TANDEM: Prioritizing Wireless Communication for Robust Industrial Process Control

Roman Naumann, Stefan Dietzel, Laura Wartschinski, Ben Schumacher, and Björn Scheuermann
Humboldt-Universität zu Berlin, Germany
Email: {roman.naumann, stefan.dietzel, wartschl, ben.schumacher}@hu-berlin.de, scheuermann@informatik.hu-berlin.de

***Abstract*—The trend towards smart factories necessitates protocols for wireless transmission of production process information. Protocols must transfer process information timely even with temporary wireless interference on the factory floor and regardless of machines' location in the network topology. TANDEM is a topology-independent wireless multi-hop network protocol that implements in-network prioritization to provide a useful, early approximation of process information. Tailored to often required time series data transmission, TANDEM applies a time-frequency transformation to prioritize more valuable information parts, decouples multi-hop communication steps to improve robustness, and uses a periodic acknowledgment overhearing mechanism to save transmissions. We analytically discuss reliability and fairness aspects and show using simulations that TANDEM's prioritization is beneficial to obtain an early preview of sensor information even in large factories. In addition, we implement TANDEM on industrial grade hardware and provide a performance evaluation.**

I. Introduction

The vision of "smart factories" promises to automate the hitherto manual and tedious process of observing machine and process parameters and leveraging them to optimize and alter configurations for better output quality. Reliable and efficient wireless communication protocols are an enabling factor for this vision. Traditionally, machine parameters were only available locally and shown on displays mounted on production machines. Operationalizing these parameters was a manual process where machine operators continuously observed and adapted parameters during new production lines' setup phase.

Making available sensor values to a centralized server allows for the collected information's algorithmic exploitation using machine learning approaches, which can shorten setup times and reduce the production of faulty parts. Moreover, large factories' machine operators can use the centralized information to gain a quick overview of all machines' current state. Using wireless communication for the transmission of sensor information has benefits over traditional approaches using Ethernet or other cable-based solutions. Wired connectors are not always available at all machine positions, and deploying new cables can be expensive and reduces flexibility.

Key challenges for efficient wireless transmission are the large quantity of available information, as well as the challenging environment. Many relevant parameters are acquired as time series data with high frequency. In our example use case, plastic injection molding, machines use a nozzle to inject heated plastic into molds while applying high pressure. The injected material then cools off, and finally, the mold is opened to eject the finished part. During the injection process, modern sensors sample temperature, pressure, nozzle position, and other parameters with up to $1000\,\mathrm{Hz}$ frequency, which may over-saturate the available wireless capacity when multiple machines with several sensors each operate in parallel, as is the case in most production set-ups. At the same time, factory floors often cover large areas and contain large amounts of metal, which may impede wireless transmissions. Therefore, for larger workshops, multi-hop communication may be required to support wireless coverage throughout the factory.

In this paper we contribute TANDEM, a transmission protocol that fulfills the needs of current industrial environments through a combination of three techniques: first we implement in-network prioritization for sensor data that provides a close approximation of current machine parameters in cases where the available wireless capacity is insufficient for complete transmission. Second, to improve data rates for challenging wireless links, our protocol builds multi-hop communication paths by first establishing reliable single-hop transmissions and then concatenating several such hops in a way that wireless capacity quickly convergences to max-min fairness while avoiding information loss even when nodes exhibit temporariy failure. Lastly, we reduce the number of wireless transmissions with a novel, lightweight broadcast acknowledgement mechanism. For information encoding, we build upon a robust single-hop protocol that uses the Discrete Cosine Transform (DCT) to quickly transmit approximate information [1]; TANDEM extends this information encoding for multi-hop scenarios, making it suitable for larger factories and when wireless interference necessitates multi-hop communication.

In the remainder of this paper, we first review related work on efficient wireless information dissemination in Section II. We describe our system model in Section III, and in Section IV, we explain our protocol in detail. We analytically discuss fairness and reliability of TANDEM in Section V and perform extensive network simulations and real-world measurements to evaluate protocol performance in Section VI. Section VII concludes the paper.

II. Related Work

TANDEM's use case closely relates to wireless mesh networks, for which a number of routing protocols have been proposed in the last decades [2]. Ad-hoc on-demand distance

© 2017 IEEE 10.1109/ICCCN.2017.8038458

vector (AODV) [3] and open link state routing (OLSR) [4] are two traditional approaches that aim to provide generic unicast routing mechanisms. AODV implements a *reactive* routing mechanism: routes are generated only when information is to be forwarded, reducing control message traffic but regularly introducing delays when sending packets. OLSR is an example of a link-state *proactive* routing mechanism, in which every node periodically maintains network topology information and routes for the whole network. In our use case, the number of nodes in the network is limited. Overhead due to proactive route maintenance, therefore, is not prohibitive. Different to forwarding routing decisions, we employ an acknowledgment mechanism that optimizes for robustness and prioritization, but we require detailed network state information including link quality, for which we utilize OLSR's topology information base [4]–[6]. We do not, however, use OLSR for actual information routing.

To determine best forwarding paths, we use the expected transmission count (ETX) metric [7], as simple hop counts would neglect differences in link quality, which may be particularly different in factory settings. ETX' core idea is to estimate the number of transmissions necessary in order to successfully transmit messages. Thereby, ETX strikes a balance between link reliability – which short links provide at the cost of a high hop count – and low delay – which can be achieved using (geographically) long hops at the cost of lossy communication. Draves *et al.* [8] compare different routing metrics for static and mobile network scenarios and argue that ETX performs best in static scenarios, such as our industrial use case.

Biswas *et al.* [9] propose an information dissemination protocol that is optimized for mesh networks and uses opportunistic overhearing to quickly cover large distances by adding meta-data called forwarding lists to each packet. In contrast to their proposal, we use overhearing during acknowledgment dissemination only, but we adhere to next-hop forwarding decisions for transmitting sensor information to reduce overhead and protocol complexity. Thereby, our protocol strikes a balance between the advantages of opportunistic overhearing, which can reduce the number of forwarding hops, and ad hoc routing, which removes the need for forwarding lists without unnecessary flooding of information and fully utilizes link-layer retransmissions without requiring hardware support.

In order to eliminate the need for complex routing path decisions and metrics, some works (e. g., [10], [11]) propose to apply network coding [12], [13] as an alternative that opportunistically disseminates coded packets instead of determining paths before transmission. While such network coding protocols eliminate path calculation complexity, they introduce additional network overhead, which may be prohibitive if network capacity is limited. Also, decoding a coded information collection requires receiving a sufficient number of independent linear combinations, disallowing prioritized and partial decoding. Approaches to allow prioritized decoding add significant computational complexity, which makes them unsuitable for most mesh networks [14].

Related to generic mesh networks, many routing algorithms have been proposed to disseminate information using wireless sensor networks [15]. Often, these approaches use routing metrics similar to those discussed above but combine them with different requirements. Sensor networks require support for a much larger number of nodes, more frequent topology changes, and are bound by severe limitations on nodes' memory, