96	0.67	26.9
F-measure	Sf,M,wD-Weighted	0.82	0.76	0.79	19.9

Table 9.8.: Best rules for articulation point detection in Motelab experiments. Values are taken for k=5.

threshold is set to 0.316 but it is closely followed by the $Sf, M, T, wB, wD - Majority$ rule that has F-measure of 0.787.

Table 9.8 also demonstrates adaptivity and flexibility of the developed approach. A user can easily choose a rule that suites its needs: excellent precision (0.96), excellent recall (0.94) or balance of two (F-measure close to 0.8) without changes in core DIBADAWN functionality or increase in communication overhead. Nodes in network may choose to use decisions of different rules, according to their needs. Multiple rules may be also used simultaneously at a single node if applications deployed at it require it (e.g., one application needs high recall, other high precision). Each of the applications may use the rule that fulfills its needs since voting rules deliver their decisions simultaneously.

9.4. Implementation of the Approach in Jist / SWANS Simulator

The previous section has shown that the proposed approach operates accurately in reality. However, some aspects of verification could not be performed in testbed due to technical limitations. In this section, Jist/swans simulator is used to evaluate algorithm behavior in large networks (comprising hundreds of nodes), in presence of mobile nodes and to evaluate its sensitivity if incorrect weights are used in voting rules.

JiST is a general purpose discrete event simulation engine that runs in a standard Java virtual machine. The resulting simulation platform is easy to use and program, and it is highly portable due to platform independence provided by Java. Development of new protocols is expedite since a developer may use all existing Java libraries within the simulation code. For instance, it was possible to directly save simulation results to a database, which would be rather difficult from other popular simulation engines, such as ns2.

SWANS is a wireless network simulator that uses the JiST platform and extends it with networking- and wireless-related constructs. The test that are performed by its developers clearly show that it out-performs existing simulation runtimes (ns2 and

GloMoSim) in time and memory consumption [33]. Also, Jist/swans natively supports Rayleigh and Ricean signal propagation models, unlike ns2 that requires custom software components[7].

Jist/swans is rather new simulator engine so the number of supported protocols is rather limited. For instance, OLSR evaluation had to be performed in ns2 because there does not exist an OLSR implementation for Jist/swans. The question that inevitably follows is whether it is possible to compare results derived from two different simulation environments as it is done in this thesis. The study [149] showed that ns2 and Jist/swans agree in evaluation results for protocols that are supported by both of them. Thus, we expect that the comparison of bridge and articulation detection approaches that is performed in this work is fair.

The complete approach that is described in Chapter 5 is implemented. As proposed in Section 5.3, the search in the forward phase of the algorithm is integrated with a reactive routing protocol. AODV was used as the basis of DIBADAWN in this work.

9.4.1. Jist/swans Simulator Setup

High-quality simulator is just one segment of appropriate simulation methodology. The simulation setup is equally important and it must be chosen so that it resembles reality. An inappropriate setup leads to unrealistic behavior of protocols and to wrong conclusions, as it was shown in Section 9.2.

There are numerous aspects of simulation setup that may influence simulation results. In order to ensure validity of conclusions, a set of different setups is used during evaluation process. The approach is evaluated in setups that use Ricean (with Ricean factor K set to seven) and Rayleigh fading, in NPART/Berlin and NPART/Leipzig topologies, with link acceptance thresholds of 0.1 and 0.316.

The simulation outcomes are compared to the outcomes of Motelab experiments and results from Section 9.2. It was observed in evaluation of biconnectivity testing algorithms based on proactive topology management protocols that the accuracy of results is generally better for lower link acceptance threshold of $t = 0.1$. If threshold is set to $t = 0.316$, the precision is high (often over 0.9) but the recall is considerably reduced as well as the F-measure of the decisions.

That prompted us to present the results of the weaker link acceptance threshold in highest detail, and only a subset of results of other simulation setups. The reasoning behind this decision is to compare DIBADAWN to scenarios where proactive performs better. The evaluation performed in Motelab has already demonstrated the stability of the proposed distributed algorithm and associated voting rules with respect to changes in the link acceptance threshold. This stability is also present in Jist/swans simulation, so the advantage of the distributed approach is only larger for the higher link acceptance threshold.

Each of the simulation setups has been studied on 50 different topologies. In the preparation phase of the simulation, it was determined which links in the network pass the acceptance threshold and this topology is used to calculate the set of bridges and articulation points. The DIBADAWN decisions are compared against this set. There exist five constant bit rate flows between random pairs of nodes in the network.

[7] The components are not compatible with all ns2 versions, and their integration in supported versions requires numerous and tedious manual corrections of unsupported source code.

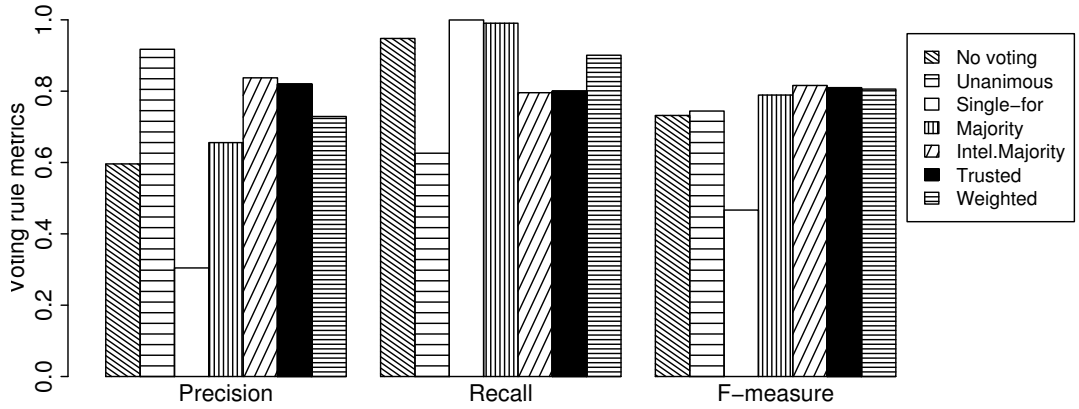


Figure 9.9.: Voting rules for bridge detection. NPART/Berlin placement, Rayleigh fading. Link acceptance threshold $t = 0.1$.

A DIBADAWN search is initiated every 5 seconds in static topologies, and in every topology 2000 searches are executed. Every 30 seconds (same as in the OLSR evaluation in Section 9.2) all nodes in the network are queried on their decisions and those decisions are compared against the correct decisions.

Nodes are configured to purge their histories of decisions, so that only the decision from last 60 seconds are preserved (the implementation also supports keeping of a fixed number of decisions, regardless of their timestamp, but that option has not been used in simulation). This may result in situation that a node does not have enough votes to execute a class of voting rules: e.g., if a node has participated in six searches within last 60 seconds, it cannot participate in voting that requires set of ten last voters. The node simply does not execute such rules. In an application of the detection algorithm where decision is needed a node would rely on decisions based on smaller voter sets until voter set reaches desired size.

9.4.2. Evaluation Results in Static Topologies

This section evaluates accuracy of proposed approach in static topologies that are created by the NPART topology generator, described in Chapter 8. Same as in Section 9.2, Rayleigh or Ricean fading are used as signal propagation model. This section contains the summary of results for the NPART/Berlin topologies. Detailed simulation results of NPART/Berlin and NPART/Leipzig topologies can be found in Appendix D.

Bridge Detection

Same as in the Motelab experiments, trusted and intelligent-majority rules are the best, and difference between them is minor. The unanimous rule is one of best rules with regard to F-measure which is unexpected and opposing to results of Motelab experiments. However, its reward is lower by approximately 30% if compared with the reward of trusted and intelligent majority rules (Figures D.10(b) and D.14(b), page 230). Thus,

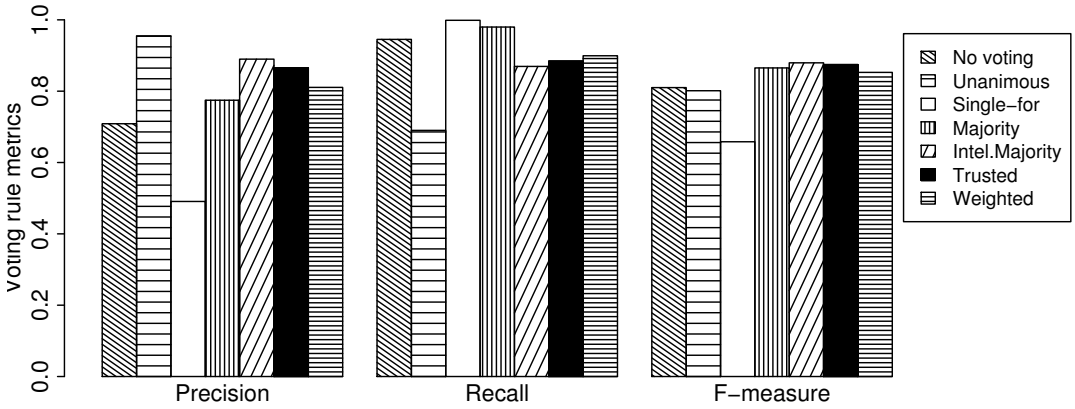


Figure 9.10.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading. Link acceptance threshold $t = 0.1$.

the unanimous rule mostly detects the bridges at network periphery. If compared with Motelab experiment results, the weighted rule has slightly lower precision, but it improves recall. In simulation evaluation it has better F-measure than in experiments, but still not as useful as the trusted and intelligent majority rules because of its need for learning phase.

In Rayleigh scenarios, the direct decisions of the algorithm have excellent recall but their precision is among the lowest (worse is only the Single-for rule). Voting rules provide considerably better precision (32.9% for the intelligent majority rule) with minor compensation in recall, resulting in increase of F-measure of up to 14.1% (for the intelligent majority rule). In Ricean fading, the improvements are also notable. For instance, the intelligent majority rule provides improvement in precision of 19.7% and in F-measure of 8.9%. It should be noted that if the accuracy of one-round decision is already high, the relative improvement is smaller in its absolute and relative value. However, with overall increase in accuracy, improvements are more difficult to achieve[8].

If we compare results of DIBADAWN to results of proactive approach from Section 9.2, DIBADAWN has clear advantage. In presence of Rayleigh fading, both approaches experience reduced accuracy, but DIBADAWN with intelligent-majority and trusted rules still has F-measure that is over 0.8. OLSR in Leipzig placement achieves F-measure that is 0.578 while in NPART/Berlin scenarios its F-measure is only 0.43. If there exists Ricean fading on the channel, DIBADAWN preserves its advantage in accuracy: F-measure of OLSR-based detection is 0.68 in NPART/Berlin scenarios and goes up to 0.82 in NPART/Leipzig scenarios while F-measure of DIBADAWN is above 0.9[9].

[8] With regard to this improvement, accuracy is similar to availability. For instance, an increase in availability from 0.99 to 0.999 may seem minor (relative improvement is only 0.9%) but it is considered as an order of magnitude improvement due to tenfold reduction in system downtime.

[9] Detailed results of DIBADAWN evaluation in NPART/Leipzig topologies can be found in Appendix D, Figures D.9 to D.12, page 230

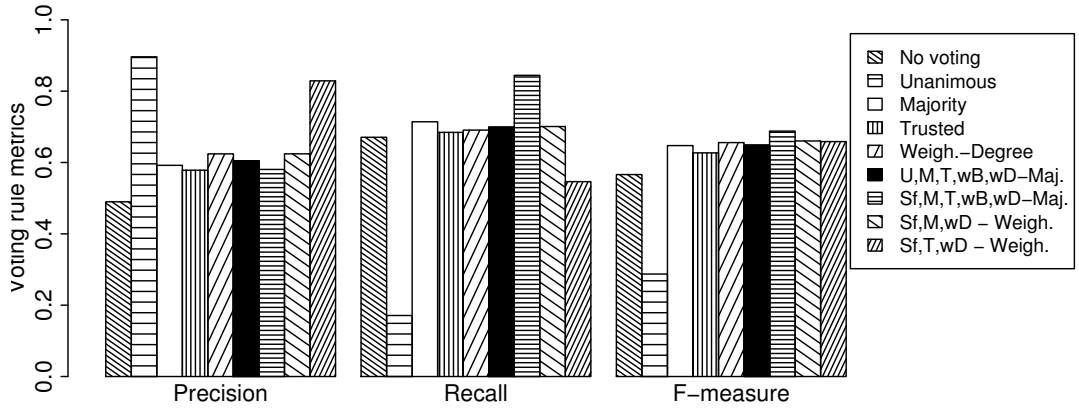


Figure 9.11.: Voting rules for articulation point detection. NPART/Berlin placement, Rayleigh fading. Link acceptance threshold $t = 0.1$.

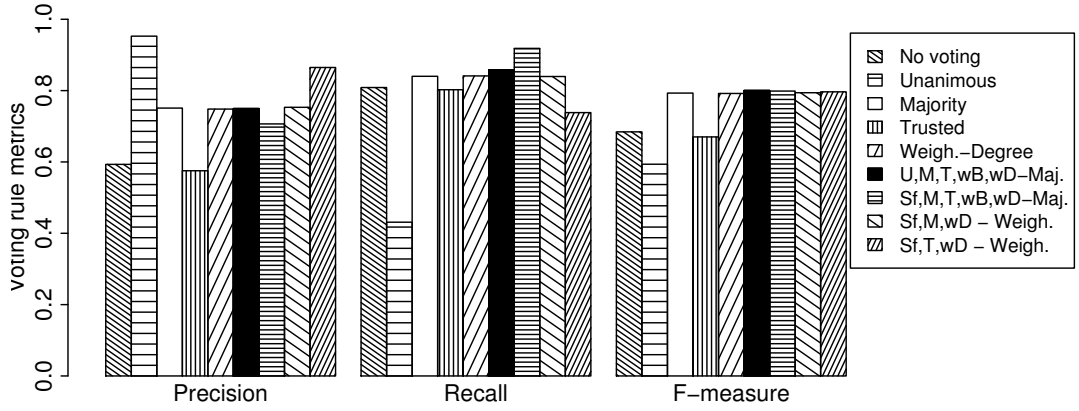


Figure 9.12.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading. Link acceptance threshold $t = 0.1$.

Articulation Point Detection

In articulation point detection, there exists rather large group of rules with similar F-measure. They provide the common tradeoff between precision and recall, keeping the F-measure almost constant. The rules from the second voting round are clearly established as leaders in individual categories. The rule $Sf, M, T, wB, wD - Majority$ has decent precision and very good recall and reward, which makes it one of our favorites (it also achieved good results in Motelab experiments). Rules $Sf, M, wD - Weighted$ and $Sf, T, wD - Weighted$ have preference to high precision, sacrificing some recall in turn, but they stay close to rule $Sf, M, T, wB, wD - Majority$ with regard to F-measure in Rayleigh scenarios and even outperform it slightly in Ricean scenarios.

Same as for the bridge detection, the direct output from the algorithm has good

recall and low precision. In comparison with it, the proposed voting rules provide both improvements in recall (e.g., the $Sf, M, T, wB, wD - Majority$ has 28.3% better recall in Rayleigh fading scenario) and in precision (e.g., $Sf, T, wD - Weighted$ has 40.7% better precision in Ricean fading scenario). This results in improvements of F-measure of up to 12% in Rayleigh and 16.9% in Ricean fading scenarios.

Compared to OLSR, DIBADAWN is particularly advantageous if Rayleigh fading is present on the communication channel. DIBADAWN's F-measure is better for 45% or more: for $t = 0.1$, DIBADAWN in NPART/Berlin has F-measure of 0.65 versus 0.44 of OLSR. In NPART/Leipzig topology, for $t = 0.1$ OLSR and DIBADAWN both reach F-measure of approximately 0.6.

It was already noted that in presence of Ricean fading OLSR-based detection drastically improves its performance. For NPART/Leipzig and $t = 0.1$ it even outperforms accuracy of DIBADAWN – its F-measure is 0.82 which is better than DIBADAWN's 0.76 ($Sf, T, wD - Weighted$ rule). However, in NPART/Berlin topologies, DIBADAWN is better than OLSR for both link acceptance thresholds. The $Sf, T, wD - Weighted$ rule is the best for link acceptance threshold $t = 0.1$. Its F-measure is 0.8 which is higher than OLSR's 0.72[10].

This comparison of DIBADAWN and OLSR-based biconnectivity testing emphasizes the importance of introduced voting rules. Without them, the detection accuracy would be worse than for OLSR-based biconnectivity testing. But the introduction of rules greatly increases the accuracy of decisions without additional communication overhead.

Discussion of Simulation Results

Despite its lower communication overhead, accuracy of DIBADAWN outperforms OLSR in all but one simulation scenario. DIBADAWN is particularly advantageous for bridge detection where thanks to its intelligent rules, it is constantly better than the OLSR based detection. Also, it has better resilience to high variations of signal strength (Rayleigh fading), where it bests the OLSR-approach in every category.

The voting rules are crucial for this advantage of DIBADAWN. Without them, decisions of distributed bridge and articulation point detection are the same or worse than the decisions made with support of OLSR. The voting rules bring clear improvements in accuracy so it was possible to improve precision of single-round decisions up to 40.7% and recall up to 28.3% (improvements may differ, depending on the scenario and the voting rule).

The results of Jist/swans evaluation are comparable with Motelab experiments. Behavior of DIBADAWN and voting rules in simulation is similar to their behavior in experiments, and the values of accuracy metrics show reasonable fit. A partial surprise is that simulations with Rayleigh fading have worse fit to experiments than those with the Ricean fading, although Motelab is located in a building where most of nodes are not placed in line-of-sight. Such behavior is explainable if we consider that in a building there are obstacles which are impenetrable by the wireless signal. In such situations, the fading plays diminutive role and shadowing is dominant – links that penetrate through multiple walls are of exceptionally poor quality and remain so. The experimental implementation of DIBADAWN has been able to profit from this sort of stability in the

[10] Detailed results of DIBADAWN evaluation in NPART/Leipzig topologies can be found in Appendix D, Figures D.17(a) to D.20, starting from page 232

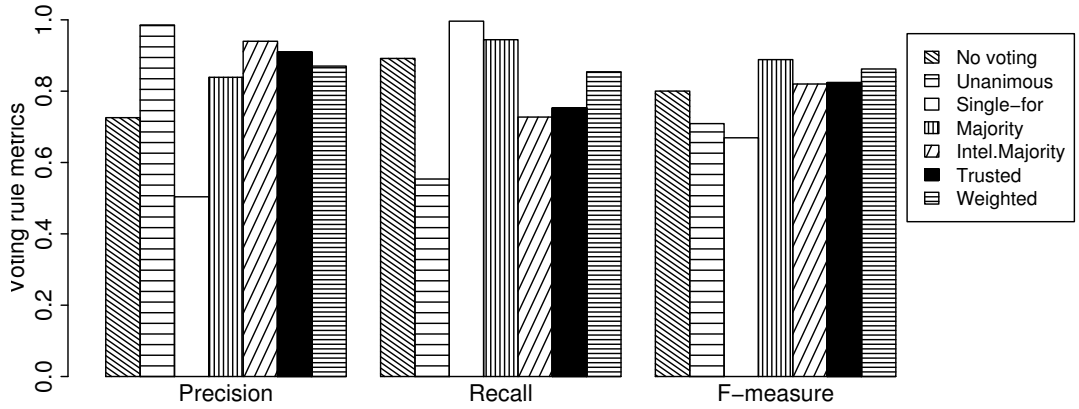


Figure 9.13.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading. Link acceptance threshold $t = 0.316$.

network since such links are unlikely to be observed and included in search tree nor recognized as cross-edges. Fading is implemented in simulation as a stochastic process and it may produce very unstable links that introduce faults in the detection algorithm providing contradicting decisions to the voting rules, reducing their effectiveness.

Finally, it must be noted that differences in accuracy of second round rules in static networks are small. It is difficult to appreciate the need for their evaluation and use, but their role will become clearer in the following sections when node mobility or local searches are introduced.

9.4.3. Assessing the Effects of Environmental Changes to Accuracy of Approach

So far, the effectiveness and accuracy of the algorithm in static topologies for a single link acceptance threshold has been evaluated. This section assesses behavior of approach and accuracy of its decisions for different link acceptance thresholds, if weights for voting rules are incorrect, and with node mobility.

Through comparison of the simulation results with the experiments in the testbed, it has been concluded that simulation setups with Ricean fading possess better resemblance to the experiences from experiments.

The results of evaluation for NPART/Berlin placement model with Ricean fading are presented in detail in order to avoid tedious repetition of similar diagrams. Additional advantage of NPART/Berlin as a node placement scenario is its lower node density, where node mobility produces larger topological changes than in denser NPART/Leipzig topology.

Detection Results for Different Link Acceptance Threshold

In this section, the link acceptance threshold is set to 0.316. The Figures 9.13 and 9.14 show the evaluation results.

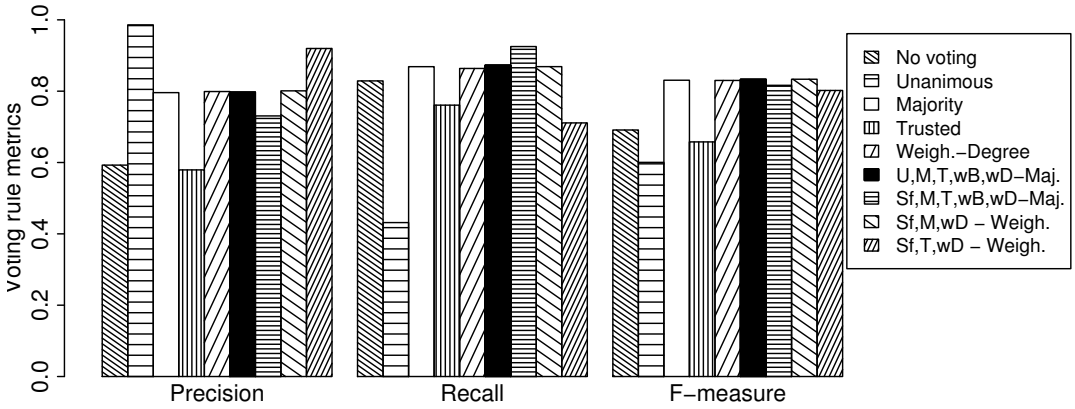


Figure 9.14.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading. Link acceptance threshold $t = 0.316$.

For link acceptance threshold of $t = 0.316$, the trusted rule for bridge detection experiences issues with detecting cycles over links that should not have been accepted to topology. As predicted in Chapter 4, the number of false positives in link detection rises with increase of link acceptance threshold. The effects are more pronounced for larger k since effects of false positives in link detection are cumulative for the trusted rule. The majority rule achieves best F-measure in this scenario, exceptional recall and decent precision. The weighted rule has better accuracy in this scenario than in simulations for $t = 0.1$ and Motelab experiments, reaffirming previous judgement that it is not particularly stable.

Evaluation of rules for this threshold demonstrates the importance of the reward metric. If we compare recall of the majority and the trusted rule, the majority rule is a clear winner (for k=5, the majority rule has recall of 0.945, the trusted rule has 0.84). However, in terms of reward, the trusted rule is actually better: its average reward (for $k = 5$) is 161 versus 155.1 of the majority rule (Figures D.25(b), and Figure D.26(b) on page 235). So, trusted rule fails to detect some bridges on network outskirts but detects more bridges in network core, which is a better option in most of application scenarios.

The improvements of the voting rules over direct detection results are modest – five to six percent in F-measure of trusted and intelligent majority rules, and eight for the majority rule. Same as before, improvements are more notable for precision – precision of the intelligent majority rule is 23.7% higher than of the direct decisions. The trusted rule retains the advantage with regard to reward over direct algorithm outcome: the direct decisions have higher recall than trusted rule (0.892 to 0.839) but the average reward of the trusted rule is 16.7% higher (161.1 to 138.5).

Tables 9.9 and 9.10 compare selected DIBADAWN rules with the OLSR-based biconnectivity testing.

The accuracy of OLSR-based detection for link acceptance threshold $t = 0.1$ were comparable with results obtained by DIBADAWN, but if the threshold is changed to $t = 0.316$, DIBADAWN has important advantages. In bridge detection (Table 9.9), DIBADAWN has slightly lower precision (maximum difference in favor of OLSR is 8.2%

Decision System	Precision	Recall	F-measure
NPART/Leipzig, Rayleigh			
OLSR	0.897	0.324	0.473
Trusted	0.8565	0.7502	0.7998
Intelligent Majority	0.8784	0.7359	0.8008
Majority	0.6587	0.9522	0.7787
NPART/Leipzig, Ricean(7)			
OLSR	0.951	0.495	0.651
Trusted	0.9066	0.8635	0.8845
Intelligent Majority	0.9291	0.8426	0.8837
Majority	0.7986	0.9628	0.8730
NPART/Berlin, Rayleigh			
OLSR	0.824	0.31	0.45
Trusted	0.7556	0.7042	0.7289
Intelligent Majority	0.8016	0.6860	0.7393
Majority	0.6554	0.9280	0.7682
NPART/Berlin, Ricean(7)			
OLSR	0.903	0.397	0.551
Trusted	0.8523	0.8393	0.8457
Intelligent Majority	0.8985	0.8084	0.8510
Majority	0.7984	0.9448	0.8654

Table 9.9.: Comparison of proactive and DIBADAWN-based bridge detection algorithms. Link acceptance threshold $t = 0.316$, DIBADAWN parameter k=5.

for NPART/Berlin topology with Rayleigh fading) but much better recall, in particular for the majority voting rule. The least advantage of recall of majority voting over OLSR is 0.467 (the absolute difference between achieved recalls, which is improvement of 94%) for NPART/Leipzig scenario with Ricean fading and the improvement factor is almost three for both scenarios with Rayleigh fading. This advantage of DIBADAWN is consequently transferred to improvements in F-measure where DIBADAWN has improvements of 0.23 to 0.32 in absolute difference which translates to improvements of 36% to 69%.

OLSR-based approach has advantage over DIBADAWN in precision for articulation point detection. In three out of four studied cases, its precision is over 0.93 which is impressive. However, this high precision comes at the price of its reluctance to declare a node as an articulation point, so it detects only a small portion of the articulation points which are detected by DIBADAWN. The advantage of DIBADAWN in recall is substantial, so even the second-voting round rules with lowest recall of all second-voting round rules clearly outperform OLSR by at least 30% (the $Sf, T, wD - Weighted$ rule in NPART/Berlin topology with Rayleigh fading). This is reflected in F-measure metric and DIBADAWN with the associated voting rules is clearly better than OLSR in all evaluated scenarios.

The comparison with OLSR-based detection stresses once more the importance of voting rules and in particular the second-voting round. If direct DIBADAWN decisions are taken, its accuracy fares worse than the OLSR, even in Rayleigh scenarios where

Decision System	Precision	Recall	F-measure
NPART/Leipzig, Rayleigh			
OLSR	0.958	0.342	0.504
U,M,T,wB,wD-Majority	0.6106	0.6758	0.6415
Sf,M,T,wB,wD-Majority	0.5804	0.8054	0.6746
Sf,M,wD-Weighted	0.7645	0.6035	0.6745
Sf,T,wD-Weighted	0.8404	0.5384	0.6563
NPART/Leipzig, Ricean(7)			
OLSR	0.97	0.57	0.718
U,M,T,wB,wD-Majority	0.7205	0.8276	0.7703
Sf,M,T,wB,wD-Majority	0.6986	0.9029	0.7877
Sf,M,wD-Weighted	0.7379	0.8137	0.7739
Sf,T,wD-Weighted	0.8845	0.7400	0.8058
NPART/Berlin, Rayleigh			
OLSR	0.807	0.286	0.42
U,M,T,wB,wD-Majority	0.7112	0.7262	0.7186
Sf,M,T,wB,wD-Majority	0.6416	0.8120	0.7168
Sf,M,wD-Weighted	0.7412	0.6183	0.6741
Sf,T,wD-Weighted	0.8282	0.5185	0.6377
NPART/Berlin, Ricean(7)			
OLSR	0.931	0.461	0.616
U,M,T,wB,wD-Majority	0.7062	0.8605	0.7757
Sf,M,T,wB,wD-Majority	0.6595	0.9308	0.7720
Sf,M,wD-Weighted	0.7843	0.8522	0.8168
Sf,T,wD-Weighted	0.8819	0.7664	0.8201

Table 9.10.: Comparison of proactive and DIBADAWN-based articulation point detection algorithms. Link acceptance threshold $t = 0.316$, DIBADAWN parameter k=5.

it has biggest advantage if voting rules are applied. For instance, F-measure of direct DIBADAWN decisions in NPART/Leipzig with Rayleigh fading is mere 0.43, which is 14% worse than the OLSR. First round voting rules improve it a bit, so majority rule with $k = 5$ has F-measure of 0.51 which is equal to OLSR-based detection. The second voting rules is decisive in improvement of DIBADAWN accuracy and with it F-measure reaches range between 0.64 and 0.67 which is a clear improvement.

Same as in Motelab experiments voting rules do not change their behavior significantly and as the consequence DIBADAWN experiences less issues with changed link acceptance threshold than OLSR. It is particularly advantageous in presence of Rayleigh fading, where it is better than the proactive-topology-management option by margin of 50% or more. Its decisions are also more balanced and it is capable of detecting considerable number of bridges and articulation points with substantial precision, unlike OLSR which in this evaluation scenario has excellent precision but poor recall and F-measure.

NPART/Leipzig, Ricean(7)

	Correct weights			Incorrect weights		
Voting rule	Precision	Recall	F-measure	Precision	Recall	F-measure
wD	0.6774	0.7674	0.7195	0.6493	0.7977	0.7158
U,M,T,wB,wD-Majority	0.7205	0.8276	0.7703	0.6363	0.8071	0.7115
Sf,M,T,wB,wD-Majority	0.6986	0.9029	0.7877	0.6172	0.8849	0.7271
Sf,M,wD-Weighted	0.7379	0.8137	0.7739	0.6493	0.7975	0.7158
Sf,T,wD-Weighted	0.8845	0.7400	0.8058	0.7955	0.7393	0.7663

NPART/Berlin, Ricean(7)

	Correct weights			Incorrect weights		
Voting rule	Precision	Recall	F-measure	Precision	Recall	F-measure
wD	0.7187	0.8229	0.7672	0.7984	0.8144	0.8063
U,M,T,wB,wD-Majority	0.7062	0.8605	0.7757	0.7836	0.8281	0.8052
Sf,M,T,wB,wD-Majority	0.6595	0.9308	0.7720	0.7356	0.8968	0.8082
Sf,M,wD-Weighted	0.7843	0.8522	0.8168	0.7987	0.8143	0.8064
Sf,T,wD-Weighted	0.8819	0.7664	0.8201	0.8619	0.7631	0.8094

Table 9.11.: Comparison of DIBADAWN-based articulation point detection for correct and incorrect weight assignments. Link acceptance threshold $t = 0.316$, DIBADAWN parameter k=5.

Weight Assignment and Its Effects on Accuracy

DIBADAWN actively uses weighted rules for articulation point detection. It has been shown so far that second-round voting rules produce better results than simple rules from the first voting round. However, all rules that achieved good results in the second voting round used some sort of weights:

- Second-round majority rules include weighted rules from the first round.
- Second-round weighted rules include weighted rules from the first round and additionally use weights that quantify accuracy of these rules from the first round.

It was explained in Chapter 5 that a learning phase must be executed in order to estimate the values of weights. In previous sections weights were obtained for a given deployment scenario: e.g., a learning phase was performed on five NPART/Berlin topologies for Ricean propagation model and these weights have been used in simulation of all subsequent NPART/Berlin topologies with Ricean fading.

It has to be considered, as pointed out in Chapter 5, that network or its environment may change during the lifetime, possibly invalidating the weights and influencing the accuracy of rules. In order to evaluate effects of such changes to detection accuracy, weights are derived from a mixed learning set in this section. Learning phase was executed on five topologies created by NPART/Berlin, five created by NPART/Leipzig, using both Ricean and Rayleigh fading propagation (20 different simulations have been used to provide data for learning set).

The non-optimal weights derived from the mixed learning set are used to evaluate detection accuracy in individual networks/propagation models. Table 9.11 shows the results of evaluation if such incorrect weights are used. The accuracy remains good, in particular of the weighted-degree rule from the first voting round and the second-round majority rules. The weighted rules in second-voting round experience larger loss in accuracy that goes up to 0.05 in NPART/Leipzig scenario. In NPART/Berlin scenarios are even observed small accuracy improvements for some rules.

The effects of weight change are minor and they do not compromise the weighted rules. The learning set which was used in this section was not optimized for a specific topology and propagation model, yet the accuracy remained good. We believe that the reasons for such behavior lie in fact that dominant factor for application of weighted rules for bridge and articulation point detection is not in the absolute value of assigned weights but in the relative ordering of weights, which remain rather constant.

Effects of Contention with Network Traffic on Accuracy of DIBADAWN

In the presented simulation results, DIBADAWN is competing for access to the wireless channel with five independent flows in the network. Additional network traffic increases probability of collisions and packet losses both in forward and backward phases of DIBADAWN. These losses are translated to algorithm faults, as described in Section 5.5 and they may increase the number of false positives and false negatives in the detection process.

In order to evaluate sensitivity of DIBADAWN to this additional traffic, additional simulations with higher number of traffic flows have been executed.

Figure 9.15 shows the performance of selected rules as function of number of background traffic flows. As it can be seen, the reduction in accuracy for bridge detection is scarcely noticeable. Decrease of accuracy for articulation point detection rules is slightly larger, but the difference is still minute.

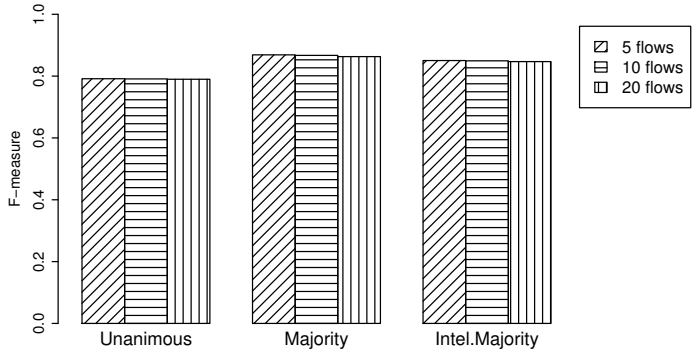
Introducing Mobility and Assessing its Effects on Accuracy

For evaluation of mobility effects on the DIBADAWN, the static placement from Berlin is combined with a subset of moving nodes. A city-alike scenario is obtained where a set of nodes forms static part of the network, and a smaller subset roams the city. RWM model has been used for mobile nodes. Pause time between successive movement stages of a node is set to 15 seconds. Node speed is selected from interval (7.2km/h, 18km/h). The lower speed-limit is approximately pedestrian walking velocity, and the upper limit is comparable with the average speed of a car in central city areas (a statistical study [65] performed by the Department of transport in Great Britain determined that the average speed of a car in central London area is 10.5m/h=16.8km/h).

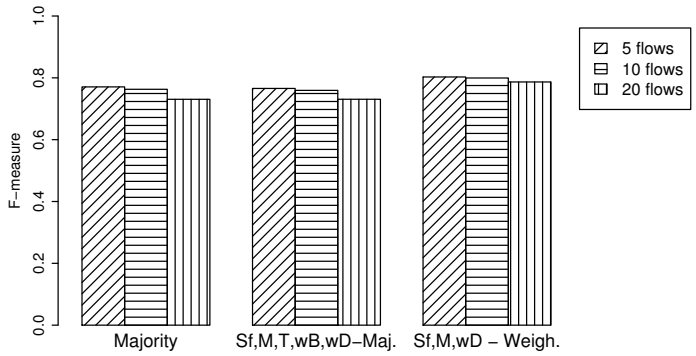
Since the available mobility models in Jist/swans simulator do not provide accurate modeling of the car movement (they tend to move faster in some sections, to move slowly or experience a stall in other sections of the network/city), it has been decided to use their average speed as the upper speed limit of RWM. In order to compensate for topological changes produced by node mobility, DIBADAWN is executed every four seconds instead of every five like in static scenarios. Since topologies are dynamic, the omniscient observer recalculates connectivity graph before it evaluates quality of DIBADAWN decisions.

Due to the sparse topology of the static part of the network, the topological changes introduced by this mobility model are considerable. For instance, the average difference Δ_{ap} in set of articulation points between successive 30-second sampling intervals is 10.83 (95% confidence interval is [9.96, 11.7]).

$$\Delta_{ap} = |(AP_i \backslash AP_{i-1}) \cup (AP_{i-1} \backslash AP_i)| \tag{9.2}$$



(a) Bridge detection.



(b) Articulation point detection.

Figure 9.15.: Comparison of F-measure of selected voting rules in presence of increased network traffic. NPART/Berlin placement, Ricean fading. Link acceptance threshold $t = 0.1$.

This difference includes both nodes that were not articulation points at the previous sampling point (AP_{i-1}), but they are in the current (AP_i); and the nodes which were articulation points at the previous sampling point but are no longer.

The effects of mobility on accuracy reduction are more pronounced for bridge than for the articulation point detection. This was to expect since the best rules for bridge detection were trusted and intelligent majority rules. They rely on trusted messages and in presence of mobility, trusted markings start working against rule accuracy: a single outdated trusted marking claiming that a link is not a bridge may overrule whole set of recent markings that claim otherwise.

Other rules are affected as well and Figure 9.16 demonstrates this trend. It can be seen that the detection has lower accuracy than in the static NPART/Berlin topology with Ricean fading (Figure 9.10). For instance, the majority rule has excellent recall but its precision is low and cannot be even compensated by increase in parameter k (Figure D.29, page 236).

The voting rules are still capable of improving accuracy of direct decisions of the

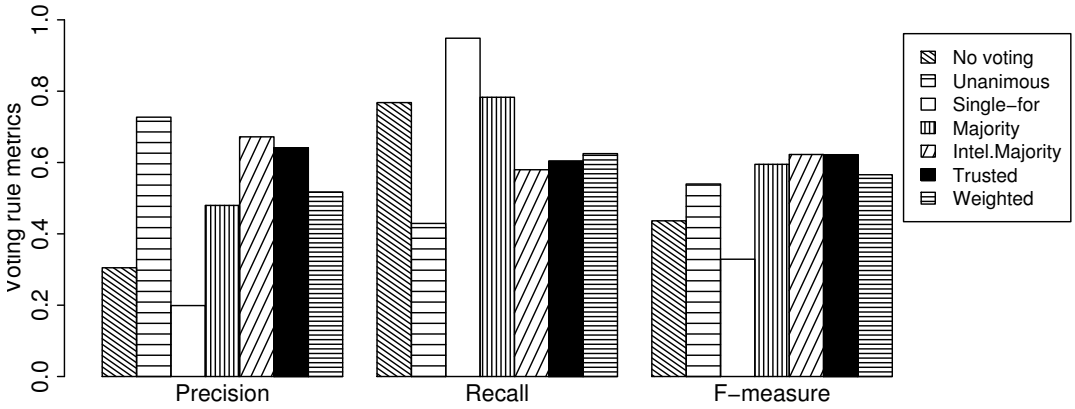


Figure 9.16.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading, mobile nodes. Link acceptance threshold $t = 0.1$.

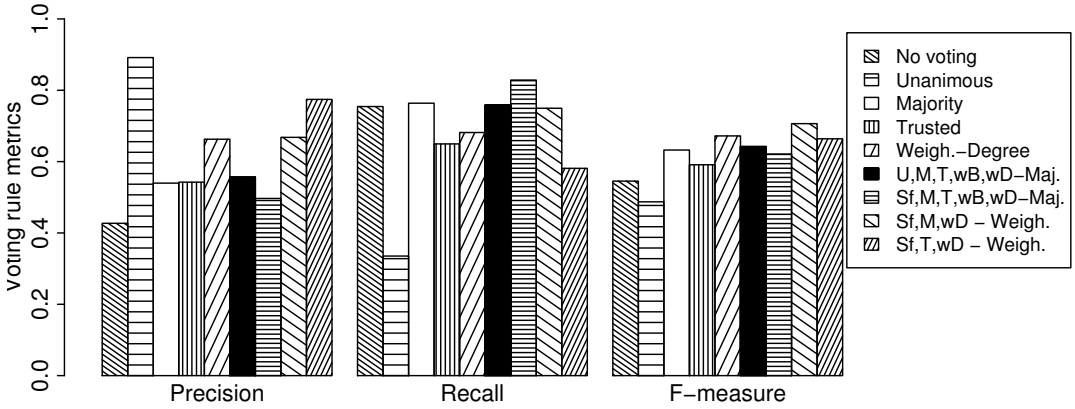


Figure 9.17.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading, mobile nodes. Link acceptance threshold $t = 0.1$.

algorithm. The changing topology and the wireless channel fading introduce numerous faults into the algorithm. So it cannot properly detect bridges neither in the more static parts of the network nor where the mobility is more expressed. The voting rules considerably improve decisions of the detection algorithm, up to 38% for the intelligent majority rule. The relative improvements are even larger for precision, where the single round detection precision of 0.3 is increased to 0.54 (or 77.6%) by the intelligent majority rule.

The weighted rules for articulation point detection show their importance in mobility scenarios. Due to their ability to differentiate between decision quality of voting rules from the first voting round, they provide excellent detection results. The second-round weighted voting rules are particularly effective. Rules $Sf, M, wD - Weighted$

and $Sf, T, wD - Weighted$ reach F-measure of 0.7. As usual, rule $Sf, M, T, wB, wD - Majority$ has best recall and reward but its precision is low (close to 0.5) and thus its F-measure suffers.

The highest improvement in F-measure over direct algorithm decision of 27.4% is provided by the $Sf, M, wD - Weighted$ which is followed closely by the $Sf, T, wD - Weighted$ rule and its improvement of 26.8%. These improvements are caused by their exceptional precision. The $Sf, T, wD - Weighted$ rule improves the precision of the direct decisions by 75.7%. If recall maximization is important, $Sf, M, T, wB, wD - Majority$ rule improves it by 8.2%, simultaneously increasing precision by 18.8%.

9.5. Locality Characteristics of Wireless Multi-hop Networks and DIBADAWN

The average cycle size in random geometric graphs was analyzed in Chapter 6 in order to estimate efficiency of DIBADAWN. The developed models show that the average cycle size is rather small even for sparse graphs and that it has rather fast convergence to three.

Based on this observation it was proposed to apply searches of limited forward radius (limited TTL of messages in the forward phase of DIBADAWN) with the goal of communication overhead reduction. According to Equations 6.17, 6.25 and 6.32, and Figure 6.6 (page 119) the average cycle size in a network with the average node degree that is close to four (average degree of NPART/Berlin is 4.17) is below 7.3 (firm limit, in accordance to Equation 6.17) and it may be as low as 6.59 (if approximation from Equation 6.32 is applied). Thus, a forward search with TTL=4 covers all average and below-average sized cycles (cycle is closed from two sides) and it should already provide quite accurate detection results.

Some doubts on direct applicability of obtained results have been expressed in Chapter 6 because of simplifications that were used in the mathematical model:

- Real networks are not uniformly distributed.
- There exists non negligible impact of stochastic communication on biconnectivity testing that was not included in model that was used in Chapter 6
- HCA cycles that are discovered by DIBADAWN may not be equally sized nor shaped as the shortest cycles.

In order to answer these questions and resolve whether local searches can be used for biconnectivity testing, this Section evaluates the accuracy of the algorithm a with limited search radius. In addition to the mathematically predicted search radius of four, the behavior of the algorithm for TTL=2 is evaluated (two is the smallest possible forward search radius which is able to detect a cycle in a network).

Searches with limited forward radius produce less overhead but they also cover less of network. In order to counter this, searches have to be executed more frequently. Instead of executing a search every five seconds as it was done in Section 9.4.2, a node in network starts a limited search every three seconds. Other parameters of the simulation setup remain the same as it was described in Section 9.4.1.

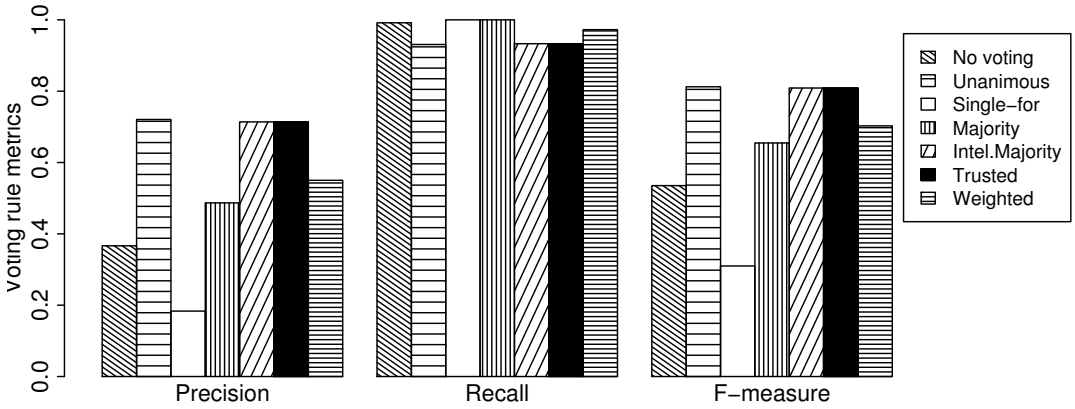


Figure 9.18.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading, forward search radius=2. Link acceptance threshold $t = 0.1$.

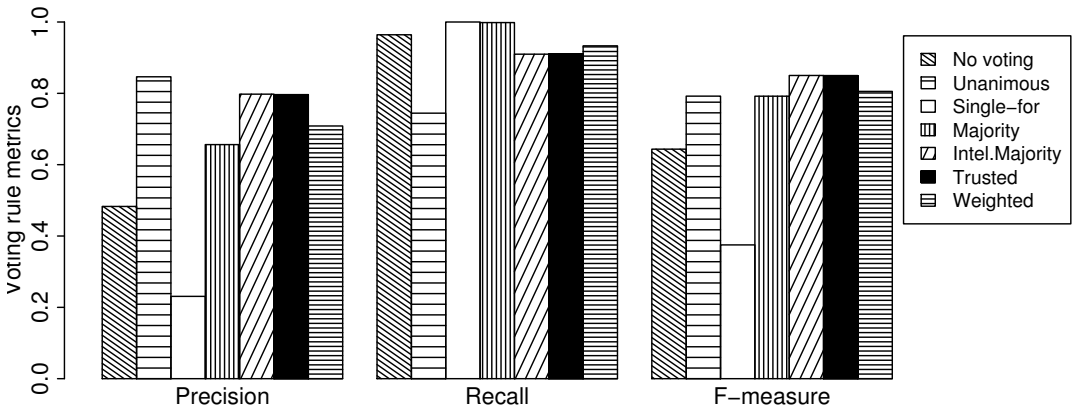


Figure 9.19.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading, forward search radius=4. Link acceptance threshold $t = 0.1$.

Bridge detection results

Figures 9.18 shows the behavior of the algorithm if forward search radius is limited to two. The recall of bridge detection is excellent if such a small forward radius is used. The reasons are simple – probability of incorrect pairing of markings and false negative in bridge detection is reduced due to extremely short path that markings travel in the backward phase of DIBADAWN.

However, small search radius also reduces precision – edges that belong to cycles longer than four are unaware of this fact and may be declared as bridges (unless they also belong to another cycle that can be discovered with this small forward radius). As the result, number of false positives grows and precision of all rules is considerably smaller than for the complete search (compare Figure 9.10 with Figure 9.18). Precision

of majority rule is close to 0.4. The weighted rule is slightly better, but it cannot develop its potentials due to limited set of weights: all markings travel either one or two hops down the search tree. Unanimous, trusted and intelligent-majority rules have acceptable precision. Their F-measure is decent due to exceptional recall they all have. Still, it is notably smaller than F-measure of rules in Figure 9.10.

If TTL of packets in the forward phase is increased to four, as suggested by the developed mathematical model, the bridge detection process behaves almost identically to the standard search where the whole network is covered by a single search. Precision, recall, and F-measure of trusted and intelligent majority rules with TTL=4 in Figure 9.19 are very close to values in Figure 9.10.

The improvements of the voting rules are much more expressed for searches with smaller search radius. Without application of voting rules, for TTL=2 the results are of minor importance: the precision is only 0.36. Precision is increased by intelligent majority rule to 0.667, or 82% with a minor loss in recall (recall is reduced by 2.7%). The voting rules provide improvement of F-measure by 47.3%.

For TTL=4 improvements are considerable, although not so drastic as for the local search with TTL=2 since the direct decisions of the algorithm are more accurate, and the intelligent majority rule improves precision by 56.4% and F-measure by 30.6%. The trusted rule provides similar improvement levels.

Articulation point detection:

Limited forward search for articulation point detection with TTL=2 causes considerable drop in accuracy and reward of almost all rules. This was expected, considering all the errors in decision process that are caused by such a small search radius. The effects of small search radius can be seen in Figure 9.20. The unanimous rule behaves more extremely than usual: disparity between its precision and recall has sharply increased. The $Sf, M, T, wB, wD - Majority$ rule has good recall and acceptable precision. If higher precision is needed, the best is the $Sf, T, wD - Weighted$ rule (the unanimous rule has higher precision, but its recall is absolutely unacceptable). The $Sf, M, wD - Weighted$ rule has similar precision but smaller recall than the $Sf, T, wD - Weighted$ rule. Maximal obtained F-measure is 0.69 for the $Sf, M, T, wB, wD - Majority$ rule which is mostly result of its high recall (0.82). $Sf, T, wD - Weighted$ rule has better precision (0.76) but lower recall (0.53) and F-measure of 0.63.

The characteristics of rules considerably improve if forward search radius is increased to four, and it almost reaches accuracy that was obtained in full searches. The best F-measure has the rule *Sf,M,T,wB,wD - Majority* – over 0.75. Other rules follow the improvement, and the reward metric is particularly increased. The rules applied on DIBADAWN decisions with search radius of two have reward between 300 and 400 (Figure D.38(b), page 239), while rules applied on DIBADAWN results with search radius of four (Figure D.40(b), page 240) improve their reward to over 400 and some of them approach 500.

For small search radius of two, the improvements of the voting rules are noticeably high both for precision and recall. So for instance, the $Sf, T, wD - Weighted$ rule improves precision by 50.3% but also recall for 7.8%. The $Sf, M, T, wB, wD - Majority$ rule increases recall by 65.6% and precision by 18.4%. This results in overall increase in F-measure which is up to 38.4% for the $Sf, M, T, wB, wD - Majority$ rule.

Importance of voting rules remain high also for the searches with TTL=4. The

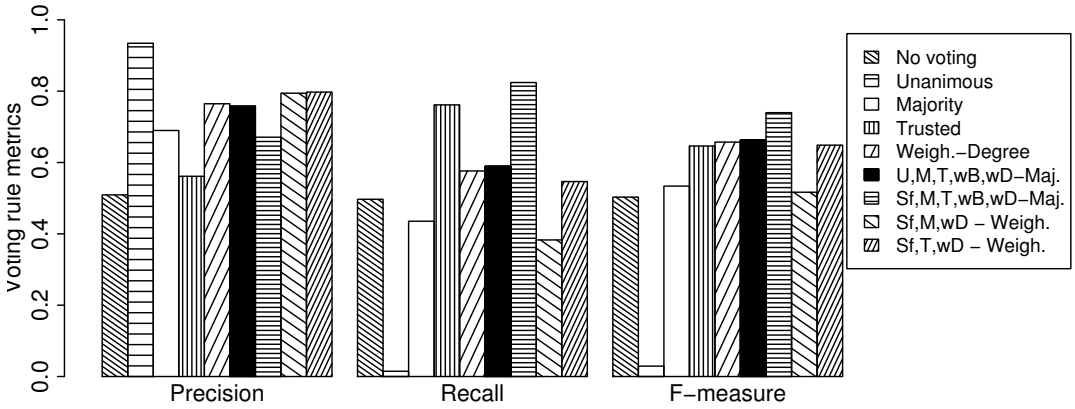


Figure 9.20.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading, forward search radius=2. Link acceptance threshold $t = 0.1$.

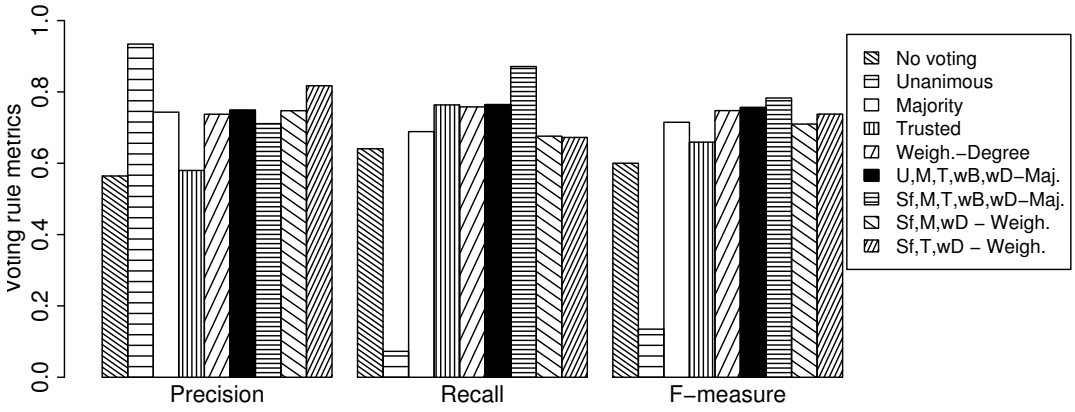


Figure 9.21.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading, forward search radius=4. Link acceptance threshold $t = 0.1$.

precision is increased up to 38.1% for $Sf, T, wD - Weighted$ rule [11], recall of the $Sf, M, T, wB, wD - Majority$ rule is higher by 36.7% which brings F-measure improvements of up to 25.4%.

Local searches for mobile networks

The evaluation of DIBADAWN application for mobile networks in Section 9.4.3 demonstrated that using the same set of parameters for DIBADAWN as in static networks

[11] Unanimous rule provides even higher increases in precision of localized searches but it is disregarded due to its minute recall.

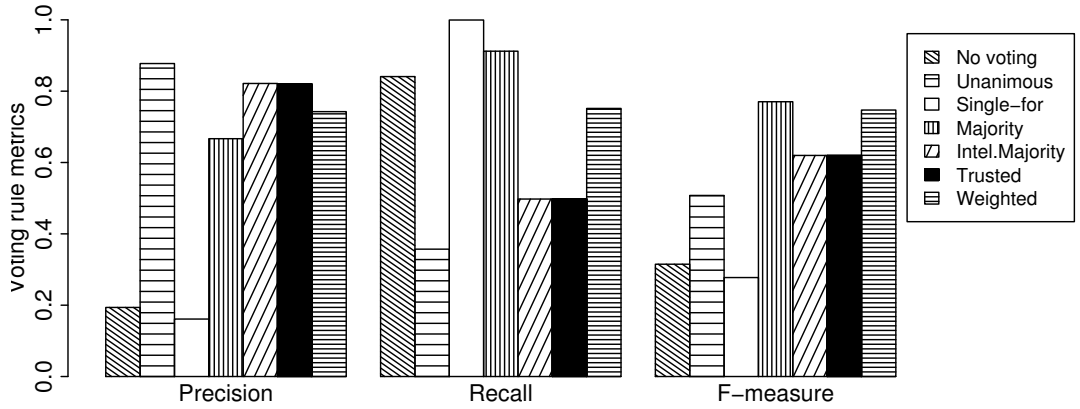


Figure 9.22.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading, mobile nodes. Link acceptance threshold $t = 0.1$. Forward search TTL=4.

results in accuracy reduction, in particular for bridge detection. Node mobility creates and breaks communication links so the DIBADAWN execution once per four seconds results in operation on outdated information.

Mobile networks require more frequent DIBADAWN executions in order to compensate for constant topology changes. However, frequent searches increase communication overhead. The good detection results of DIBADAWN with limited forward search radius seem to provide excellent basis for resolution of issues introduced by node mobility: searches are local and reduce communication overhead, thus they can be executed more frequently, and frequent searches capture the actual network state more accurately.

Figures 9.22 and 9.23 show evaluation results of localized search in mobile scenario that was defined in Section 9.4.3. TTL of messages in the forward phase of DIBADAWN is limited to four, and searches are initiated once per second.

Increased search frequency brings improvements for bridge detection. Using complete searches, F-measure of bridge detection was approximately 0.6 while now it is over 0.7. The articulation point detection stays at the same level – F-measure of the $Sf, M, T, wB, wD - Majority$ rule reaches 0.7.

Similar as in the previous evaluations of the localized search capabilities, the voting rules are crucial for its applicability. The bridge detection without voting rules would have minute practical importance with its poor precision and low F-measure. With voting rules, the results show remarkable improvement, so for instance the intelligent majority rule improves precision 267% and F-measure 119%. Improvements in articulation point detection are not as huge as for bridge detection but still notable and important. For instance, the $Sf, M, T, wB, wD - Majority$ rule increases precision by 30.6%, recall by 22.4% and F-measure by 26.9%.

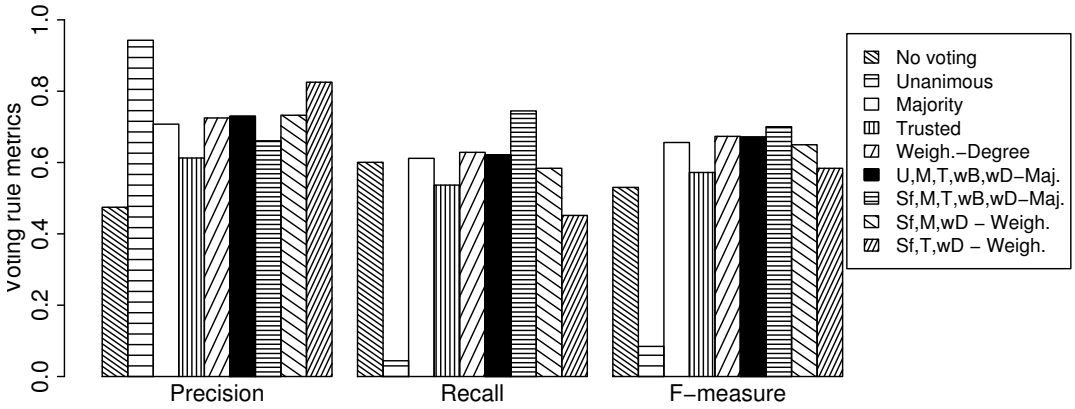


Figure 9.23.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading, mobile nodes. Link acceptance threshold $t = 0.1$. Forward search TTL=4.

Summary of localized searches

It can be concluded from presented results that localized searches show great potential. Localized search with TTL=4 imposes a loss of approximately 0.05 in the F-measure, which is balanced by considerably smaller overhead per search. Using TTL smaller than four is not recommended. Due to very high recall, their F-measure may be acceptable, but they also introduce considerable number of false positives (reduced precision). If best accuracy of decisions is required, DIBADAWN users should employ full searches.

Local search is excellent solution for bridge and articulation point detection in mobile networks. It can be executed more frequently since a single search produces less communication overhead than a complete network search, thus capturing accurate topological information in the network. Supported by voting rules it then achieves comparable or better detection accuracy than the complete network searches.

The usefulness of information delivered by localized searches is rather limited without voting rules. They thrive on diversity of topological information delivered by the successive searches and provide accuracy that is comparable with those established by complete network searches. The voting rules are particularly important for the searches with minimal TTL of two, where they improve accuracy up to 47.3% for bridge detection (F-measure of the intelligent-majority rule).

9.6. Notes on Bayesian Rules

Bayes classification has been described in Section 5.6.3, but so far it has been omitted from the presentation of evaluation results. It is treated separately in this work since it has shown very peculiar properties. In the evaluation, it has been noticed that its accuracy strongly depends of the evaluation setup:

- Motelab experiments: the results were similar to those of weighted voting rule.

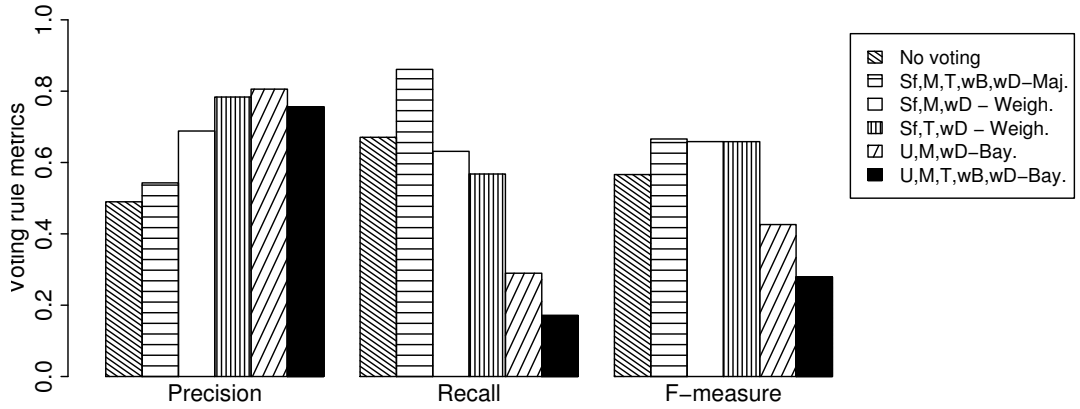


Figure 9.24.: Comparison of majority, weighted and Bayesian voting rules for articulation point detection. NPART/Berlin placement, Rayleigh fading. Link acceptance threshold $t = 0.1$.

Good precision, good recall, good F-measure.

- NPART/Berlin: they achieve excellent precision, in particular for higher values of k. For example, the $U, M, wD - Bayes$ rule with $k = 10$ reaches precision of 0.986 in Ricean fading environment. Recall is average in Ricean fading, but due to the excellent precision value of F-measure remains good. In Rayleigh scenarios, recall experiences considerable reduction.

- NPART/Leipzig: In presence of Rayleigh fading, Bayes classification constantly votes against acceptance of articulation point hypothesis. In Ricean fading environment it provides excellent precision and reduced recall (similar as in NPART / Berlin scenarios).

Bayesian rules also experience reduction in reward due to their reduced recall. This property is not considered as the major issue – if DIBADAWN users are in need of a very precise decision strategy, they would willingly accept such a trade-off. The key problem of Bayesian rules is their instability: not a single rule in NPART/Leipzig setup with Rayleigh fading was operational while in NPART/Berlin node placement with the same propagation model and in the Motelab experiments they provided excellent precision. Such behavior is opposite to behavior of majority-second-voting-round and weighted-second-voting-round rules that exhibit constant performance over all evaluation scenarios.

Because of this unpredictability, Bayesian rules have been removed from detailed evaluation in this work. Still, their good performance in Motelab experiments cannot be ignored. It is planned to test them in different testbeds and to determine whether Motelab setup was accidentally beneficial for them, or they are indeed applicable in real deployments.

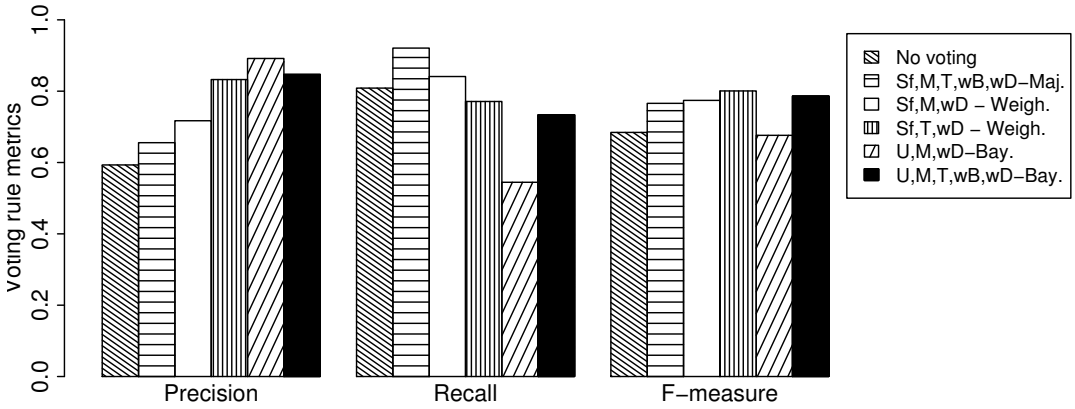


Figure 9.25.: Comparison of majority, weighted and Bayesian voting rules for articulation point detection. NPART/Berlin placement, Ricean fading. Link acceptance threshold $t = 0.1$.

9.7. Application of DIBADAWN for Improvement of Route Discovery Rates

This section provides a simple but illustrative example of DIBADAWN's applicability in WMNs: bridge detection can be used for success rate improvement of route discovery in reactive routing protocols.

Reactive routing protocols have a threshold defining how many times a route search is performed. If destination is not found within these attempts, it is declared as unreachable. For instance, unless a route reply is received after three route discovery attempts, AODV protocol declares the destination node unreachable. Failure of a routing algorithm to discover an existing route can be abstracted as a logical partitioning of a network – the source node which initiates route search believes that destination is unreachable because the network has partitioned or because the destination has failed.

AODV additionally uses the technique named "expanding ring search". It should limit the overhead of route searches by gradually increasing the TTL of successive route requests. The route search starts with low TTL value and if destination is not found, TTL of route request packets is gradually increased and searches repeated until the whole network is searched, followed by two more attempts if needed[12].

AODV uses network flooding as the route discovery mechanism.Numerous network-wide broadcasting methods have been devised in the past, but flooding remains the main route discovery mechanism in reactive routing protocols (e.g., [54] [96] [137]). It retains its importance in protocols since it is robust, simple to implement, usually finds the shortest path between nodes, and does not require specific additional information such as the location awareness of nodes or additional assumptions such as the circular communication range [129].

The flooding as a route discovery mechanism performs well in dense parts of a network

[12]More details can be found in Section 6.4 in [137]

where packet losses cannot significantly influence the search quality – the destination is reached through other available paths. The worst consequence of a packet loss to the route discovery in a dense network is that the shortest path is not found, but a route with suboptimal length.

As already discussed in context of topology dissemination, the flooding is considered overly redundant in dense networks and considerable efforts have been put in order to reduce this redundancy [129] [174]. For instance, [129] proposes that a node should not always participate in broadcast but with a certain probability. Articulation point existence has particularly negative influence to the success ratio of such probabilistic broadcasting scheme – an articulation point may decide not to forward the message, believing it will reach the remainder of the network through other paths, although they do not exist.

The issues of flooding in context of route discovery may arise when it reaches a bridge in the network. If a route request packet is lost at a bridge, the search will not reach a potentially large set of nodes in the network that are positioned beyond the bridge. In consequence, the search has to be restarted after its timeout expires. New attempt covers the already visited nodes, just to attempt the bridge traversal one more time. This is not just a hypothetical scenario since it is exactly what was observed in the community networks: they have numerous bridges, some of them with high packet loss rates (in Berlin, 5.3% of bridges had probability of packet loss higher than 0.9 and 22.6% of them higher than 0.5).

We can conclude that the flooding is highly redundant in dense sections of a network, but insufficiently effective in sparse sections of the network. As a remedy for its reduced coverage in presence of bridges, we propose repeated transmission of route request messages by nodes that are incident to bridges. In [155] was proposed to perform multiple MAC broadcast at every node in the network as countermeasure to the effects of packet losses. Such approach is simple for implementation and helpful in sparse parts of a network but it may work against the improvement in dense sections because it increases the packet collision probability. The idea in this work is to re-transmit the packets only where it is really needed and where it brings highest benefits – at nodes incident to bridges. Such scheme protects the dense parts of the network from unnecessary retries and contention increase.

DIBADAWN is the obvious choice for bridge detection in this application scenario – it is distributed and it can reuse information from route searches in its forward phase.

Simulation setup

The emphasis of simulation evaluation is on the measurement of the raw route discovery success rate so the route caching is disabled. The route caching may (or may not) improve route discovery success rates depending on previous requests, caching strategy and configuration parameters. In presence of caching, the route request success rates are highly dependant on the time an entry is preserved in route cache and on traffic patterns (location of traffic sources, destinations and route-request frequency). But if the raw (without caching) success rate is improved, it also increases the route discovery success rate of protocols that employ caching.

AODV is used as the sample reactive routing protocol due to its wide use, numerous implementations [21] [22] [53] and a formal description in RFC 3561 [137].

The effects of route cashing are eliminated through simulation setup. Route searches are initiated every 200 seconds, which gives more than enough time for removal of all entries from cache ([137] recommends cache hold time for routes to be 30 seconds).

The expanding-ring search of simulation of AODV is disabled. Expanding-ring strategy was devised for use in small- and medium-sized, dense networks. Such network properties increase the probability that a route is found in vicinity of the search source, indeed reducing the number of disseminated route requests. The default protocol parameter values for expanding ring search are set for use in small dense networks: route request TTL starts at one and increases by two until it reaches seven [137]. In Jist/swans version of the protocol the network diameter (TTL_THRESHOLD) has been increased to 19 but the incremental factor (TTL_INCREMENT) remains two.

The issues of the expanding ring search are notable if network is not so dense and/or if it has larger diameter. Successive short-ranged searches cannot reach the destination node if it is on a larger distance than the TTL value of RREQ packets. The AODV protocol calculates the RREQ timeout as

$$2NODE\_TRAVERSAL\_TIME(TTL\_VALUE + TIMEOUT\_BUFFER)$$

which by its default parameter values equals to $80ms(TTL\_VALUE + 2)$. Thus, the successive short-ranged searches and their timeouts may considerably increase the delay between the initial route request performed by an application and the route discovery. Some implementations of AODV introduce additional fixed delay between route discovery retries (up to 2 seconds) further escalating the delay issue. The undesirable consequences of such behavior have been observed by AODV's developers so in its successor (the DYMO routing protocol [54]) they no longer recommend the use of the expanding-ring search.

For this application scenario of DIBADAWN, only bridge detection is required. Thus, localized searches may be used without significant loss in detection accuracy, but with benefit of the overhead reduction. DIBADAWN searches are initiated every eight seconds at a random node in network with forward search radius of four. If DIBADAWN determines that a node is incident to a bridge, route request message will be broadcasted three times at such a node.

The forward search radius of two would be sufficient for this application of the detection algorithm and it would further reduce the communication overhead of bridge detection process. However, the higher TTL value is deliberately chosen in order to simultaneously obtain reasonably accurate articulation point detection results (see Section 9.5), so that they may be utilized by other applications and services in the network. Although it creates higher communication overhead, such setup is more realistic.

Four different topology types are used in the evaluation process: NPART/ Berlin and NPART/Leipzig with link acceptance threshold set to 0.1 and 0.316. The link acceptance threshold is important in this scenario in order to determine which paths exist in the referent topology – the nodes are asked to route test messages only to nodes from the same network component. Combination of two-ray ground and Ricean fading is used as the signal propagation model.

Simulation results

As a prelude, AODV was evaluated in a simulation setup that uses only the two-ray ground propagation model. In NPART/Berlin topologies, 0.992 of routes are found, and 0.997 of route searches are successful in NPART/Leipzig topologies. In [129] [174] the same behavior was demonstrated for uniformly placed nodes, so it can be concluded that node placement model and bridge existence is irrelevant if such idealized propagation model is employed.

In presence of Ricean fading, the behavior of route discovery process drastically changes (Table 9.12). If link acceptance threshold is set to be $t = 0.1$ and NPART/Berlin placement is used, its success ratio falls to be barely over half. As predicted in discussion of the problem, high bridge share and unreliable links have drastically reduced search success rate. The effects are less pronounced if link acceptance threshold is higher ($t = 0.316$) so the success rate grows to 0.67. NPART/Leipzig topologies are less affected than NPART/Berlin because of smaller ratio of bridges in such topologies and in particular because the bridges are not as centrally placed as in NPART/Berlin (see Figure 7.7, page 134). Still, with success rate below 0.78 they are far away from the ideal values which are obtained with the idealized propagation model.

DIBADAWN-assisted route discovery outperforms the traditional AODV for both link acceptance thresholds and in both topology types. The difference is particularly remarkable for NPART/Berlin node placement with link acceptance threshold $t = 0.1$, where the improvement of the search success rate reaches 40%. The success rate improvements in NPART/Leipzig topologies is lower because of the lower bridge share but DIBADAWN-assisted searches still improve the discovery ratio by more than 10%.

Due to the low frequency of searches, a considerable number of packet transmissions are caused by AODV's local-neighborhood detection through heartbeats. Thus, the average number of sent packets per initiated search (columns *TX/search* in Table 9.12)is not drastically larger in case of DIBADAWN if compared with the plain AODV.

The overhead-per-success metric (it is shown in columns *TX/success* in Table 9.12) is more relevant than the overhead-per-search metric, since only the successful route discoveries are relevant for a user. It is defined as the average number of packet transmissions per successful route discovery.

If this metric is used, the DIBADAWN-assisted discovery shows clear advantage over plain AODV – despite the overhead introduced by DIBADAWN, the number of sent packets per successful route discovery is either the same or lower than in case of the RFC-compliant AODV. In the standard AODV, numerous searches are unsuccessful but they also contribute to the communication overhead, while DIBADAWN is more efficient, discovers more routes with less retries, compensating for its initial additional overhead.

In the simulation setup route request frequency has been deliberately set to a very low value: there are 18 searches per hour in a network with 275 nodes which means that a node initiates a route search once per 15.27h on the average. Such low frequency increases the relative overhead of DIBADAWN per route search. If nodes are more active, the communication overhead per search introduced by DIBADAWN is much smaller and its advantage larger.

$t = 0.1$

	AODV			DIBADAWN(trusted)+AODV		
	Succ. ratio	TX/search	TX/success	Succ. ratio	TX/search	TX/success
Berlin	0.514	5939.78	11544.0	0.69	7782.26	11269
Leipzig	0.745	9446.8	12637.38	0.826	9836.43	11904.29

$t = 0.316$

	AODV			DIBADAWN(trusted)+AODV		
	Succ. ratio	TX/search	TX/success	Succ. ratio	TX/search	TX/success
Berlin	0.671	7573.27	11250.15	0.884	8657.54	9793.89
Leipzig	0.778	10088.7	12972.19	0.875	9988.36	11419.77

Table 9.12.: Success ratios for AODV with and without bridge awareness.

Summary

Results of this section demonstrate that bridges impact not only network's reliability but also the behavior of routing protocols and their route discovery success rates.

In presence of bridges in topology, the route discovery mechanism might not be able to find a route between two nodes even if a valid route exists. Evaluation of AODV routing protocol has shown that the probability of discovery of a route can be as low as 0.51.

The DIBADAWN-assisted route request forwarding brings clear improvements in success ratio – depending on the node placement model and link acceptance threshold, the improvements are between 10% and 40%. Because of their higher success rate, DIBADAWN-assisted searches produce smaller communication overhead than the traditional AODV, if the average number of sent packets per successful route discovery is used as a metric. If nodes in a network use DIBADAWN for improvements of the route discovery process, they have at their disposal the DIBADAWN detection results "for free" and can use them for other purposes and applications, such as the traffic shaping or replica management.

9.8. Overview of Voting rules

DIBADAWN is evaluated in experiments and in various simulation setups: static and mobile networks, complete and local search, accurate and faulty voting weights, etc. The number of voting rules and evaluation cases makes it difficult to track their overall behavior so this section summarizes the behavior of voting rules through all of these setups.

The overview of rule behavior is presented in Tables 9.13 and 9.14. Instead of numerical quantification of rule accuracy that was used so far, descriptive quantifiers are used in order to provide a quick guide to rule characteristics and their applicability. The rules are compared by six categories: precision, recall and reward that are obtained in static topologies (they summarize rule behavior in Motelab experiments, different topologies, propagation models and link acceptance thresholds), rule behavior in presence of node mobility, rule applicability for localized detection, and rule stability – is the rule stable over all evaluated scenarios, or its accuracy fluctuates from one evaluation setup to another.

Rule name	Precision	Recall	Reward	Mobility	Local Search	Stability
Unanimous	++	-	- -	-	-	+
Single-for	- -	++	++	-	-	++
Majority	+	++	++	+	-	+
Int.majority	++	+	+	++	++	+
Trusted	+	++	++	+	++	++
Weighted	+	-	-	+	+	-

Table 9.13.: Summary of characteristics of voting rules for bridge detection.

Rule name	Precision	Recall	Reward	Mobility	Local Search	Stability
Unanimous	++	- -	- -	-	- -	++
Single-for	- -	++	++	-	-	++
Majority	-	+	+	+	+/-	-
Trusted	-	++	++	-	+	-
Weighted B.	- -	- -	- -	+	-	- -
Weighted D.	+	+	+	+	+	++
U,M,T,wB,wD - Maj.	+	+	+	+	+	++
Sf,M,T,wB,wD - Maj.	+	++	++	+	++	++
Sf,M,wD - Weighted	+	+	+	++	-	+
Sf,T,wD -Weighted	++	+	-	++	++	+

Table 9.14.: Summary of characteristics of voting rules for articulation point detection.

Voting rules for bridge detection

The unanimous rule offers excellent precision. Its recall is acceptable in some scenarios but its reward is constantly one of the worst, or the worst of all. Its accuracy is not affected much by mobility, but reward remains low, indicating that it detects mostly bridges on network outskirts. Similar behavior is obtained for localized searches.

Single-for rule is the opposite of the unanimous: its recall and reward are exceptional, but its precision and F-measure are very low. It preserves this (rather undesirable) behavior over all evaluation scenarios, earning excellent stability.

The majority rule offers good recall and reward and its precision is acceptable. It adapts itself rather well to the topological changes in mobility scenarios. For localized, in particular for TTL=2 it falls behind best rules.

The intelligent majority and trusted rules offer good to excellent precision (the intelligent majority rule has slight advantage), and high recall and reward (the trusted rule has advantage, in particular in Motelab experiments). In presence of node mobility intelligent majority is slightly better. For localized searches they achieve almost identical accuracy. The differences between them are small, and which one is to be applied depends on user preferences for precision or recall.

The weighted rule is the biggest disappointment with its good precision but meager achievements in all other categories. Also, it is not particularly stable because of differences in its accuracy in Motelab experiments and simulation results.

Voting rules for articulation point detection

The unanimous rule has exceptional precision in all scenarios, but its recall and reward are very low. Unlike the unanimous rule for bridge detection, its performance is particularly bad for mobile scenarios and local searches. The single-for rule follows the same pattern as in bridge detection and it should be avoided even if user needs high recall

– there are rules with comparable recall and reward but much better precision (e.g., $Sf, M, T, wB, wD - Majority$ rule).

The majority rule has decent recall and reward rates in static scenarios but its precision is not particularly good. It is one of better performers in mobility scenarios. It belongs to better rules for localized search with TTL=2 but already for TTL=4 is overcome by other rules.

The trusted rule has exceptional recall and reward, but its precision is very low. There exists differences in its precision between Motelab experiments and simulator studies so the rule is considered unstable.

The weighted-Bridge voting performed poorly in all studied scenarios: low precision, low recall, low reward. Such behavior has been suspected already at its definition: the weights in Figure 5.13(a) do not show clear trends like the weights based on perceived degrees from Figure 5.13(b). The rule tends to be unstable even within a single evaluation scenario (e.g., hectic changes of F-measure in Figure D.8(a), page 229).

Thus, it is a bit surprising that both the trusted and the weighted-bridge rules belong to some of the successful second-round voting rules. The reason behind it may be caused by diversity they bring in decision process – their internal decision logic is different than in other rules used in the second round of voting, so they capture important additional information.

The weighted-degree rule is the best of all the first-voting round rules. Its precision, recall and reward are balanced, and it participates in all presented second-voting-round rules. The $Sf, M, wD - Weighted$ rule has almost identical precision like the weighted-degree rule, but higher recall and F-measure. In some scenarios the improvement in recall is minor, but occasionally it goes up to 10.8%, like in mobility scenario (Figure 9.17). This rule demonstrates capability of second-round voting to improve one property (in this case recall) without reduction in another (precision).

The $Sf, M, T, wB, wD - Majority$ rule provides excellent recall and reward in static and mobile scenarios while its precision is good. Its accuracy for TTL=2 is exceptional, considering the extremely limited search radius (F-measure is close to 0.7). If search TTL is set to four, its F-measure reaches 0.75.

The $U, M, T, wB, wD - Majority$ rule has slightly higher precision and lower recall than the $Sf, M, T, wB, wD - Majority$ rule, with similar F-measure. Both of these rules provide excellent stability – no matter in which scenario they are used, they preserve the same behavior and belong to top performers.

The $Sf, T, wD - Weighted$ rule has very good precision and decent recall. It is particularly good in mobile networks and one of better in localized search with TTL=4 (its precision is second only to the unanimous rule but its recall is much higher). Its main issue is rather low reward.

We can conclude that majority second-round voting provides excellent results with rather small additional effort (it is not necessary to evaluate accuracy of the first-round voting rules as for the second-round weighted voting). Their recall and reward rates are higher than precision, so they are to be used in scenarios where this is more important. If precision is more important than recall, the weighted second-voting-round rules should be used.

9.9. Summary

The evaluation of DIBADAWN and voting rules was performed in Motelab wireless testbed and in the Jist/swans simulator. One of the evaluation goals was to test the stability of the algorithm and voting rules under various conditions, so they were evaluated in setups that combine different topology types, link acceptance thresholds, propagation models. Additional test were performed to evaluate behavior of DIBADAWN in presence of mobile nodes, inaccurate voting weights, and varying traffic rates.

The results of both evaluation methodologies are consistent and ensure us that the proposed approach may be effectively deployed in practice. The implementation experiences from the Motelab experiments confirm that DIBADAWN may be successfully executed even by nodes with very limited processing capabilities, using less than 10KB of operating memory.

The approach which was proposed in the literature was also evaluated in this chapter: proactive topology management and application of Tarjan's DFS for biconnectivity testing to the obtained topology. The simulation results confirmed that the proactive routing protocols cannot adapt to harsh conditions which are encountered in reality, as it was predicted in Chapter 4. Proactive topology management approach provides excellent, almost perfect detection results in scenarios with the idealized two-ray-ground propagation model, but in presence of signal fading, the optimal detection results promptly vanish, and only the decisions at the node level are acceptable.

Channel fading introduces numerous faults to DIBADAWN as well. Empowered with the extensions from the voting theory, accuracy of DIBADAWN decisions outperforms the proactive approach although DIBADAWN has smaller communication overhead. DIBADAWN advantage is particularly large in no-line-of-sight scenarios (Rayleigh fading), and in all scenarios where higher link acceptance threshold is needed.

The localized searches (searches with radius smaller than the network diameter) were proposed in Chapter 6. They are based on the mathematical model founded upon the uniform node placement and it had to be determined if it is applicable in non-uniform node placement which are common in real networks. Evaluation in Section 9.5 has shown that it performs well despite simplifications and approximations that were used in deriving of equations for the average cycle size. The evaluation performed in this chapter closes the engineering cycle: it is common to use simplifications during system modeling, but conclusions derived from model must be evaluated under realistic conditions in order to ensure the agreement of model predictions with the actual behavior of system.

The complete-network searches are recommended if detection accuracy is the major priority. If communication overhead minimization is more important (e.g., in resource constrained networks such as wireless sensor networks), localized searches provide better option: their accuracy is slightly lower than of the complete searches, but the communication overhead per search is considerably lower.

As expected, node mobility reduces the accuracy of approach and demands for higher frequency of DIBADAWN execution in order to capture new state of network topology. Increase in search frequency improves the detection results but it also produces higher communication overhead. A local DIBADAWN search produces substantially less overhead than the complete search, thus local searches may be executed more frequently than complete searches. Frequent localized searches improve DIBADAWN accuracy so that it approaches the accuracy observed in the static scenarios. These characteristic

make localized searches exceptionally well-suited for mobility scenarios.

The characteristics of voting rules were also evaluated in this chapter. Not all of the rules proposed in Chapter 5 have desirable characteristics and high accuracy. The best rules for bridge and articulation point detection have been recognized in this chapter. The rule accuracy was not the only selection criteria. It has been observed in experiments and simulation that some rules provide very good accuracy in several evaluation setups, but disappoint in others. Such rules are considered unstable and they are not recommended for application in practice.

The evaluation has demonstrated that voting rules provide three major benefits:

- Voting rules improve decision accuracy at zero communication overhead. Without voting rules, DIBADAWN was constantly outperformed by the OLSR-based biconnectivity testing. The successive searches capture different information on network topology and provide different views of the topology to the decision making nodes. Nodes are able to utilize this information, improve the accuracy of decisions and outperform OLSR-based approach. The increase in accuracy brought by the voting rules is confirmed both in simulation and experiments. For instance, the F-measure of direct DIBADAWN decisions for bridge detection in Motelab experiments is 0.69, while the trusted rule with $k = 5$ provides F-Measure of 0.87.

- Voting rules provide better adaptivity of the biconnectivity testing in WMNs to application needs, without changes in core functionality of the detection algorithm. The bridge and articulation point detection algorithm is executed as it is described in Section 5.3, but voting rules enable a user to choose the detection characteristics that are best suited for the needs of its application. A good example of this capability of the approach can be found in Table 9.8 which evaluates experimental results. User may choose a rule with excellent precision (0.96), excellent recall (0.94) or balance of two (F-measure close to 0.8).

- Voting rules deliver their decisions simultaneously so it is possible to concurrently use more than one rule. Due to the tradeoff between precision and recall, it is often impossible to produce decisions that excel in both categories, while it is possible to increase one metric at the expense of the other. Two application that have mutually contradicting needs (the first application needs high precision, the second high recall) can both get decision outcomes that suite their needs. Proactive detection approaches may be configured for a given property (e.g. improved precision) but they cannot simultaneously deliver decisions that possess an opposing property (excellent recall).

10. Summary and Outlook

This Chapter summarizes the contributions of the thesis, discusses the benefits and issues of the approach, and points out several promising extensions of the presented work.

10.1. Contributions

At the beginning of the work on this topic, there were several common assumptions in the literature: proactive topology management provides absolutely accurate topology information at every node in static networks with a minor reduction of accuracy in presence of node mobility, bridges and articulation points are rare in connected networks (equivalently – networks are dense and almost always 2-connected), the artificial node placement models provide acceptable approximation of real networks (Table 10.1).

In this thesis each of these claims was verified and it was shown that they cannot be taken for granted, or that they are plainly incorrect. The principal reason for incorrectness of related approaches was improper modeling of WMNs and lack of model comparison to measurements.

Probably the most astonishing revelation in course of this work was that the related approaches for biconnectivity testing in WMNs are completely unaware of issues in topology recognition process introduced by unreliable message delivery in wireless networks. All of them have been blindly accepting the on-off communication link model that has been established in wired networks (where it is accurate), ignoring the stochastic nature of the wireless communication. This is best illustrated by the fact that none of the related approaches attempts to define network topology for communication channel with the stochastic behavior, as it was done in Section 2.2.1. Such rough approximation of wireless link behavior in literature has resulted in numerous unrealistic assumptions in protocol development, improper parameter selection and unawareness of difficulties in

	State-of-the-art	This work
Network models	Uniform and grid placement, Path loss propagation	NPART and Uniform placement, Rayleigh and Ricean propagation, Measurements
Occurrence of bridges and articulation points	Low / non-existent	Numerous unavoidable in practice
General approach	Proactive topology management, DFS for detection	Distributed algorithm, hybrid activation, decision making under uncertainty
Evaluation methodology	Simulation	Simulation and experiments
Accuracy of proactive approach	Excellent accuracy	At node level: acceptable, At network level: no added value

Table 10.1.: Comparison of state-of-the-art for bridge and articulation point detection and findings of this work.

obtaining the detailed global system state in a large distributed system with loss-prone communication channels.

One of the commonest assumptions in literature is that a proactive topology management protocol can deliver accurate topological information to nodes in a WMN. In Chapter 4, stochastic models of heartbeat link detectors have been developed. The probability of erroneous link-detection caused by unreliable communication channel was derived. The results show that errors in link detection inevitably exist no matter which HLDs are applied in a network, or which link acceptance threshold is used.

If errors in output of an HLD are to be kept within acceptable range, its parameters have to be selected appropriately to fit the network characteristics and the link acceptance threshold. However, parameters that are optimal for one network type and link acceptance threshold may be completely inappropriate for a network with different characteristics. The optimally configured HLDs can provide error probability of 0.05 to 0.1 (depending on network characteristics and threshold t), while the link-detection error probability of misconfigured HLDs reaches 0.7.

The effects of HLD errors on behavior of proactive topology management protocols are profound. The mean value of the number of erroneously detected links grows linearly to the number of network links: $|e| \cdot P_E$ (P_E is the probability of erroneous detection of a link). The possibly inaccurate output of HLDs is propagated through a network by proactive topology management protocols. The topology delivered by a proactive topology management protocol to network nodes is inaccurate with high probability and the results of biconnectivity testing may also be erroneous. For instance, in a network with 200 edges and optimally configured HLDs for link acceptance threshold $t = 0.5$, the probability of having all link correctly detected is $3.3 \cdot 10^{-7}$.

At the beginning of work on biconnectivity testing in WMNs, it was difficult to accept this uncertainty and the fact that correctness of decisions cannot be guaranteed. This stands in particular contrast to known graph-theory approaches for biconnectivity testing where correctness of decisions is not questioned but taken as a mandatory characteristic of biconnectivity testing algorithms. The differentiation with graph theory with regard to correctness guarantees had to be made in order to create an approach capable of operation in wireless multi-hop networks. The goal of biconnectivity testing in WMNs is no longer to guarantee correctness but to maximize accuracy of decisions.

DIBADAWN is a distributed bridge and articulation point detection algorithm for wireless networks, where nodes are cooperating through message exchange. It is based on a class of Echo protocols that were altered so that they are better adapted to issues (in particular, the packet losses) encountered in WMNs but also to some of the benefits of wireless medium (e.g., the broadcasting nature of the communication channel). The articulation point detection is inspired by Tarjan's DFS.

The introduced changes improve the resilience of the algorithm to message losses and reduce its communication overhead. The original Echo algorithms have $O(e)$ message complexity of an execution round while DIBADAWN's message complexity is $O(n)$. Biconnectivity testing based on the proactive topology management protocols does not create communication overhead itself since it is executed at a single node, but the topology management as its prerequisite creates high overhead of $O(n^2)$.

Message losses and node failures cause faults in DIBADAWN and it may produce incorrect decisions, reducing its accuracy. The systematic treatment of failure modes has simplified the identification of possible corrective actions and analysis of their benefits

and drawbacks. Based on this analysis, DIBADAWN was extended to recognize some of the faults and to prevent their conversion to erroneous algorithm states.

Although beneficial, the detailed fault-error-failure analysis is seldom seen in WMN research. It provides clearer view at capabilities of an algorithm in presence of faults, it is easier to determine the usability limits of the algorithm as well as to develop its improvements.

It is difficult, if not impossible, to recognize all faults and remove all erroneous states from the algorithm. A set of voting rules organized in two tiers is proposed as fault masking mechanism for improvement of the accuracy of decisions. The voting rules operate on a set of latest outputs of DIBADAWN. The rules improve decision accuracy but they do not increase the communication overhead.

The approach developed in this work does not require the same perception of the network topology at each of its nodes, which is a precondition for bridge and articulation point detection algorithms which rely on accurate topology knowledge. The global knowledge is powerful and useful in networks with reliable communication, but it is counter-productive in systems loaded with uncertainty such are the WMNs. Its ideal is to know all about the network, while our approach aims to know much less, yet sufficient for biconnectivity testing. In a system loaded with uncertainty, this second approach is brings more benefits. In particular, the proposed voting rules exploit the diversity of knowledge about network's topology obtained from successive searches at each of network's nodes.

Without voting rules, DIBADAWN was constantly outperformed by the OLSR-based biconnectivity testing. This is expected since DIBADAWN uses $HLD(1,1)$ which in static networks has higher probability of errors in link detection than the combination of $HLD(2,2)$ and $HLD(3,2)$ that are used by OLSR (Figure 4.5(a) at page 52 and 4.7(a) at page 54).

The successive searches in DIBADAWN capture different information on network topology and provide different views of the topology to the decision-making nodes. Voting rules are able to utilize this information, to considerably improve the accuracy of decisions, and to outperform OLSR-based biconnectivity testing. The increase in accuracy brought by the voting rules is confirmed both in simulation and experiments. So for instance, the F-measure of direct DIBADAWN decisions for bridge detection in Motelab experiments is 0.69 while the trusted rule applied on last five algorithm execution results in F-Measure of 0.87. Similar improvements are observed in simulation. For instance, if Rayleigh fading is used and link acceptance threshold set to 0.316, DIBADAWN clearly outperforms OLSR-based detection: its F-measure for bridge detection is 36% to 69% higher, and F-measure of articulation point detection is about 30% higher than the F-measure of the OLSR.

In addition to the overall improvement in accuracy, voting rules provide adaptivity and flexibility to the presented approach. Without changes in functionality of DIBADAWN, a user can easily choose a rule that suites the needs of its applications. For example, the evaluation of DIBADAWN in Motelab experiments shows that users may choose between excellent precision (0.96), excellent recall (0.94) or balance of two (F-measure close to 0.8). Since DIBADAWN functionality remains unchanged, different nodes in a network may choose to use decisions of different rules without fear that it may affect other users of DIBADAWN. More than one application may be concurrently deployed at a single node and each of the applications may be using the rule that fulfills its needs,

because voting rules deliver their decisions independently and simultaneously.

The acceptance of the uncertainty in the system and possible inaccuracies in detection provided us considerable freedom in algorithm design and canceled some bounds that were imposed on the deterministic detection approaches. The analytical results for estimation of the average shortest cycle size in random geometry graphs have been obtained in Chapter 6. The average length of shortest cycle in RGGs converges to three already in networks of moderate density and it is less than eight in rather sparse networks (the average node degree of four). This information is used for further reduction of communication overhead in DIBADAWN: instead of searches that cover whole network, it may be possible to employ localized searches that cover only the k-neighborhood of a node. The developed RGG models are used to calculate the appropriate forward search radius for a network so the accuracy of bridge and articulation point detection remains comparable to accuracy of complete searches. The localized searches are particularly useful in mobile networks, where they can be executed with higher frequency, capturing topological changes faster and providing high accuracy of decisions.

Development of DIBADAWN as a biconnectivity testing algorithm that operates under uncertainty would remain a pure academical exercise if the common assumption in related work, that artificial topologies (such as uniform and grid placement) capture characteristics of real wireless multi-hop networks, would hold. This assumption was challenged through a series of measurements in community networks of Berlin and Leipzig.

The degree distribution observed in reality is considerably different than the degree distribution of placement models which are widely used in theory and simulation of WMNs. Particularly important for this work is that bridges and articulation points are much more numerous in reality than in artificial models. Comparison of topological properties between the measurements and the artificially generated topologies showed substantial differences, demonstrating that the models described in literature are not always sufficiently precise.

Various existing node placement generators for WMNs have been tested in unsuccessful attempts to recreate the topological characteristics of community networks from the case study. No matter which parameter combination was chosen, the properties of topologies produced by existing node placement algorithms could not match the properties of measured topologies. In order to fill the observed gap between theory and measurements, a node placement algorithm for realistic topologies (NPART) was developed and implemented as a freely available tool.

10.2. Outlook

This thesis enables research in three main directions: improvements of DIBADAWN, extension of NPART capabilities so that it becomes a universal node placement and mobility modeling tool, and applications of DIBADAWN for dependability improvements in WMNs.

10.2.1. Improvements of DIBADAWN and Voting Procedures

The proposed approach to distributed bridge and articulation point detection has been studied in detail. Its capability to operate in environments with channel fading was

demonstrated and as well as its flexibility in fulfilling needs of its users. Its flexibility and adaptivity to various deployment scenarios open numerous possibilities for its extension and further research.

Bayes classification rules from the second voting round form a potentially powerful rule set but they have not been presented in detail in this work because of their poor stability over different evaluation scenarios. Their good performance in Motelab experiments cannot be ignored so it is planned to **perform additional tests of Bayes rules** in other testbeds in order to determine whether Motelab setup was accidentally beneficial for them, or they are indeed applicable in real deployments.

The proposed approach relies on cooperative nodes and fault-free implementation of the algorithm. The experiences from implementation of the algorithm for Motelab experiments taught us that maintaining correct implementation is a tedious process, requiring numerous tests after every change in source code. Testing process is very complex since the errors in algorithm execution may originate from its implementation but also from faults caused by the packet losses. It is easy to envision that some of the implementations may not be as thoroughly tested as the implementation for the Motelab testbed, creating an additional source of faults for the biconnectivity testing. Finally, faults need not only occur because of packet losses and inadvertent implementation errors, but they may be deliberately injected by malicious users and nodes. Thus it is necessary to evaluate **the effect of Byzantine node behavior to detection accuracy** (this work has evaluated the detection accuracy in case of unpredictable and loss-prone behavior of communication channel).

Peronne [138] has studied several well-known routing algorithms and shown that all of them are highly susceptive even to very primitive attacks (such as antenna manipulation). DIBADAWN should be tested in presence of nodes with faulty implementation, primitive attacks from [138] and in presence of malicious nodes that deliberately target its functionality, attempting to maximize errors in detection process.

The preliminary studies indicate that DIBADAWN may successfully recognize nodes with inadvertent software errors: e.g., a node that does not directs messages to its parent in backward phase of algorithm is easily identified. Some other unintentional coding errors may be difficult or impossible to detect by other nodes (e.g., inaccurate pairing of backward messages, not participating in backward phase occasionally). DIBADAWN should be able to operate in presence of such faults since similar faults are caused by the unreliable communication channel. It is unclear at the moment what would be the behavior of DIBADAWN in presence of attackers that attempt to maximize the damage in detection process. It is planed to assess their impact on decision quality, and to develop countermeasures if possible (identify the faulty nodes).

Localized searches have been proposed in Chapter 6 and evaluated in Section 9.5 where it was shown that locality search has minor effects to the accuracy of decisions. In the evaluation, the search radius was calculated from the average node degree for all nodes in a network, regardless of the local node density.

Such application of the locality principle from Chapter 6 brings two implications. First, it is necessary to know the average node degree in the network in order to calculate the appropriate search radius. This requires a certain apriori knowledge that can be obtained through degree sampling. But as the time passes and network changes, this estimation may deviate from the actual state of the network. Second, averaging the node degree over whole network obscures the accurate local information. Thus, **an**

adaptive and localized approach to calculation of local-search radius should be developed.

Nodes are supposed to independently estimate their degree and use Equations 6.17, 6.25 or 6.32 to estimate the TTL value for DIBADAWN forward messages. Thus, in dense part of the network nodes should use smaller search radius and in sparser sections the search radius should be higher. Not only that this should improve search characteristics, but it also eliminates the apriori estimation of the average node degree in the network.

In presented version of the detection approach, nodes are exclusively using their own decisions, without cooperation with adjacent nodes: nodes cooperate in the exchange and forwarding of edge markings but each node keeps decision of its voting rules for itself. **The cooperation of nodes in the decision making process and exchange of decisions** may provide accuracy gains and open possibilities for development of a new class of cooperative voting rules.

Self-estimation of competencies. It has been explained in Chapter 5 that learning phase is necessary if algorithm is to use any of its weighted voting rules. The evaluation of sensitivity of the approach to altered weights in Section 9.4.3 has shown that DIBADAWN obtains good accuracy even with sub-optimal weight assignments. However, it would be beneficial to develop an online process that would be capable of weight estimation. It is likely that such approach would include intensive testing of links to assess which acceptance thresholds they pass and a partial proactiveness that exchanges the obtained link information in its $k-neighborhood$ and utilizing the locality of WMNs. That would decouple DIBADAWN from learning phase, increase its independence and speed-up its deployment into new environments. Additionally, the self-estimation of vote competence would enable weight assignment on node basis, instead of the globally determined and applied weights. Such autonomous node-centric weight assignment should improve accuracy of weighted voting rules.

10.2.2. Improvements of NPART

The node placement algorithm NPART provides clear improvements if compared with most of the existing artificial topology generators: its topologies are realistic and it is very flexible – changes in the input data suffice to alter the properties of created topologies.

NPART and accompanying metrics are suitable for generation of topologies that resemble community wireless mesh networks, but they may be unsuitable for other topology types. If NPART is to become universal, it must be capable of generating various classes of topologies such as the wireless sensor networks or industrial-application WMNs. We believe that NPART core is general enough to support generation of other topology classes, but it may be necessary to introduce new topology-quality metrics.

NPART guarantees the connectivity of generated topologies. In wireless mesh or sensor networks, this is a common property. In some application scenarios, such as Disruption Tolerant Networks, complete network connectivity is not encountered. For such scenarios, NPART may be used to produce connected subnetworks of a globally disconnected network.

NPART should be integrated with realistic node-mobility generators. In Chapter 9 NPART topologies were combined with the RWM applied to a subset of nodes to produce

a mobile network, but the quality of resulting model would have been better if we were able to introduce a realistic movement model. Mobility traces and measurements can be found in the CRAWDAD archive [6] so the basis for this extension exists.

Each of the possible extensions must preserve the process which was used during the development of NPART: measure a real system, analyze its properties, compare properties of the system with properties of the output of existing modeling tools. If they disagree, the algorithm has to be improved and evaluated until a new version of the tool is developed, capable of reproducing the properties observed in reality.

10.2.3. DIBADAWN Application Scenarios

Flexibility of DIBADAWN detection methodology with regard to its accuracy, easy selection of rules so that high precision, recall or F-measure are delivered, and its low overhead make it applicable in various scenarios. The low resource consumption allows it to operate even in networks with highly constrained resources.

Information on bridge and articulation point existence can provide considerable benefits to other communication protocols as it was demonstrated on the example of the reactive route discovery in Section 9.7. We believe that DIBADAWN can be used for other purposes as well:

Real-time traffic support: limited bandwidth of communication links and limited processing power of wireless nodes restrict the traffic throughput in a network. These limitations apply for all nodes and communication links in the network, but in absence of bridges and articulation points, even if some paths are congested another path with sufficient performance may be discovered (because of the shared communication medium and contention, the uncongested path may not be always found, even if network is 2-connected). However, if a traffic flow must traverse a bridge or use articulation point as a router on its path, alternative path does not exist and the congestion issues may escalate – bridges and articulation points are possible bottlenecks for the traffic that traverse them.

If there is no limitation on number of flows that share a congested communication link or overloaded node, they may all miss their arrival deadlines because of the long waiting times that are created at the congestion site. Rejection of incoming flows once an articulation point reaches its capacity limits may be beneficial for already existing flows. In order to preserve the traffic fairness in the network, a reservation or a priority system needs to be employed.

Partitioning prevention: decisions of DIBADAWN could be used for connectivity preservation of WMNs as it was already proposed for other biconnectivity testing algorithms [57] [80] [85] [123] [168]. If bridges and articulation points are detected, various corrective actions may be performed, depending on network type, its task, and willingness of a node to act in order to prevent the partitioning:

- Mobile agents for link strengthening: as it was proposed in [34] [63] nodes are ordered to move to specific locations in order to minimize the time intervals in which the network is 1-connected. Both [34] and [63] require location awareness and assume path-loss propagation model (so that it is easy to calculate if there will be a link between two nodes from their distance δ). It is questionable if these results can be applied in real mobile networks for three principal reasons:

  - Signal propagation and link existence do not follow the on-off pattern which is assumed in [34] [63]. Thus, even if nodes reach the desired location, they may not improve network connectivity because of obstacles that prevent establishment of a link.
  - Node may not be able to reach the target location.
  - Node cooperativeness may not be ideal. A node may reject to move to a requested location because the utility of preserving network connectivity is smaller than utility of action that it is already performing: e.g., it cannot be expected that a person (in a general purpose mobile WMN) remains at a given location for an hour in order to prevent network partitioning, if it will cause him/her to miss an appointment or to be late for work. Even in task-oriented networks, where such mundane concerns of day-to-day life can be ignored, it may be dangerous for nodes to go to, or to stay at the location which was proposed by the network management software (e.g., a fire fighter should not stay in a burning building if evacuation order has been issued).

  Obviously, an approach that includes node utility and effects of environment should be developed. Due to numerous external limitations that are imposed on a network, connectivity preservation cannot be guaranteed but a probabilistic metric should be associated with a given corrective action set.

- Instead of depending on external-to-network actions which may not perform as a partitioning prevention protocol expects, it may be better to use more predictable, technology oriented approaches, even if their benefits are somewhat limited. For instance, survivability of static networks may be improved through traffic/workload limitation at articulation points. Since they are the only connection between 2-connected network components, articulation points tend to spend their energy faster than ordinary nodes which can distribute the load. Thus, in order to extend network lifetime, an articulation point should select the traffic flows that are to be forwarded and those that are to be rejected. Idea is similar to the real-time traffic shaping, but instead of timeliness, the property of interest is energy preservation.

- If node energy is not an issue (e.g., nodes have permanent energy sources), articulation points and nodes incident to bridges may coordinate increase in transmission power in vicinity of the critical point in attempt to provide (at least) 2-connectivity of network. The basic idea is simple but it requires that a set of nodes reaches a common decision on used transmission power in order to avoid unidirectional links (most of routing protocols cannot operate with unidirectional links). Reaching a group consensus in a network with unreliable communication channel is complex and communication intensive task [105].

Replication of services and data in network: often it is not possible to prevent partitioning of a network, despite all the effort that is put in avoidance of such events. Hardware and software failures, energy depletion, or even physical destruction of nodes cannot be fully resolved by communication protocols.

Replication of critical data or network services in the network may reduce or even eliminate the effects of partitioning. Replication seems to be particularly beneficial in

networks that are to be used for the disaster management [9]. In disaster management, a network is typically used for tracking of rare-events. The network must be operational for prolonged periods of time during which nodes may fail because of hardware failures or energy depletion. Additionally, the event of interest (e.g., a forest fire or earthquake) may destroy some of the participating nodes.

In such conditions, it is beneficial to spatially replicate crucial network services and data, so that a failure of a subset of network nodes does not compromise execution of key tasks of a network. Random replication of resources over the network may not be satisfactory, as they may be placed in 1-connected network subcomponents. Information on bridge and articulation point existence may provide useful input not only for better placement of replicas (e.g., place replicas so that a node failure cannot create a network partition without an operational copy) but also for determining the number of replicas that are necessary in order to reach the desired robustness.

Improvement of service availability in logistic networks: The project Smart-Kanban [89] uses wireless technology in order to improve efficiency of processes and services in large warehouse and logistics centers. The containers used in the system are equipped with sensors (e.g., weight, acceleration) that capture state changes of items and transmit them to servers running the logistics software. The up-to-date knowledge of the system state allows server to initiate and coordinate various operations in logistics network. For instance, if a worker places a wrong (not requested by a customer) item to the container which is to be delivered to the customer, error may be detected by inappropriate weight of the packet and a corrective action can be immediately applied. In a traditional logistics system, error would be detected at customer's site, resulting in increased transportation costs and dissatisfaction of the customer.

The logistics is supported by a heterogeneous network: a wired backbone has limited coverage, so it is extended by a WLAN mesh network. Smart containers use IEEE 802.15 radios in order to improve their energy efficiency. Some of mesh nodes serve as gateways between 802.15 and 802.11 networks.

In order to make timely and accurate decisions in process runtime, the logistics software needs accurate and timely input from smart containers. Also, it needs to reliably deliver instructions on corrective actions to actors in the network. Node failures in this scenario are frequent. Smart containers may fail due to battery failure, nodes in the WLAN mesh have a high failure rate due to high oscillations in power network in industry. This errors are typically transient and require either reboot of a node or update of its software, but while the failed nodes are unavailable, network may be disconnected and servers may not get the data from smart containers.

It is envisioned to apply the automatic IT service availability assessment process [112][116][117] for offline evaluation of logistics services and processes. The availability assessment will be primarily used in wired part of the network and if target service-availability is not met, network structure should be changed in order to improve it. The presented biconnectivity testing algorithm will be applied in the wireless part of the network in order to increase efficiency of maintenance process. A maintenance worker is to be urgently dispatched to repair a node that provides 2-connectivity, while a non-critical node in dense part of the network may be repaired later. The combination of offline availability assessment and online detection of nodes and edges critical for network connectivity should result in increased network reliability and availability of services deployed in it, while keeping the maintenance costs within acceptable margins.

A. Analysis of the Exact Approach to Counting of Components in Random Geometric Graphs

Penrose provided in [136] integral expression for the probability p_k that a node belongs to the k-sized partition for unbounded Poisson point process, with intensity λ and for generalized connectivity function $g(x)$ where x is a distance vector. In the case where the connectivity function is indicator function $1_{|x|\leq R}$, the expression for p_k is:

$$p_k(\lambda) = \frac{\lambda^{k-1}}{(k-1)!} \int ... \int e^{-\lambda V(0,x_1,x_2,...,x_{k-1})} dx_1 ... dx_{k-1}.$$

where the integral is over all $x_1, ..., x_{k-1}$ such that the union of the radius R balls centered at $0, x_1, ..., x_{k-1}$ is connected, and $V(0, x_1, x_2, ..., x_{k-1})$ is the volume of that union.

This integral cannot be solved analytically, its calculation is complex and burdened with redundancies. Quintanilla and Torquato use in [142] a constructive paradigm to increase efficiency of numerical integration. They have shown that probability p_k that a node is part of a k-sized partition is:

$$p_k = \sum_{1=k_0<...<k_i=k} p_k(k_0, ..., k_i) \tag{A.1}$$

where

$$p_k(k_0, ..., k_i) = c(k_0, ..., k_i) \int_{B_1^{k_1-k_0}} dx_2 ... dx_{k_1} \int_{C(k_0,k_1)^{k_2-k_1}} dx_{k_1+1} ... dx_{k_2} \cdot$$

$$\int_{C(k_{i-2},k_{i-1})^{k_i-k_{i-1}}} dx_{k_{i-1}+1} ... dx_k \quad e^{-\lambda V_k(0,...,x_{k-1})} \tag{A.2}$$

and

$$c(k_0, ..., k_i) = \frac{\lambda^{k-1}}{(k_1-k_0)!...(k_i-k_{i-1})!} \tag{A.3}$$

where $B_i = B_R(x_i)$ is the ball with radius R centered at x_i and $C(k,l) = (B_{k+1} \cup ... \cup B_l) \backslash (B_1 \cup ... \cup B_k)$.

However, the authors note that it is inherently difficult to parameterize the domains of integration (even in unbounded case) and this problem is even more acute in the bounded case with possible interactions between boundaries and $V_k(0, ..., x_{k-1})$. But even if it would be possible to automatize the whole process, the overall complexity of it would prevent the application of the model to networks with large number of nodes.

In [142] was noted that the number of integrals grows fast with increase of N. It is proven here that the number of integrals has exponential growth:

First, we need to determine the number of solutions of the equation:

$$x_1 + x_2 + ... + x_u = z \tag{A.4}$$

where x_i are integers, $x_i > 0$ and $z \in \mathbf{N}$. Solutions are ordered, so for example the solution $1+2+3+4=10$ is not equal to the solution $2+1+3+4=10$. The number of solutions for this equation is equal to the number of different ways we can place $u-1$ breaks among z figures. For instance, the equation $1+2+3+4=10$ corresponds to the following ordering of $z=10$ figures and $u-1=3$ breaks :

$$O \quad | \quad OO \quad | \quad OOO \quad | \quad OOOO$$

The number of possible solutions is the number of different ways to choose $u-1$ locations out of $z-1$, i.e., $\binom{z-1}{u-1}$.

In our problem of counting the number of integrals one node is always fixed to be at the origin, therefore, we deal with $z = k-1$ nodes. It can be seen from Equation (A.1) that the total number of such integrals can be obtained as the sum of the number of solutions of the Equation (A.4) where the number u is varied from 1 (corresponding to the complete graph) to $k-1$ (the chain-like structure).

$$\sum_{i=1}^{k-1} \binom{k-1-1}{i-1} = \sum_{i=1}^{k-1} \binom{k-2}{i-1} = 2^{k-2}, \qquad k \geq 2.$$

In order to accurately calculate the partition distribution function for N nodes, every p_k has to be computed individually. The number of numerical integrals to compute in order to obtain the complete distribution function is:

$$\sum_{k=2}^{N} 2^{k-2} = 2^{N-1} - 1, \qquad N \geq 2. \tag{A.5}$$

As the calculation of an integral is already a complex operation and the total number of nodes in a wireless network can be measured in hundreds, it is necessary to use approximations instead of the exact solution for expected number of components of a RGG.

B. Detailed Proof of the Distributed Bridge and Articulation Point Detection Algorithm for Wireless Networks

This appendix provides a proof of correctness for the Distributed Bridge and Articulation Point Detection Algorithm for Wireless Networks. Figures B.1 and B.2 show the algorithm, written in pseudo-code.

The algorithm described in this appendix is simplified and as such it is not directly applicable to WMNs (e.g., it uses two global variables). However, it is devised for that application area and it can be easily adjusted for operation in WMNs. The global stack for node ordering which is used in this Appendix is exchangeable with the timeout scheme presented in Chapter 5, and the queue for processing order is implicitly known by nodes in a WMN by the time they are visited.

The version of the algorithm with higher message complexity is used in proofs for clarity reasons – it excludes the buffer management as performance improvement technique from the core functionality of the algorithm. There is no difference for the functionality of algorithm whether markings are sent one by one, or all in the same packet. Buffering of markings in the backward phase of detection algorithm is supported in its implementation in Section 5.3 and it was used both in simulations and experiments in Chapter 9. Also, instead of sending explicit BRIDGE markings as in Chapter 5 it is assumed here that all edges are pre-marked as bridges, and the algorithm removes markings from those that belong to cycles of the graph.

Lemma B.1 *The algorithm terminates.*

Proof:
The algorithm has two phases. We show that both of them cannot enter deadlocks or infinite loops.

Forward phase (method *constructTree*) is a generalized tree construction and cross-edge detection algorithm. If a vertex was not visited before, it is marked as visited and it adds all its adjacent nodes to the *processing* queue in an arbitrary order. Depending on the processing order of neighbors of a node, different trees may be constructed (e.g., DFS or BFS). A node initiates invocation of the *constructTree* method of its neighbors only if it has not been already visited. Therefore, the *expanding* part of the tree construction method of each vertex is invoked only once. Thus, forward phase of the algorithm finishes, each node p is visited $d(p)$ times and the list of cross-edges *crossEdges* is finite (due to finite number of edges in the graph).

In the backward phase, the methods *detectCycles* and *forwardMsg* are called once for each node in the graph (line 30, Figure B.1). The order of invocations is the opposite of order in which vertices were visited in the first phase. The ordering is guaranteed by the stack *nodeOrder*. The *detectCycle* method iterates through a finite list *crossEdges*

```
1   list of global variables used by the algorithm:
2     global stack nodeOrder;
3     global queue processing;
4   each vertex has following data structures:
5     list crossEdges;
6     boolean visited, isArticulationPoint;
7     queue backQueue;
8     list of message-sets for each neighbor msg_i
9     //(created dynamically upon neighbor first discovery and set to BRIDGE_ni);
10    list of adjacent nodes adj; //equivalent with list of incident edges
    vertex parent;
11  messages contain:
12    message initiator: source;
13    message unique id: id;
14    message forwarder from last hop: forwardedBy;
15  global:
16    for all v in V {
17      v.resetNodeFields(); each edge obtains unique BRIDGE mark;
18      visited=isArticulationPoint=false;
19    }
20    while(at least one node is not visited) {
21      select a node v that is not visited;
22      processing.enqueue(null, v, 0);
23      while (processing not empty)
24        (caller, callee, depth)=processing.dequeue();
25        calee.constructTree(caller, depth);
26      }
27    }
28    reorder stack nodeOrder by descending tree depths of nodes
29    while (nodeOrder not empty) {
30      v=nodeOrder.pop(); v.detectCycles(); forwardMsg();
31    }
32  constructTree(vertex visitor, integer treeDepth) {
33    if(not visited) {
34      visited=true;parent=visitor;
35      nodeOrder.push(this,treeDepth);
36      depth=treeDepth;
37      insert all w from adj() in arbitrary order in processing queue as tuple (this,w,depth+1);
38    } else if(visitor.parent!=this) // ignore edges belonging to tree
39      crossEdges.add((visitor));
40  }
```

Figure B.1.: DIBADAWN algorithm (1).

of a node and enqueues messages for its tree parent. For each detected cross-edge, a message is propagated down the constructed tree until it its pair is found (method *receiveBackMsg*). In worst case, there is no message pairing before root of the tree, so each message is propagated by the method *forwardMsg* down to the root. The distance to root is finite as well as the number of cross-edges, thus this phase of algorithm will finish in finite time. ♢

B.1. Bridge Detection

Lemma B.2 *Once a cross-edge* (p, q) *is detected, all edges belonging to cycle* $p, ..., HCA_{pq}, ..., q, p$ *are marked as NOBRIDGE. Status of edges that do not belong to this cycle is not changed.*

Proof:
This functionality is implemented in methods *detectCycles*, *receiveBackMsg* and *for-*

```
41  detectCycles() {
42      for each crossEdge in crossEdges {
43          if(this<crossEdge) uniqueCycleId=this::crossEdge;
44          else uniqueCycleId=crossEdge::this;
45          mark (this,crossEdge) as NOBRIDGE;
46          msg_crossEdge = msg_crossEdge ∪ {NOBRIDGE_uniqueCycleId}
47          receiveBackMsg(this, new backMessage(this, uniqueCycleId));
48      }
49  }
50  receiveBackMsg(vertex notifier, message msg) {
51      mark(this,notifier) as NOBRIDGE;
52      msg_notifier = msg_notifier ∪ {NOBRIDGE_this,notifier}
53      if backQueue.contains(msg) backQueue.remove(msg);
54      else backQueue.enqueue(msg);
55  }
56  forwardMsg() {
57      if(not backQueue.empty()) {
58          mark(this,parent) as NOBRIDGE;
59          for each msg in backQueue {
60              msg_parent = msg_parent ∪ {NOBRIDGE_uniqueCycleId}
61              parent.receiveBackMsg(this,msg);}
62  }
63  APdetection() {
64      create zero filled matrix closure[|adj()|][|adj()|]
65      for all pairs (i,j) from adj() × adj()
66          if(msg_i ∩ msg_j ≠ ∅) closure[i][j] = 1;
67      for all pairs (i,j) from adj() × adj()
68          if(closure[i][j] == 1)
69              for k in adj()
70                  if(closure[j][k] == 1) closure[i][k] = 1;
71      if (closure is not 1-matrix) isArticulationPoint=true;
```

Figure B.2.: DIBADAWN algorithm (2).

wardMsg. HCA cycles (Highest Common Ancestor, Definition 2.16) consist of edges that belong to the constructed tree: $p, ..., HCA_{pq}, ..., q$ and a single cross-edge pq. Two vertices independently detect cross-edge in the forward phase of the algorithm and mark it as a NOBRIDGE, and each of them enqueues a message for corresponding tree parents that describes the event (Line 47). The cross-edge has a unique identification that is used to determine the stopping point of the forwarding of corresponding NOBRIDGE marking.

A node that receives a NOBRIDGE marking message (method *receiveBackMsg*), reacts by marking the edge that delivered the message as NOBRIDGE. It either stores the message in internal queue *backQueue* if it is encountered for the first time. If the same marking already exists in the queue, both messages are deleted (the method "pairs" them). During cycle detection in method *detectCycles* both edges use the same rule to create cycle's identification so it will be identical, enabling its recognition and pairing. Pairing indicates that two identical messages that were traveling over different cycle segments have met. The pairing occurs only on HCA_{pq} and the whole HCA cycle is marked as NOBRIDGE since:

- Pairing is not possible higher than HCA in the tree: messages are forwarded only over tree edges (Line 61) resulting in disjoint tree paths for two copies of same message: $p, ..., HCA_{pq}$ and $q, ..., HCA_{pq}$. Also, as the message is forwarded down the path, the method *forwardMsg* marks edges that belong to paths as NOBRIDGE (Lines 51 and 58). The edge pq is marked in detection process, thus

whole cycle is marked. The statement is valid even if one of nodes is HCA of the cycle (an example is shown in Figure 5.8(a), page 93) since all messages pass through the queue *backQueue* where they can be paired.

- Messages cannot travel lower than HCA_{pq} since first time they meet, they are deleted (Line 53). The stack *nodeOrder* guarantees the accurate pairing of messages: level k in the tree is processed before any vertex at level $k-1$ can forward its messages. Thus, message cannot move lower than HCA_{pq} and mark edges outside the detected cycle. $\diamondsuit$

Comment on Lemma B.2: The crucial component of the algorithm is the stack that tracks node visits and orders the nodes by depth in the tree. It enforces the execution of backward phase in descending order of tree levels. It thus guarantees the correct pairing of NOBRIDGE messages and prevents their propagation to tree levels lower than the HCA of the corresponding cross-edge.

If execution order of backward phase is invalidated, a node may prematurely forward NOBRIDGE message down the tree invalidating in turn the correctness of the algorithm.

Although it is possible to detect them, cycles with two or more cross-edges are ignored in process of detection (e.g., cycle PQTS in Figure 5.4, page 80). Since each of cross-edges forms a cycle with a tree edge, inclusion of cycles with more than one cross-edge is not necessary and it would only increase number of messages in the backward phase.

Lemma B.3 *If an edge belongs to a cycle it is always marked as NOBRIDGE.*

Proof:
Let us observe a graph G(V,E) and a tree T in it, constructed by DIBADAWN. Three types of cycles can be distinguished in it:

1. Cycles consisting only of cross-edges
2. Cycles consisting of one cross-edge and two or more tree edges
3. Cycles consisting of two or more cross-edges and at least one tree edge

By definition, cross-edges are edges in the graph G that do not belong to the tree T and they close a cycle. The tree construction algorithm traverses each edge at least once, therefore all cross-edges are discovered and marked as NOBRIDGE by the algorithm. As the direct consequence, cycles consisting only of cross-edges are successfully marked as NOBRIDGE.

If a cycle consists of tree edges and only one cross-edge, all edges belonging to the cycle are marked as NOBRIDGE according to Lemma B.2.

The remaining case is that cycle C consists of multiple cross-edges and one or more tree edges:

Let us start from the tree T and add to it, one by one, only the cross-edges that are in C. For each cross-edge that is added, its whole HCA cycle is marked as NOBRIDGE (Lemma B.2). Once all cross-edges of the cycle C are added to the tree, the set of tree edges in the cycle C is a subset of all tree edges that are marked as NOBRIDGE by HCA cycles.

Let us assume the opposite: there exists an edge pq so that it belongs to the tree T and to the cycle C but it has not been marked as NOBRIDGE. That means it has no

cross-edge (that also belongs to cycle C) higher in the tree than itself nor incident in the tree (otherwise Lemma B.2 would not hold). It further means that the cycle C is closed only over edges that belong to the tree T, which is contradiction by the definition of a tree. $\diamondsuit$

Theorem B.1 *The algorithm marks each edge that belongs to a cycle as NOBRIDGE and each edge that does not belong to a cycle as BRIDGE.*

Proof:

Theorem B.1 is direct consequence of Lemma B.2 and B.3. In Lemma B.2 it is shown that if an edge belongs to a detected HCA cycle, it gets marked as NOBRIDGE while Lemma B.3 proves that sufficient number of cycles is detected by the algorithm so that an edge that belongs to a cycle cannot be left unmarked. Since all edges are marked as BRIDGE at the start of the algorithm, and after its termination all non-bridge edges are marked as NOBRIDGE, obviously all markings are correct.$\diamondsuit$

B.2. Detection of Articulation Points

The relation $\circledcirc$ (Definition 5.2, page 81) as implemented in DIBADAWN algorithm is equivalence relation since it fulfills three conditions required of equivalence relations:

- **Reflexivity**: $p \circledcirc p$ since $msg_p \neq \oslash => msg_p \cap msg_p \neq \oslash$
- **Symmetry** is consequence of associativity of $\cap$
- **Transitivity** is consequence of the transitive closure in method **APdetection**.

Lemma B.4 *If and only if edges incident to a node n belong to more than one equivalence class of relation $\circledcirc$, n is an articulation point [31].*

Proof:
The lemma is a consequence of the transitivity of relation $\circledcirc$: All edges that belong to one equivalence class are able to form cycles. Therefore, nodes that are adjacent to a node n and have edges that belong to same equivalence class of $\circledcirc$ must have paths that connect them independently of n (due to cycle existence). Thus, removal of node n cannot compromise connectivity of G. Opposite is valid for the same reason – since edges from different equivalence classes cannot form a cycle, they are not 2-connected and removal of n disconnects the graph G. $\diamondsuit$

Comment on Lemma B.4:

It was important to assign unique identification to bridges in DIBADAWN (Figure B.1, line 17) to prevent faulty calculation of equivalence classes: if all bridge markings are identical, the transitive closure unites them to a single equivalence class. If a node is incident only to bridges in the graph - result would be a single equivalence class, declaring that node is not an articulation point, which is obviously false (e.g., node M in Figure 5.4).

Theorem B.2 *Algorithm delivers sufficient information to a node so that it can correctly calculate equivalence classes of the relation $\circledcirc$. Thus, it also correctly detects articulation points.*

Proof:
The proof relies on Theorem B.1 which states that DIBADAWN provides the correct information whether a link is a bridge or not.

Depending on tree structure and node location in it, different cases may occur during algorithm execution. DIBADAWN resolves correctly each of the cases:

1) Pendant nodes send only one BRIDGE marking over the single link, resulting in one equivalence class and correct decision that node is not an articulation point.

2) If degree of a node is larger than one and at least one of incident edges has a BRIDGE marking, node is declared as an articulation point since it has at least two equivalence classes: one or more formed by NOBRIDGE markings (if they exist), and one for each of the incident bridges.

3) If a node is not an HCA and has no incident bridges (e.g., node B in Figure 5.4), it forwards all messages that it receives down the tree. Since all messages are forwarded, message sets form a single equivalence class for relation $\circledcirc$ (transitive closure over the down-tree edge).

4) If a node has no incident bridges and it is an HCA (e.g., node E in Figure 5.4), four subcases can be distinguished:

A node is not an articulation point, DIBADAWN declares it as an articulation point: This statement is equivalent to statement: there exists a cross-edge, higher in the tree than the node, that connects two (as calculated by DIBADAWN) equivalence classes, but DIBADAWN is unaware of it. Since every edge in the graph is traversed at least once, that hypothetical cross-edge is detected and NOBRIDGE messages are generated and sent down the tree towards HCA of the cycle. If the node is not aware of these NOBRIDGE markings, that means they were paired higher in the tree than this node. Since Lemma B.2 has shown that DIBADAWN does not pair the NOBRIDGE messages prematurely, this is a contradiction and DIBADAWN does not create false positives.

A node is an articulation point, DIBADAWN does not declare it as an articulation point: similar to the previous subcase – it can only occur if there has been incorrect message pairing, so that a NOBRIDGE message escapes down the tree (below its HCA), joining the edges from different equivalence classes to a single (erroneous) equivalence class over the link to the tree parent. This would require that DIBADAWN does not pair NOBRIDGE messages at their HCA, which impossible by Lemma B.2.

A node is an articulation point, DIBADAWN declares it as an articulation point: the method *APdetection* reuses well known and theoretically proven Warshall's algorithm [170] for transitive closure of relation $\circledcirc$ and for derivation of its equivalence classes. Thus, if DIBADAWN has the information which leads to conclusion that a node is articulation point, it will declare it. Also, it is already shown (Subcase 2) that it is not possible that an articulation point is not discovered, thus DIBADAWN has sufficient information to correctly execute the detection.

A node is not an articulation point, DIBADAWN does not declare it as an articulation point: The transitive closure allocates the edges incident to the node to a single equivalence class since they share a common cycle. The transitive closure is correctly implemented, so this is the desired and correct behavior. $\diamondsuit$

C. Precision and Recall of Random Markings

Instead of implementing, deploying and running a detection or decision algorithm, decision makers can instead decide randomly. Such approach is tempting since does not produce overhead and its implementation is trivial.

Let us assume that there are two mutually exclusive hypotheses h_0 and h_1 of which one is correct, according to some objective criterium. The probability of occurrence of hypothesis h_0 is p_0 and of $h_1 = 1 - p_0$. Each decision maker chooses hypothesis h_0 with probability p_{v0} and rejects it (chooses h_1) with probability $p_{v1} = 1 - p_{v0}$.

Without loss of generality, let us assume that hypothesis h_0 is the correct one. Since decision makers have uniform behavior (all of them choose h_0 with the same probability) it is possible to directly calculate probabilities of true and false positives/negatives. The confusion matrix, similar to one presented in Table 2.3 is shown in Table C.1.

The precision and recall are calculated in accordance with their definitions from Equations 2.3 and 2.4:

$$precision = \frac{TP}{TP + FP} = \frac{p_0 p_{v0}}{p_0 p_{v0} + (1 - p_0) p_{v0}} = p_0. \tag{C.1}$$

$$recall = \frac{TP}{TP + FN} = \frac{p_0 p_{v0}}{p_0 p_{v0} + (1 - p_{v0}) p_0} = p_{v0}. \tag{C.2}$$

If hypothesis h_1 is correct, same approach yields that precision is p_1 and recall p_{v1}.

For application scenario of bridge and articulation point detection, this means that the average precision of random markings is equal to average fraction of bridges/articulation points in the network, while the recall depends only on node's willingness to perform the marking. Since precision is independent of recall, the F-measure is maximized if $p_{v0} = 1$.

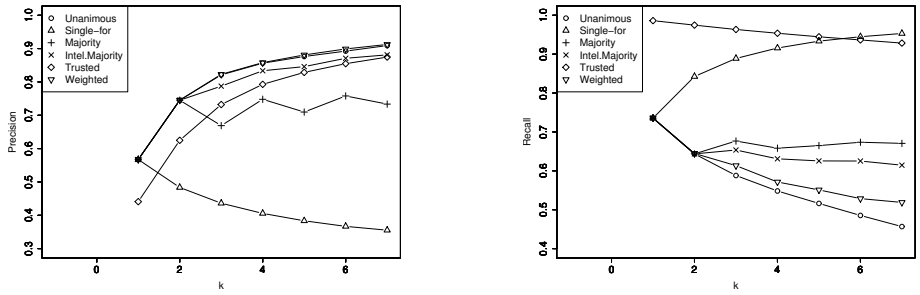
On example of Freifunk network in Berlin, the average precision for bridge detection is 0.15 with variable recall, and F-measure can take values up to 0.3. Such decisions are obtained for free, but their accuracy is much lower than accuracy provided by OLSR and DIBADAWN (Chapter 9).

	actual positive	actual negative
decided positive	true positive $p_0 p_{v0}$	false positive $(1 - p_0) p_{v0}$
decided negative	false negative $p_0(1 - p_{v0})$	true negative $(1 - p_0)(1 - p_{v0})$

Table C.1.: Confusion Matrix for Random Markings.

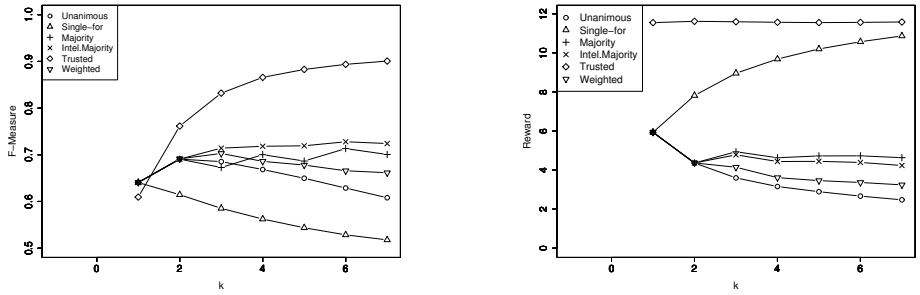
D. Detailed Evaluation Results

This appendix provides the detailed evaluation results from the Chapter 9. For every evaluation scenario that is presented in the appendix, four metrics are shown: precision, recall, F-measure and reward. While the evaluation in Chapter 9 was focused on voting rule behavior for $k = 5$, the results presented in this appendix show their behavior for larger set of values. In Motelab experiments, k has been varied between one and seven, and in simulation between one and ten.



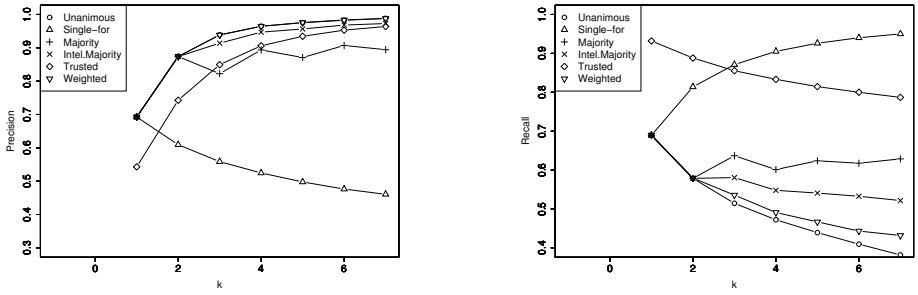
(a) Precision. (b) Recall.

Figure D.1.: Voting rules for bridge detection. Motelab experiments, link acceptance threshold $t = 0.1$.



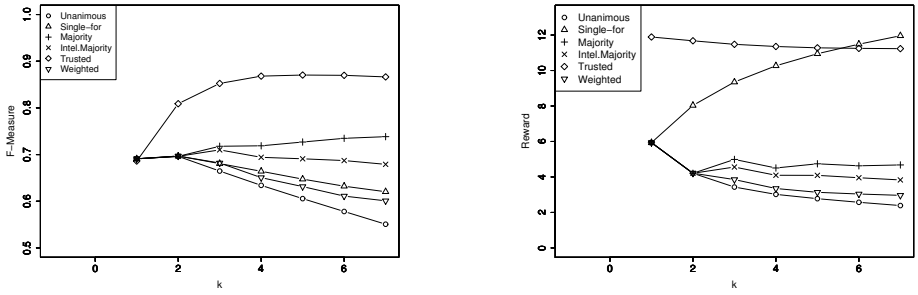
(a) F-measure. (b) Reward.

Figure D.2.: Voting rules for bridge detection. Motelab experiments, link acceptance threshold $t = 0.1$.(2)



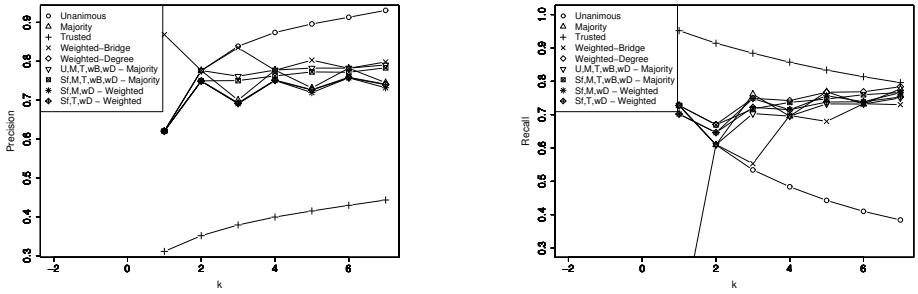
(a) Precision. (b) Recall.

Figure D.3.: Voting rules for bridge detection. Motelab experiments, link acceptance threshold $t = 0.316$.



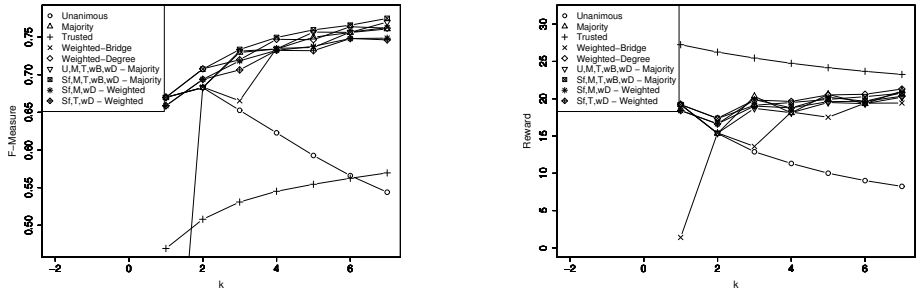
(a) F-measure. (b) Reward.

Figure D.4.: Voting rules for bridge detection. Motelab experiments, link acceptance threshold $t = 0.316$.(2)



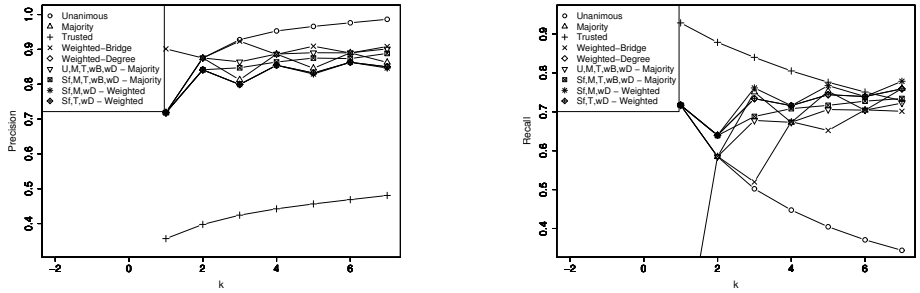
(a) Precision. (b) Recall.

Figure D.5.: Voting rules for articulation point detection. Motelab experiments, link acceptance threshold $t = 0.1$.



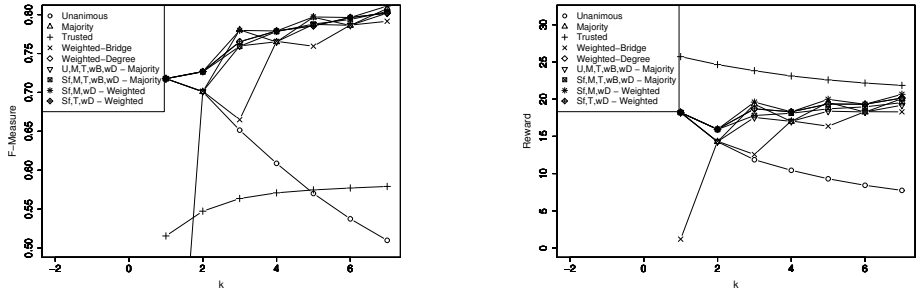
(a) F-measure. (b) Reward.

Figure D.6.: Voting rules for articulation point detection. Motelab experiments, link acceptance threshold $t = 0.1$.(2)



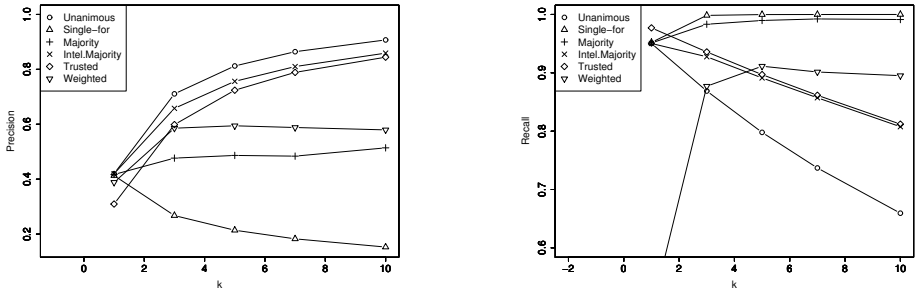
(a) Precision. (b) Recall.

Figure D.7.: Voting rules for articulation point detection. Motelab experiments, link acceptance threshold $t = 0.316$.



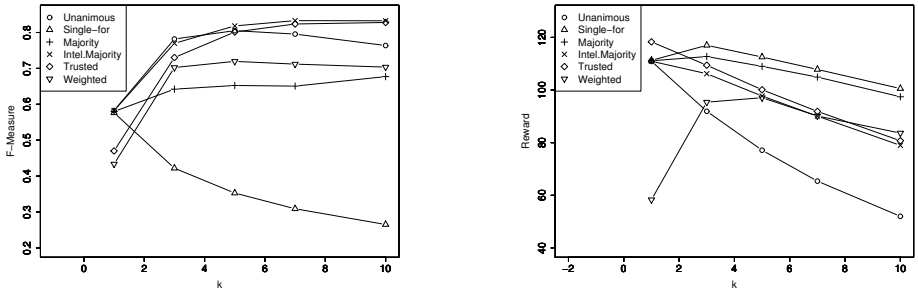
(a) F-measure. (b) Reward.

Figure D.8.: Voting rules for articulation point detection. Motelab experiments, link acceptance threshold $t = 0.316$.(2)



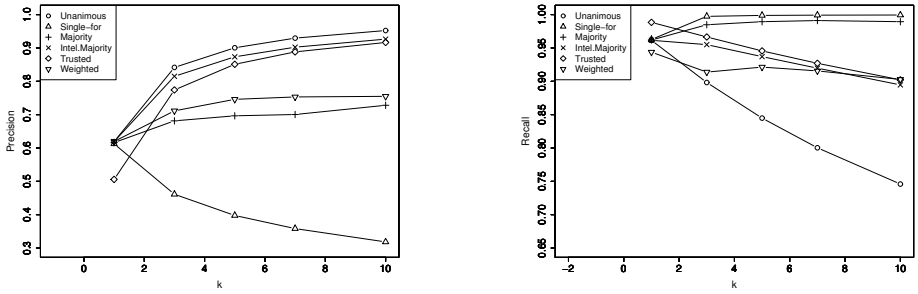
(a) Precision. (b) Recall.

Figure D.9.: Voting rules for bridge detection. NPART/Leipzig placement, Rayleigh fading. Link acceptance threshold $t = 0.1$.



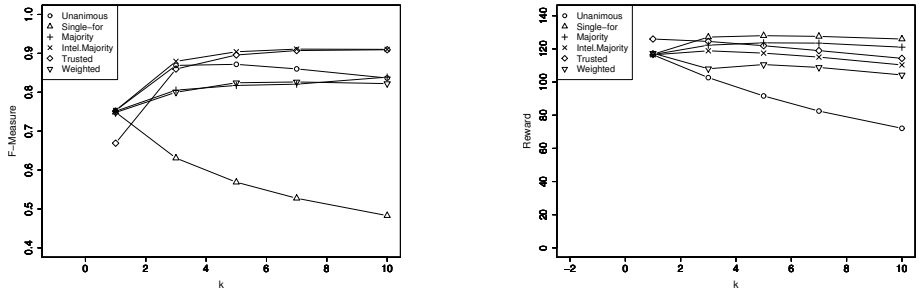
(a) F-measure. (b) Reward.

Figure D.10.: Voting rules for bridge detection. NPART/Leipzig placement, Rayleigh fading. Link acceptance threshold $t = 0.1$.(2)



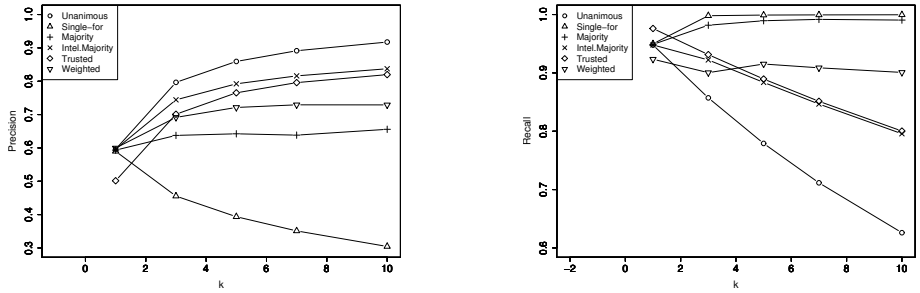
(a) Precision. (b) Recall.

Figure D.11.: Voting rules for bridge detection. NPART/Leipzig placement, Ricean fading. Link acceptance threshold $t = 0.1$.



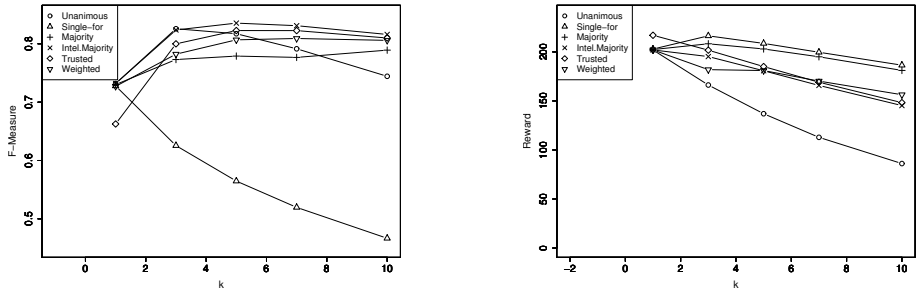
(a) F-measure. (b) Reward.

Figure D.12.: Voting rules for bridge detection. NPART/Leipzig placement, Ricean fading. Link acceptance threshold $t = 0.1$.(2)



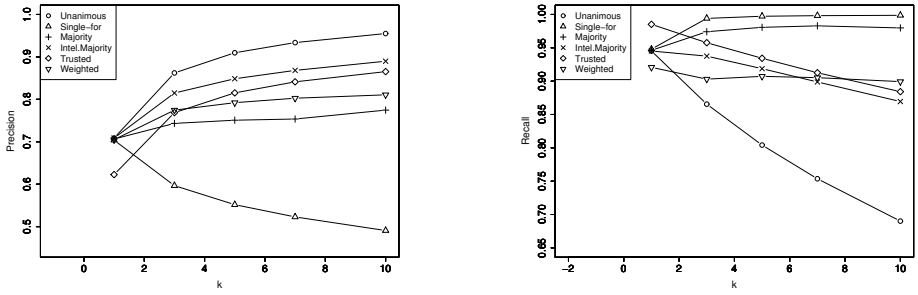
(a) Precision. (b) Recall.

Figure D.13.: Voting rules for bridge detection. NPART/Berlin placement, Rayleigh fading. Link acceptance threshold $t = 0.1$.



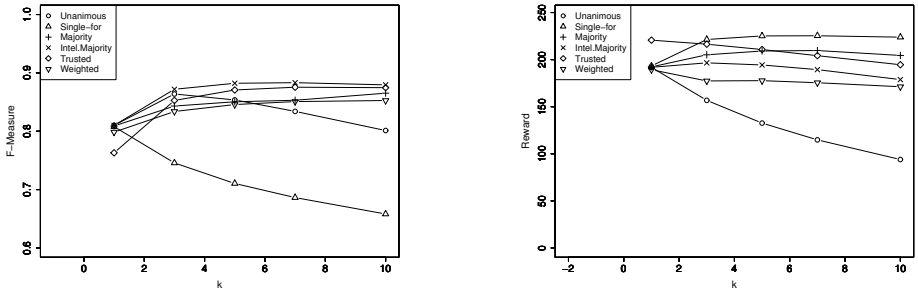
(a) F-measure. (b) Reward.

Figure D.14.: Voting rules for bridge detection. NPART/Berlin placement, Rayleigh fading. Link acceptance threshold $t = 0.1$.(2)



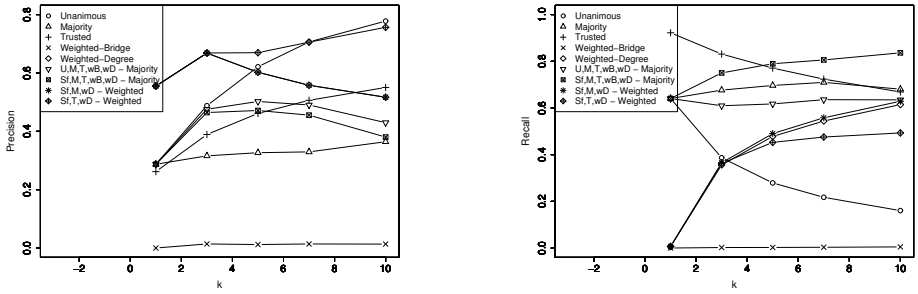
(a) Precision. (b) Recall.

Figure D.15.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading. Link acceptance threshold $t = 0.1$.



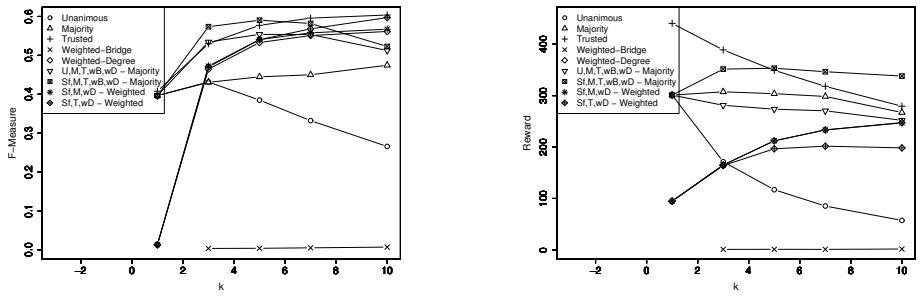
(a) F-measure. (b) Reward.

Figure D.16.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading. Link acceptance threshold $t = 0.1$.(2)



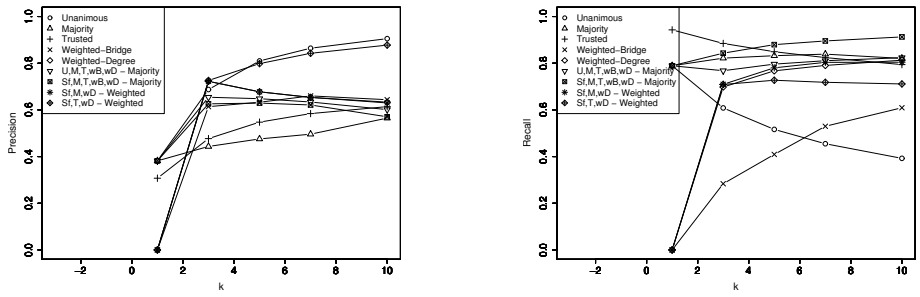
(a) Precision. (b) Recall.

Figure D.17.: Voting rules for articulation point detection. NPART/Leipzig placement, Rayleigh fading. Link acceptance threshold $t = 0.1$.



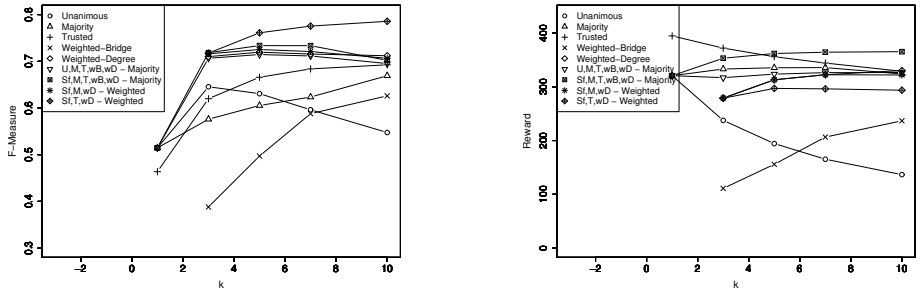
(a) F-measure. (b) Reward.

Figure D.18.: Voting rules for articulation point detection. NPART/Leipzig placement, Rayleigh fading. Link acceptance threshold $t = 0.1$.(2)



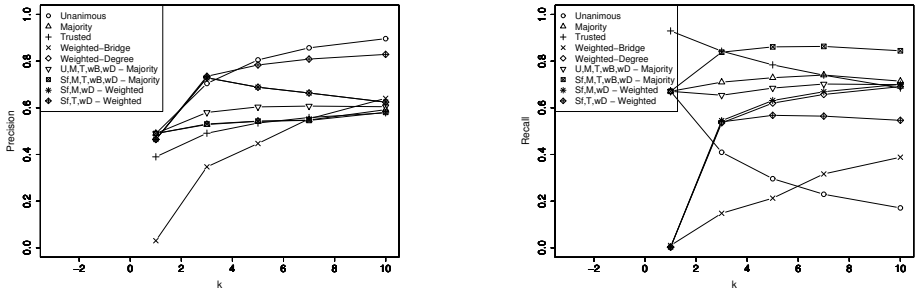
(a) Precision. (b) Recall.

Figure D.19.: Voting rules for articulation point detection. NPART/Leipzig placement, Ricean fading. Link acceptance threshold $t = 0.1$.



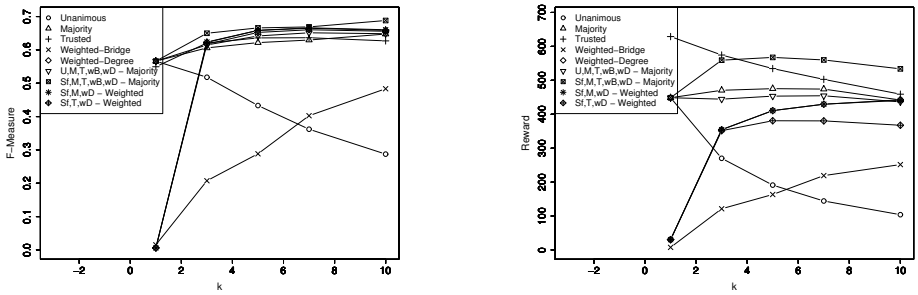
(a) F-measure. (b) Reward.

Figure D.20.: Voting rules for articulation point detection. NPART/Leipzig placement, Ricean fading. Link acceptance threshold $t = 0.1$.(2)



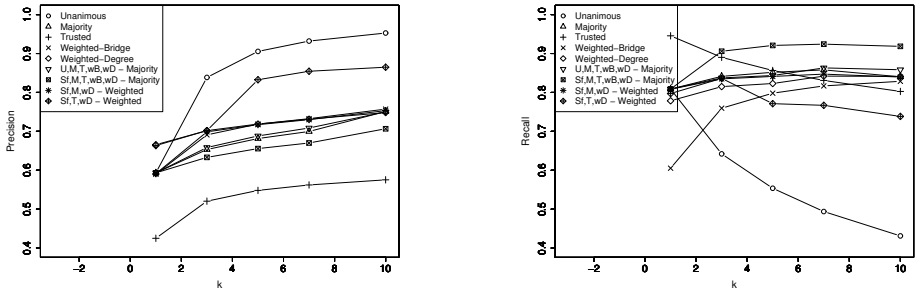
(a) Precision. (b) Recall.

Figure D.21.: Voting rules for articulation point detection. NPART/Berlin placement, Rayleigh fading. Link acceptance threshold $t = 0.1$.



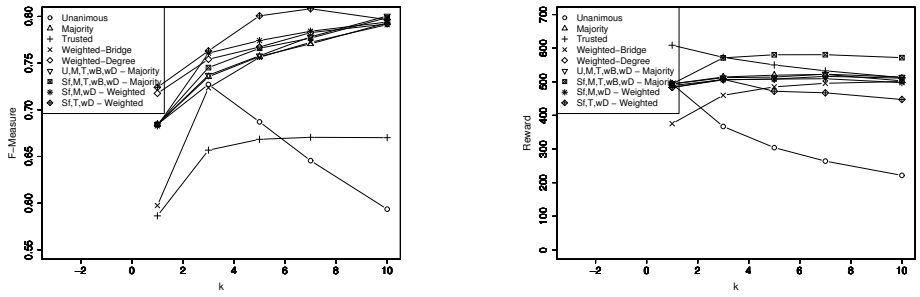
(a) F-measure. (b) Reward.

Figure D.22.: Voting rules for articulation point detection. NPART/Berlin placement, Rayleigh fading. Link acceptance threshold $t = 0.1$.(2)



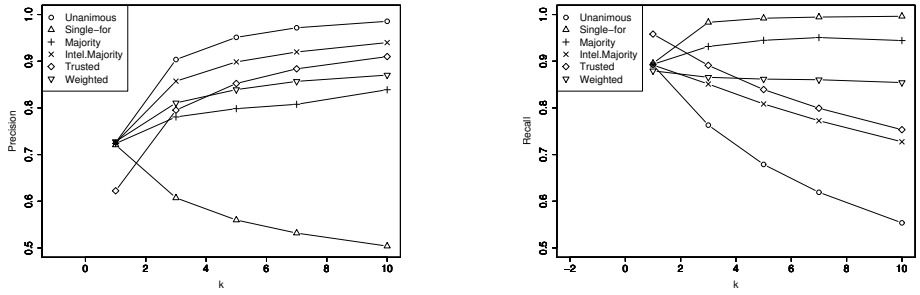
(a) Precision. (b) Recall.

Figure D.23.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading. Link acceptance threshold $t = 0.1$.



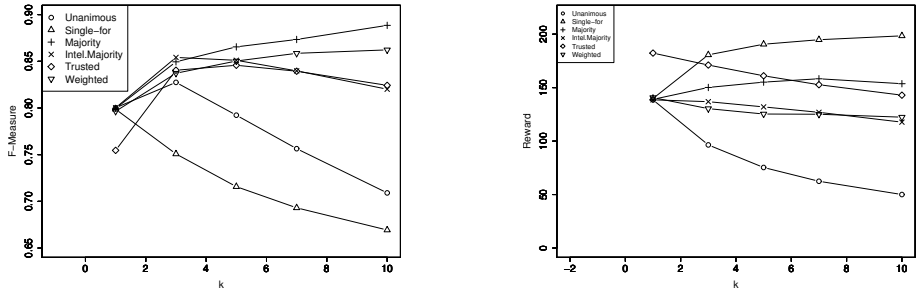
(a) F-measure. (b) Reward.

Figure D.24.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading. Link acceptance threshold $t = 0.1$.(2)



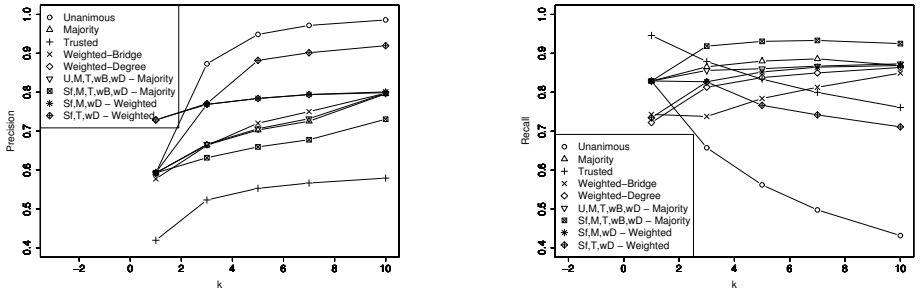
(a) Precision. (b) Recall.

Figure D.25.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading. Link acceptance threshold $t = 0.316$.



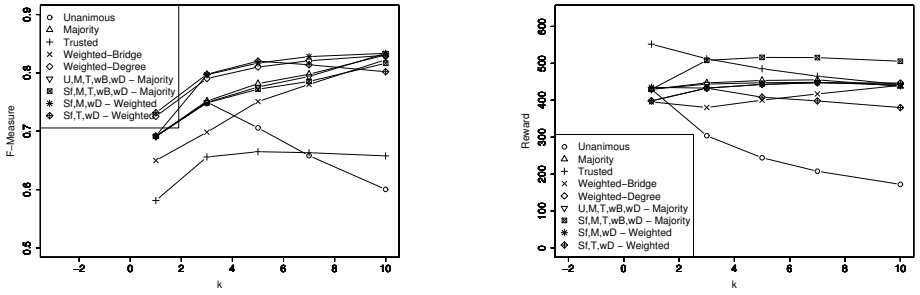
(a) F-measure. (b) Reward.

Figure D.26.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading. Link acceptance threshold $t = 0.316$.(2)



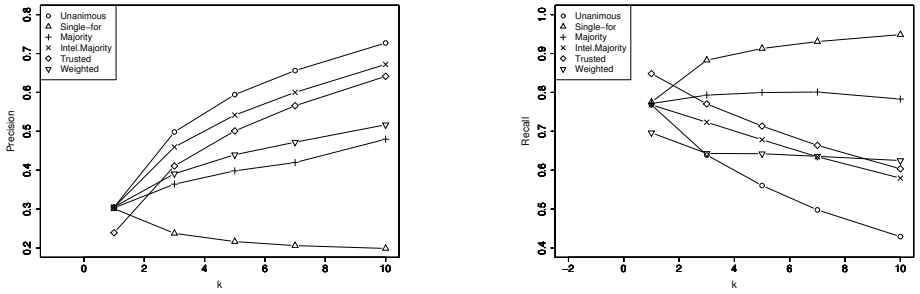
(a) Precision. (b) Recall.

Figure D.27.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading. Link acceptance threshold $t = 0.316$.



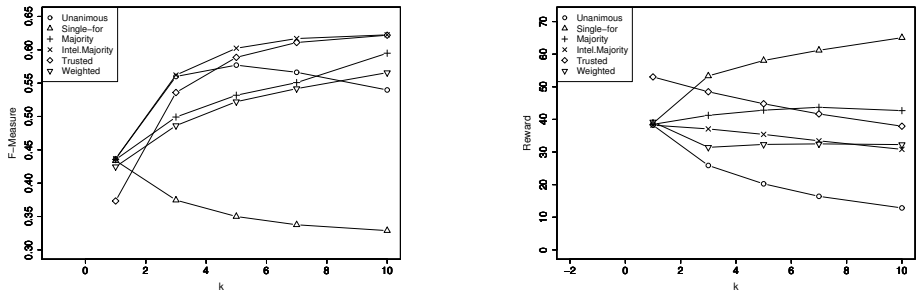
(a) F-measure. (b) Reward.

Figure D.28.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading. Link acceptance threshold $t = 0.316$.(2)



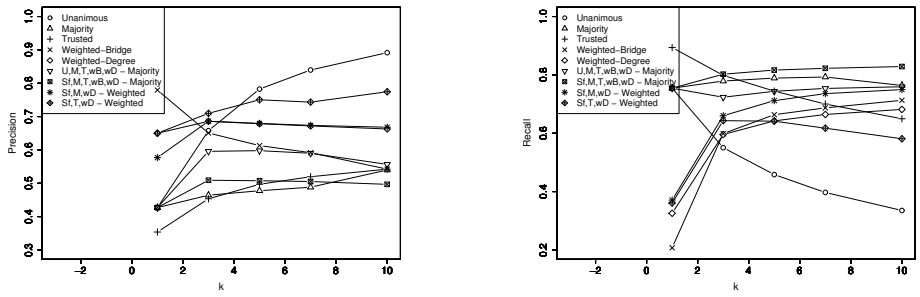
(a) Precision. (b) Recall.

Figure D.29.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading, mobility. Link acceptance threshold $t = 0.1$.



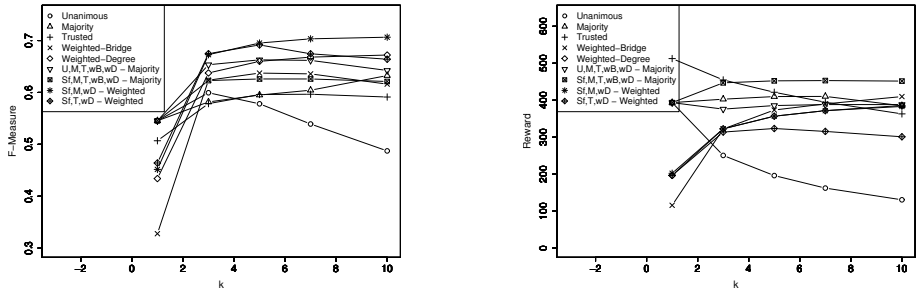
(a) F-measure. (b) Reward.

Figure D.30.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading, mobility. Link acceptance threshold $t = 0.1$.(2)



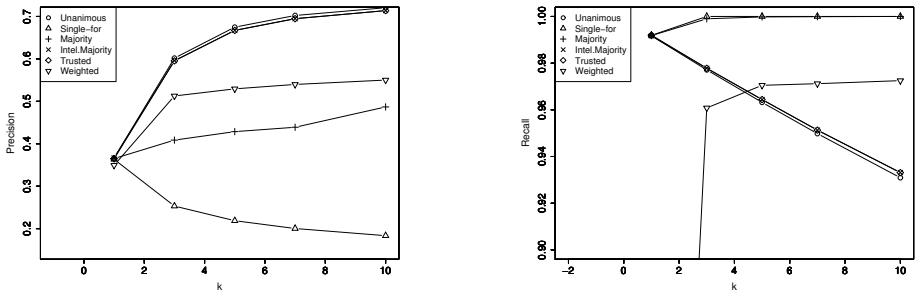
(a) Precision. (b) Recall.

Figure D.31.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading, mobility. Link acceptance threshold $t = 0.1$.



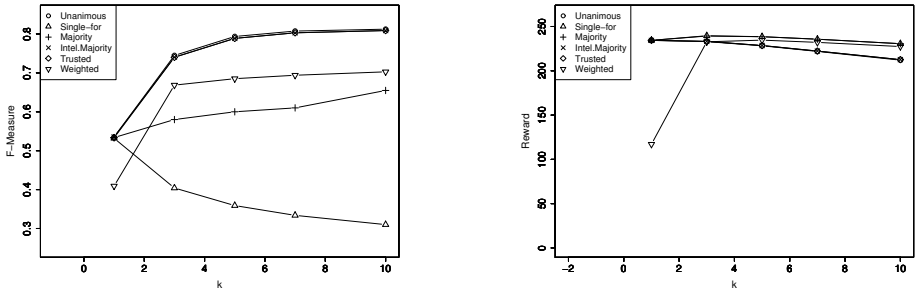
(a) F-measure. (b) Reward.

Figure D.32.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading, mobility. Link acceptance threshold $t = 0.1$.(2)



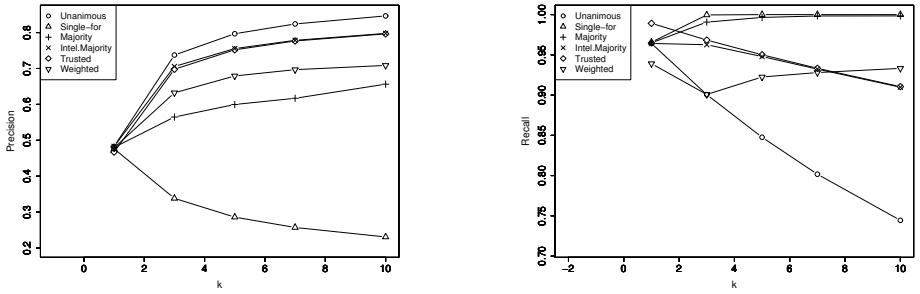
(a) Precision. (b) Recall.

Figure D.33.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading, forward search radius=2. Link acceptance threshold $t = 0.1$.



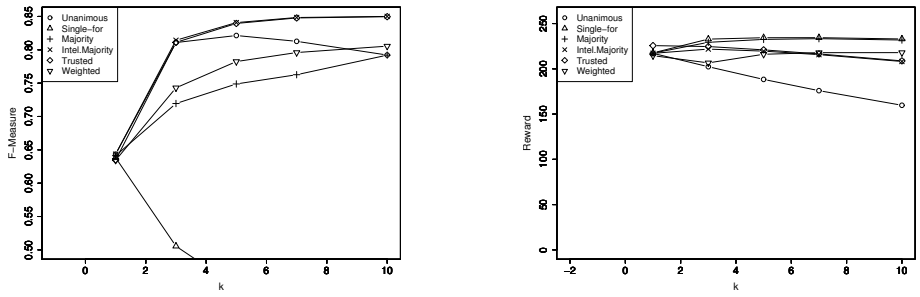
(a) F-measure. (b) Reward.

Figure D.34.: Voting rules for bridge detection.NPART/Berlin placement, Ricean fading,forward search radius=2.Link acceptance threshold $t = 0.1$.(2)



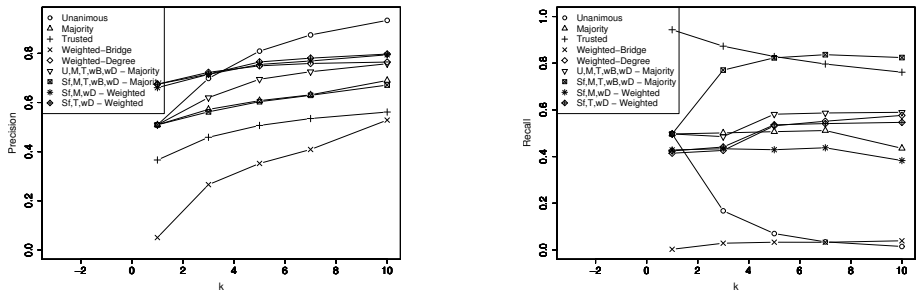
(a) Precision. (b) Recall.

Figure D.35.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading, forward search radius=4. Link acceptance threshold $t = 0.1$.



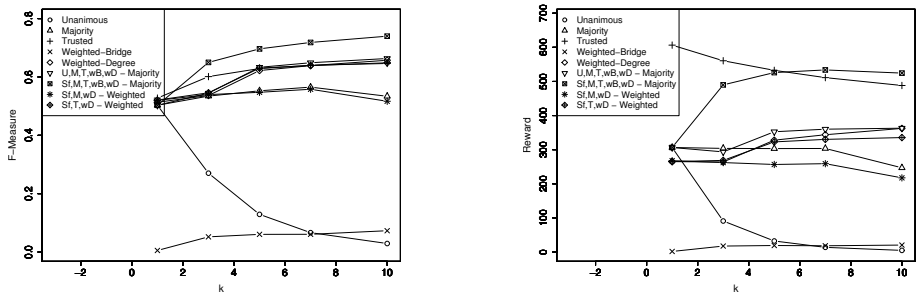
(a) F-measure. (b) Reward.

Figure D.36.: Voting rules for bridge detection.NPART/Berlin placement, Ricean fading,forward search radius=4.Link acceptance threshold $t = 0.1$.(2)



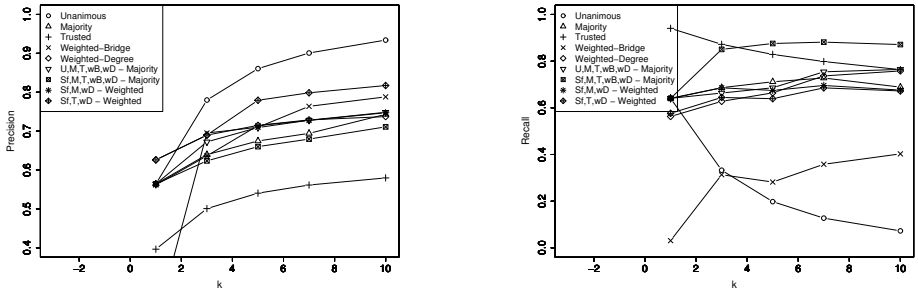
(a) Precision. (b) Recall.

Figure D.37.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading, forward search radius=2, threshold $t = 0.1$.



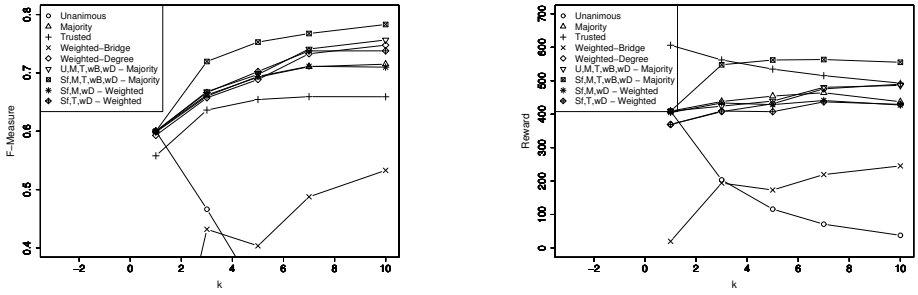
(a) F-measure. (b) Reward.

Figure D.38.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading, forward search radius=2, threshold $t = 0.1$.(2)



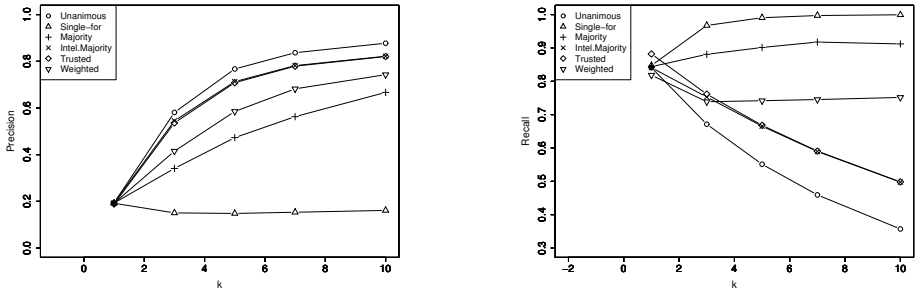
(a) Precision. (b) Recall.

Figure D.39.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading, forward search radius=4, threshold $t = 0.1$.



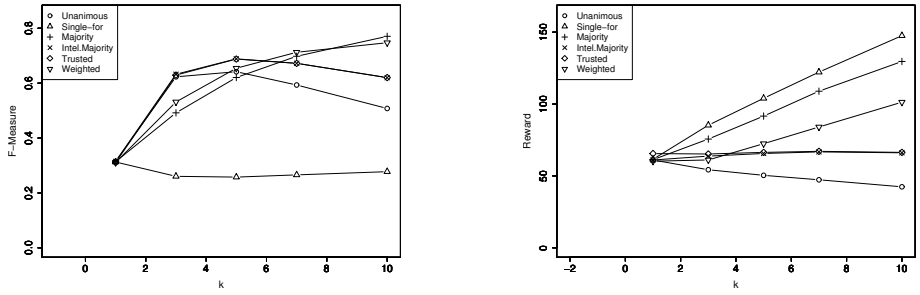
(a) F-measure. (b) Reward.

Figure D.40.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading, forward search radius=4, threshold $t = 0.1$.(2)



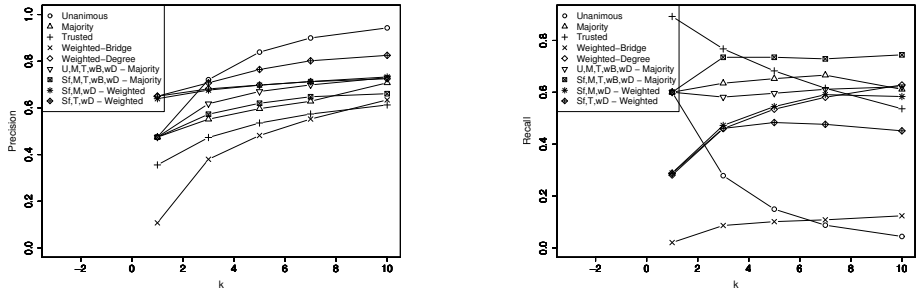
(a) Precision. (b) Recall.

Figure D.41.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading, mobility, threshold $t = 0.1$. Forward search TTL=4.



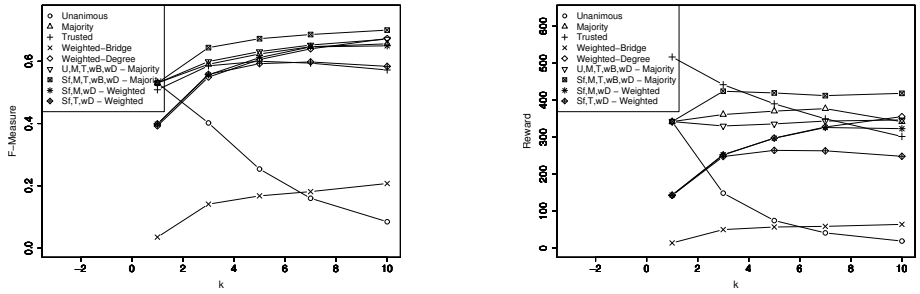
(a) F-measure. (b) Reward.

Figure D.42.: Voting rules for bridge detection. NPART/Berlin placement, Ricean fading, mobility, threshold $t = 0.1$. Forward search TTL=4. (2)



(a) Precision. (b) Recall.

Figure D.43.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading, mobility, threshold $t = 0.1$, search TTL=4.



(a) F-measure. (b) Reward.

Figure D.44.: Voting rules for articulation point detection. NPART/Berlin placement, Ricean fading, mobility, threshold $t = 0.1$, search TTL=4. (2)

Bibliography

[1] Car 2 Car, communication consortium, 2005. URL http://www.car-to-car.org/.

[2] Wizzy digital courier, 2006. URL http://www.wizzy.org.za/.

[3] MIT Roofnet, 2008. URL http://pdos.csail.mit.edu/roofnet.

[4] UCSB MeshNet, 2008. URL http://moment.cs.ucsb.edu/meshnet/.

[5] B.A.T.M.A.N. (Better Approach To Mobile Ad-hoc Networking) routing protocol, 2008. URL https://www.open-mesh.net/batman.

[6] CRAWDAD - A Community Resource for Archiving Wireless Data At Dartmouth, 2008. URL http://crawdad.cs.dartmouth.edu/.

[7] Berliner Freifunk-Community, 2008. URL http://olsrexperiment.de/.

[8] Leipziger Freifunk-Community, 2008. URL http://leipzig.freifunk.net/.

[9] Metrik - Modellbasierte Entwicklung von Technologien für selbstorganisierende dezentrale Informationssysteme im Katastrophenmanagement, 2008. URL http://metrik.informatik.hu-berlin.de/grk-wiki/index.php/Hauptseite.

[10] Hannover Free Network Map, 2008. URL http://map.freifunk-hannover.de/map.php.

[11] Merriam-Webster's Dictionary, Eleventh Edition, 2008. URL http://www.merriam-webster.com/.

[12] Motelab testbed, 2008. URL http://motelab.eecs.harvard.edu.

[13] MySQL, 2008. URL http://www.mysql.com.

[14] OLSR implementations, 2008. URL http://www.olsr.org/.

[15] Omnet++ simulator, 2008. URL http://www.omnetpp.org/.

[16] ORBIT: Open-access research testbed for next-generation wireless networks, 2008. URL http://www.orbit-lab.org/.

[17] Safespot project, 2008. URL http://www.safespot-eu.org.

[18] Secure Vehicular Communication - SEVECOM, 2008. URL http://www.sevecom.org.

[19] Jist/SWANS simulator, 2008. URL http://jist.ece.cornell.edu/.

[20] Aletheia - semantische föderation umfassender produktinformationen, 2009. URL `http://www.aletheia-projekt.de`.

[21] Kernel AODV, 2009. URL `http://w3.antd.nist.gov/wctg/aodv_kernel/`.

[22] AODV UU - Ad-hoc On-demand Distance Vector Routing for real world and simulation, 2009. URL `http://core.it.uu.se/core/index.php/AODV-UU`.

[23] I. Aad, J. Hubaux, and E. W. Knightly. Denial of service resilience in ad hoc networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking, MOBICOM 2004*, 2004.

[24] N. Aboudagga, M. T. Refaei, M. Eltoweissy, L. DaSilva, and J. Quisquater. Authentication protocols for ad hoc networks: Taxonomy and research issues. In *Proceedings of the 8th International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems, MSWiM 2005*, 2005.

[25] A. Aguiar and J. Gross. Wireless Channel Models. Technical report, TU Berlin, 2003.

[26] M. Aguilera, W. Chen, and S. Toueg. Failure detection and consensus in the crash-recovery model. *Distributed Computing*, 13(2):99–125, 2000.

[27] E. Akkoyunlu, K. Ekanadham, and R. Huber. Some constraints and tradeoffs in the design of network communications. In *SOSP '75: Proceedings of the Fifth ACM Symposium on Operating Systems Principles*, pages 67–74, 1975.

[28] G. Alonso, E. Kranakis, R. Wattenhofer, and P. Widmayer. Probabilistic protocols for node discovery in ad-hoc, single broadcast channel networks. In *Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, 2003.

[29] J. B. Andersen, T. S. Rappaport, and S. Yoshida. Propagation measurements and models for wireless communication channels. *IEEE Communications Magazine*, 33:42–49, 1995.

[30] A. Avizienis, J. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *1*, 1:11–33, 2004.

[31] S. Baase. *Computer Algorithms.* Addison-Wesley, 1988.

[32] J. Barnat and I. Černá. Distributed breadth-first search LTL model checking. *Formal Methods in System Design*, 29(2):117–134, 2006.

[33] R. Barr, Z. Haas, and R. van Renesse. Jist: An efficient approach to simulation using virtual machines. *Software Practice & Experience*, 35:539–576, 2005.

[34] P. Basu and J. Redi. Movement control algorithms for realization of fault tolerant ad hoc robot networks. Technical Report 8359, BBN Technologies, 2002.

[35] S. Bates. On edges and connectivity in ad hoc networks. In *Proceedings of Global Telecommunications Conference, GLOBECOM '04*, volume 6, pages 3588– 3593, 2004.

[36] C. Bettstetter. On the minimum node degree and connectivity of a wireless multihop network. In *Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2002.

[37] C. Bettstetter and C. Hartmann. Connectivity of wireless multihop networks in a shadow fading environment. *Wireless Networks*, 11:571–579, 2005.

[38] C. Bettstetter and S. Konig. On the message and time complexity of a distributed mobility-adaptive clustering algorithm in wireless ad hoc networks. In *Proceedings of European Wireless*, 2002.

[39] C. Bettstetter, G. Resta, and P. Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Transactions on Mobile Communications*, 2:257–269, 2003.

[40] C. Bettstetter, M. Gyarmati, and U. Schilcher. An Inhomogeneous Spatial Node Distribution and its Stochastic Properties. In *Proceedings of International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM 2007)*, 2007.

[41] C. Bezuidenhout, G. Grimmett, and A. Loeffler. Percolation and minimal spanning trees. *Journal of Statistical Physics 92*, 1998.

[42] D. Blough and H. Brown. The broadcast comparison model for on-line fault diagnosis in multicomputer systems: Theory and implementation. *IEEE Transactions on Computers*, 48(5):470–493, 1999.

[43] L. Booth, J. Bruck, M. Cook, and M. Franceschetti. Ad hoc wireless networks with noisy links. In *IEEE International Symposium on Information Theory*, 2003.

[44] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7:609–616, 2001.

[45] R. Braden. *Requirements for Internet Hosts – Communication Layers*, 1989. URL `http://tools.ietf.org/rfc/rfc1122.txt`.

[46] C. Cachin. Modeling complexity in secure distributed computing. In *Future directions in distributed computing: research and position papers*, pages 57–61, 2003.

[47] G. Calinescu and P. Wan. Range assignment for high connectivity in wireless ad hoc networks. *Ad-Hoc, Mobile, and Wireless Networks*, LNCS 2865:235–246, 2003.

[48] T. Camp, J. Boleng, and L. Wilcox. Location information services in mobile ad hoc networks. In *Proceedings of the IEEE International Conference on Communications ICC 2002.*, volume 5, 2002.

[49] P. Carmi, M. Segal, M. Katz, and H. Shpungin. Fault-tolerant power assignment and backbone in wireless networks. In *Proceedings of Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom Workshops)*, 2006.

[50] J. Cartigny, F. Ingelrest, D. Simplot-Ryl, and I. Stojmenovic. Localized lmst and rng based minimum-energy broadcast protocols in ad hoc networks. *Ad Hoc Networks*, 3:1–16, 2005.

[51] J. K. Cavers. *Mobile Channel Characteristics*. Kluver Academic Publishers, 2000.

[52] I. Chakeres and E. Belding-Royer. The utility of hello messages for determining link connectivity. In *Proceedings of the 5th International Symposium on Wireless Personal Multimedia Communications*, 2002.

[53] I. Chakeres and E. Belding-Royer. Aodv routing protocol implementation design. In *Proceedings of the 24th International Conference on Distributed Computing Systems (Workshops)*, 2004.

[54] I. Chakeres and C. Perkins. *Dynamic MANET On-demand (DYMO) Routing (IETF Draft)*, March 2007. URL `www.ietf.org/internet-drafts/draft-ietf-manet-dymo-08.txt`.

[55] E. Chang. Echo algorithms: Depth parallel operations on general graphs. *IEEE Transactions on Software Engineering*, 8:391–401, 1982.

[56] L. Chao and H. Aiqun. Reducing the message overhead of aodv by using link availability prediction. In *Proceedings of the Third International Conference on Mobile Ad-Hoc and Sensor Networks*, 2007.

[57] K. Chen, S. H. Shah, and K. Nahrstedt. Cross-layer design for data accessibility in mobile ad hoc networks. *Wireless Personal Communications*, 21, 2002.

[58] M. Cinque, D. Cotroneo, G. De Caro, and M. Pelella. Reliability requirements of wireless sensor networks for dynamic structural monitoring. In *Proceedings of the International Workshop on Applied Software Reliability (WASR 2006)*, 2006.

[59] T. Clausen and P. Jacquet. *The Optimized Link State Routing protocol (RFC 3626)*, October 2003. URL `www.ietf.org/rfc/rfc3626.txt`.

[60] A. Clementi, P. Penna, and R. Silvestri. Hardness results for the power range assignment problem in packet radio networks. *Randomization, Approximation, and Combinatorial Optimization. Algorithms and Techniques*, LNCS 1671:197–208, 1999.

[61] M. Conti. Multi-hop ad hoc networking: from theory to reality. In *Proceedings of the 10th International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems, MSWiM 2007*, 2007.

[62] N. Cressie. *Statistics for Spatial Data*. John Wiley & Sons, 1993.

[63] S. Das, H. Liu, A. Nayak, and I. Stojmenovic. A localized algorithm for bi-connectivity of connected mobile robots. In *Proceedings of The Third International Conference on Mobile Ad-hoc and Sensor Networks*, 2007.

[64] D. DeCouto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, MOBICOM 2003*, pages 134–146, 2003.

[65] *Transport Statistics Great Britain - 2006 Edition*. Department For Transport, 2006.

[66] L.P. Devroye. The expected size of some graphs in computational geometry. *Computers and Mathematics with Applications*, 15, 1988.

[67] E. Dijkstra and C. Scholten. Termination detection for diffusing computations. *Information Processing Letters*, 11(1):1–4, 1980.

[68] U. Dralle and A. Reinefeld. A distributed algorithm for optimal concurrent communication and load balancing in parallel systems. In *Proceedings of the International Conference and Exhibition on High-Performance Computing and Networking*, pages 588–600. Springer-Verlag, 1997.

[69] E. Ertin, A. Arora, R. Ramnath, M. Nesterenko, V. Naik, S. Bapat, V. Kulathumani, M. Sridharan, H. Zhang, and H. Cao. Kansei: a testbed for sensing at scale. In *Proceedings of The Fifth International Conference on Information Processing in Sensor Networks, IPSN 2006.*, 2006.

[70] K. Fall and K. Varadhan. *the ns2 Manual*, 2007. URL `http://www.isi.edu/nsnam/ns/ns-documentation.html`.

[71] L. Fang, W. Du, and P. Ning. A Beacon-Less Location Discovery Scheme for Wireless Sensor Networks. In *Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM2005*, 2005.

[72] W. Feller. *An Introduction to Probability Theory and Its Applications, Vol. 1.* Wiley, 1968.

[73] M. Franceschetti, L. Booth, M. Cook, R. Meester, and J. Bruck. Continuum percolation with unreliable and spread-out connections. *Journal of Statistical Physics*, 118, 2005.

[74] N. Freris and P. Kumar. Fundamental limits on synchronization of affine clocks in networks. In *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007.

[75] E. Gansner and S. North. An open graph visualization system and its applications to software engineering. *Software Practice and Experience*, 30(11):1203–1233, 2000.

[76] M. Garey and D. Johnson. *Computers and intractability, a guide to the theory of NP-completeness*. W.H.Freeman, 1979.

[77] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *Proceedings of Programming Language Design and Implementation*, 2003.

[78] E.N. Gilbert. Random plane networks. *Journal of the Society of Industrial and Applied Mathematics*, 9, issue 4:533–543, 1961.

[79] V. Giruka and M. Singhal. Hello protocols for ad-hoc networks: overhead and accuracy tradeoffs. In *Proceedings of Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*, 2005.

[80] D. Goyal and J. Jr. Caffery. Partitioning avoidance in mobile ad hoc networks using network survivability concepts. In *International Symposium on Computer and Communications (ISCC)*, 2002.

[81] S. Graham and P. R. Kumar. Time in general-purpose control systems: The control time protocol and an experimental evaluation. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, 2004.

[82] M. Gudmundson. Correlation model for shadow fading in mobile radio systems. *Electronic Letters*, 27:2145–2146, 1991.

[83] E. Hamida, G. Chelius, and E. Fleury. Revisiting neighbor discovery with interferences consideration. In *Proceedings of the Third ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor and Ubiquitous Networks*, 2006.

[84] V. Handziski, A. Köpke, A. Willig, and A. Wolisz. TWIST: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In *Proceedings of the Second International Workshop on Multi-hop Ad Hoc Networks: From Theory to Reality*, 2006.

[85] M. Hauspie, D. Simplot, and J. Carle. Partition detection in ad-hoc networks using multiple disjoint paths set. In *Proceedings of the First International Workshop on Objects Models and Multimedia Technologies (OMMT)*, 2003.

[86] M. Hefeeda. Forest fire modeling and early detection using wireless sensor networks. Technical report, Simon Fraser University, 2007.

[87] A. Herms, G. Lukas, and S. Ivanov. Realism in design and evaluation of wireless routing protocols. In *Proceedings of the First International Workshop on Mobile Services and Personalized Environments*, 2006.

[88] P. Ibach and J. Zapotoczky. Vorteile und beschränkungen durch open source lizenzierung im projekt magicmap. In *Beiträge der 37. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Informatik trifft Logistik*, 2007.

[89] P. Ibach, P. Troger, and B. Milic. Selbst-organisierendes autarkes kanban-system auf basis eigenintelligenter, vernetzter und ultrakostengünstiger sensorknoten, 2009. URL `http://www.rok.informatik.hu-berlin.de/rok/research/smart_kanban/inde%x_html`.

[90] M. Impett, M. Corson, and V. Park. A receiver-oriented approach to reliable broadcast in ad hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2002)*, 2002.

[91] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1):2–16, 2003.

[92] S. Ivanov. and E. Nett. Fault-tolerant coverage planning in wireless networks. In *Proceedings of IEEE Symposium on Reliable Distributed Systems*, pages 175–184, 2008.

[93] R. Jansen, S. Hanemann, and B. Freisleben. Proactive distance-vector multipath routing for wireless ad hoc networks. In *Proceedings of Communication Systems and Networks, CSN2003*, 2003.

[94] J.W. Jaromczyk and G.T. Toussaint. Relative neighborhood graphs and their relatives. *Proceedings of the IEEE*, 80:1502–1517, 1992.

[95] T. Johansson and L. Carr-Motyckova. Reducing interference in ad hoc networks through topology control. In *The Third ACM International Workshop on Foundations of Mobile Computing*, 2005.

[96] D. Johnson, D. Maltz, and Y. Hu. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (RFC 4728)*, February 2007. URL `www.ietf.org/rfc/rfc4728.txt`.

[97] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *Proceedings of 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2002.

[98] P. Karp and H. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *6th ACM Conference on Mobile Computing and Networking*, 2000.

[99] L. Kirousis, E. Kranakis, D. Krizanc, and A. Pelc. Power consumption in packet radio networks. *Theoretical Computer Science*, 1-2:289–305, 2000.

[100] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. In *Proceedings of the ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM)*, pages 78–82, 2004.

[101] E. Krause. *Taxicab Geometry: An Adventure in Non-Euclidean Geometry*. Dover, New York, 1986.

[102] D. Krioukov, F. Chung, K. Claffy, M. Fomenkov, A. Vespignani, and W. Willinger. The workshop on internet topology (wit) report. *Computer Communication Review*, 37(1):69–73, 2007.

[103] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad-hoc routing: of theory and practice. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC'03)*, 2003.

[104] L. Kuncheva. On the optimality of naive bayes with dependent binary features. *Pattern Recognition Letters*, 27:830–837, 2006.

[105] L. Lamport, M. Pease, and R. Shostak. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4:382–401, 1982.

[106] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *Proceedings of the First international Conference on Embedded Networked Sensor Systems*, 2003.

[107] X. Li, P. Wan, Y. Wang, and C. Yi. Fault Tolerant Deployment and Topology Control in Wireless Networks. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2003.

[108] Y. Liang and C. Rhee. An optimal algorithm for finding biconnected components in permutation graphs. In *Proceedings of the 1995 ACM 23rd Annual Conference on Computer Science CSC95*, 1995.

[109] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. In *Proceedings of the Firrst International Workshop on Service Assurance with Partial and Intermittent Resources, SAPIR 2004*, 2004.

[110] X. Liu and M. Haenggi. Toward Quasiregular Sensor Networks: Topology Control Algorithms for Improved Energy Efficiency. *IEEE Transactions on Parallel and Distributed Systems*, 17:975–986, 2006.

[111] M. J. de Caritat, Marquis de Condorcet. Essai sur l'application de l'analyse á la probabilité des décisions rendues á la pluralité des voix. http://gallica.bnf.fr/ark:/12148/bpt6k417181, 1785.

[112] M. Malek, B. Milic, and N. Milanovic. Analytical Availability Assessment of IT Services. In *Proceedings of The 5th International Service Availability Symposium (ISAS 2008)*, 2008.

[113] D. Marandin. Performance evaluation of failed link detection in mobile ad hoc networks. In *Proceedings of The Third Annual Mediterranean Ad Hoc Networking Workshop*, 2004.

[114] M. McGlynn and S. Borbash. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In *Proceedings of the Second ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2001.

[115] Mesh Dynamics. *Network-Centric Warfare and Wireless Communications*, 2008.

[116] N. Milanovic, B. Milic, and M. Malek. Modeling Business Process Availability. In *Proceedings of the IEEE International Workshop on Methodologies for Non-functional Properties in Services Computing*, 2008.

[117] N. Milanovic, B. Milic, and M. Malek. *Developing Effective Service Oriented Architectures: Concepts and Applications in Service Level Agreements, Quality of Service and Reliability*, chapter Model-based Methodology and Framework for Assessing Service and Business Process Availability. IGI Global, 2009.

[118] B. Milic and M. Malek. Dropped edges and faces' size in gabriel and relative neighborhood graphs. In *Proceedings of The Third IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2006)*, 2006.

[119] B. Milic and M. Malek. Analyzing Large Scale Real-World Wireless Multihop Network. *IEEE Communication Letters*, 11(7), 2007.

[120] B. Milic and M. Malek. Adaptation of Breadth First Search Algorithm for Cut-edge Detection in Wireless Multihop Networks. In *Proceedings of 10th ACM-IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM 2007)*, 2007.

[121] B. Milic and M. Malek. *Handbook of Wirless Ad Hoc and Sensor Networks*, chapter 1: Properties of wireless multihop networks in theory and practice. Springer Verlag, 2009.

[122] B. Milic and M. Malek. NPART - Node Placement Algorithm for Realistic Topologies in Wireless Multihop Network Simulation. In *Proceedings of the Second International Conference on Simulation Tools and Techniques (SIMUTools)*, 2009.

[123] B. Milic, N. Milanovic, and M. Malek. Prediction of partitioning in location-aware mobile ad hoc networks. In *Proceedings of the Hawaii International Conference on System Sciences, HICSS-38, (Minitrack on Quality of Service in Mobile and Wireless Networks)*, 2005.

[124] M. Mock, R. Frings, S. Trikaliotis, and E. Nett. Clock synchronization for wireless local area networks. In *Proceedings of 12th Euromicro Conference on Real-Time Systems (Euromicro-RTS 2000)*, 2000.

[125] T. Mukherjee, S. Gupta, and G. Varsamopoulos. Analytical Model for Optimizing Periodic Route Maintenance in Proactive Routing for MANET's. In *Proceedings of International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM 2007)*, 2007.

[126] J. Mullen and H. Huang. Impact of multipath fading in wireless ad hoc networks. In *Proceedings of The Second ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, 2005.

[127] M. Naserian and K. Tepe. Game theoretic approach in routing protocol for wireless ad hoc networks. *Ad Hoc Networks*, 2008.

[128] E. Ngai, Y. Zhou, M. R. Lyu, and J. Liu. Reliable Reporting of Delay-Sensitive Events in Wireless Sensor-Actuator Networks. In *Proceedings of The Third IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2006.

[129] S. Ni, Y. Tseng, Y. Chen, and J. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the International Conference on Mobile Computing and Networking, MOBICOM 1999*, 1999.

[130] S. Nikolopoulos and L. Palios. On the parallel computation of the biconnected and strongly connected co-components of graphs. *Discrete Applied Mathematics*, 155:1858–1877, 2007.

[131] S. Nitzan and J. Paroush. Optimal decision rules in uncertian dichotomous choice situations. *International Economic Review*, 23:289–297, 1982.

[132] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, 1992.

[133] B. On, H. Shin, M. Choi, and M. Park. A hierarchical ack-based protocol for reliable multicast in mobile networks. In *Proceedings of the 8th IEEE International Conference on Networks (ICON 00)*, 2000.

[134] F. Onat and I. Stojmenovic. Generating random graphs for wireless actuator networks. In *Proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2007.*, 2007.

[135] A. Panchenko and A. Thümmler. Efficient phase-type fitting with aggregated traffic traces. *Perform. Eval.*, 64(7-8):629–645, 2007.

[136] M. D. Penrose. On a continuum percolation model. *Advances in Applied Probability, 23*, 1991.

[137] C. Perkins, E. Belding-Royer, and S. Das. *Ad hoc On-Demand Distance Vector (AODV) Routing (RFC 3561)*, July 2003. URL `www.ietf.org/rfc/rfc3561.txt`.

[138] L. F. Perrone. Could a caveman do it? the surprising potential of simple attacks. *IEEE Security and Privacy*, 5:74–77, 2007.

[139] T. Plesse, J. Lecomte, C. Adjih, M. Badel, and P. Jacquet. Olsr performance measurement in a military mobile ad-hoc network. In *OLSR Performance Measurement in a Military Mobile Ad-hoc Network*, 2004.

[140] R. Punnoose, P. Nikitin, and D. Stancil. Efficient simulation of ricean fading within a packet simulator. In *Proceedings of the Vehicular Technology Conference*, 2000.

[141] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying for flooding broadcast messages in mobile wireless networks. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS)*, 2002.

[142] J. Quintanilla and S. Torquato. Clustering in a continuum percolation model. *Advances in Applied Probability, 29*, 1997.

[143] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, 2005. URL `http://www.R-project.org`.

[144] T. Rappaport. *Wireless Communications, Principles and Practice*. Prentice Hall, 2008.

[145] P. Reinecke and K. Wolter. Phase-type approximations for message transmission times in web services reliable messaging. In Samuel Kounev, Ian Gorton, and Kai Sachs, editors, *Performance Evaluation – Metrics, Models and Benchmarks*, volume 5119, pages 191–207. Springer, 2008.

[146] S. A. Roach. *The Theory of Random Clumping.* Mentuen, London, 1968.

[147] R.A. Sahner, K.S. Trivedi, and A. Puliafito. *Performance and Reliability Analysis of Computer Systems.* Kluwer Academic Publishers, 2002.

[148] P. Samar and S. Wicker. Link dynamics and protocol design in a multihop mobile environment. *IEEE Transactions on Mobile Computing*, 5(9):1156–1172, 2006.

[149] E. Schoch, M. Feiri, F. Kargl, and M. Weber. Simulation of ad hoc networks: ns-2 compared to jist/swans. In *Proceedings of the First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems*, 2008.

[150] L. Shapley and B. Grofman. Optimizing group judgmental accuracy in the presence of interdependencies. *Public Choice*, 43:329–343, 1984.

[151] R. Sombrutzki, A. Zubow, M. Kurth, and J. Redlich. Self-organization in community mesh networks: The Berlin RoofNet. In *Proceedings of IEEE OpComm2006*, 2006.

[152] M. Souryal and N. Moayeri. Channel-adaptive relaying in mobile ad hoc networks with fading. In *Proceedings of The First IEEE Conference on Sensor and Ad Hoc Communications and Networks, SECON2004*, 2004.

[153] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of SIGCOMM 2005*, 2005.

[154] I. Stojmenovic, M. Russell, and B. Vukojevic. Depth first search and location based localized routing and qos routing in wireless networks. In *Proceedings of the Proceedings of the 2000 International Conference on Parallel Processing (ICPP 2000)*, 2000.

[155] I. Stojmenovic, A. Nayak, and J. Kuruvila. Design guidelines for routing protocols in ad hoc and sensor networks with a realistic physical layer. *IEEE Communications Magazine*, 43:101–106, 2005.

[156] A. Subbiah and D. Blough. Distributed diagnosis in dynamic fault environments. *IEEE Transactions on Parallel Distributed Systems*, 15(5):453–467, 2004.

[157] M. Takai, J. Martin, and R. Bagrodia. Effects of wireless physical layer modeling in mobile ad hoc networks. In *Proceedings of the Second ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2001.

[158] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1:146–160, 1972.

[159] M. Thomas, S. Phand, and A. Gupta. Using group structures for efficient routing in delay tolerant networks. *Ad Hoc Networks*, 2008.

[160] J. Thomsen and D. Husemann. Evaluating the use of motes and tinyos for a mobile sensor platform. In *Proceedings of Parallel and Distributed Computing and Networks*, 2004.

[161] A. Thümmler, P. Buchholz, and M. Telek. A novel approach for phase-type fitting with the em algorithm. *IEEE Transactions on Dependable and Secure Computing*, 3(3):245–258, 2006.

[162] R. Thurimella. Sub-linear Distributed Algorithms for Sparse Certificates and Biconnected Components. *Journal of Algorithms*, 23(1):160–179, 1997.

[163] G. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12:261–268, 1980.

[164] *Tropos MetroMesh Proven: Metro-Scale Wi-Fi in Chaska, MN.* Tropos Networks, 2005.

[165] Y. Tseng, S. Ni, and E. Shih. Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. In *Proceedings of the The 21st International Conference on Distributed Computing Systems ICDCS2001*, 2001.

[166] A. Vahdat and D. Becker. Epidemic routing for partially disconnected ad hoc networks. Technical Report CS200006, Duke University, USA, 2000.

[167] N. Vicari. Models of WWW-Traffic: a Comparison of Pareto and Logarithmic Histogram Models. Technical Report 198, Institute of Computer Science, University of Wuerzburg, 1998.

[168] K.H. Wang and B. Li. Group mobility and partition prediction in wireless ad-hoc networks. In *Proc. of IEEE International Conference on Communications (ICC 2002)*, 2002.

[169] Y. Wang, I. Stojmenovic, and X. Li. Bluetooth scatternet formation for single-hop ad hoc networks based on virtual positions. *Journal of Internet Technology*, 6:43–52, 2005.

[170] S. Warshall. A theorem on boolean matrices. *Journal of the ACM (JACM)*, 9: 11–12, 1962.

[171] W. Wei and A. Zakhor. Path selection for multi-path streaming in wireless ad hoc networks. In *Proceedings of International Conference on Image Processing*, 2006.

[172] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: a wireless sensor network testbed. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, 2005.

[173] D. B. West. *Introduction to Graph Theory.* Prentice Hall, 1996.

[174] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 194–205, 2002.

[175] Y. Wu and Y. Li. Construction algorithms for k-connected m-dominating sets in wireless sensor networks. In *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2008.

[176] B. Xiao, G. W. Liu, and J. Z. Si-Lu. An improvement for local route repair in mobile ad hoc networks. In *Proceedings of the 6th International Conference on ITS Telecommunications*, 2006.

[177] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *Proceedings of IEEE INFOCOM*, 2003.

[178] H. Zhang. The optimality of naive bayes. In *Proceedings of The 17th International FLAIRS Conference*, 2004.

[179] Y. Zhang and Q. Huang. Adaptive tree: A learning-based meta-routing strategy for sensor networks. In *Proceedings of IEEE Consumer Communications and Networking Conference, CCNC2006*, 2006.

[180] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2005.

[181] S. Zhu, W. Wang, and C. Ravishankar. A New Power-Efficient Scheme to Deliver Time-Sensitive Data in Sensor Networks. In *Proceedings of The Third IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2006.

[182] H. Zimmermann. OSI reference model - the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28:425–432, 1980.

List of Figures

List of Tables

Selbständigkeitserklärung

Ich erkläre hiermit, dass

- ich die vorliegende Dissertationsschrift mit dem Titel Distributed Biconnectivity Testing in Wireless Multi-hop Networks selbständig und ohne unerlaubte Hilfe angefertigt habe;
- ich mich nicht bereits anderwärts um einen Doktorgrad beworben habe oder einen solchen besitze;
- mir die Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät II der Humboldt-Universität zu Berlin, gemäß Amtl. Mitteilungsblatt Nummer 34/2006, bekannt ist.

Berlin, den 27.8.2009. Bratislav Milic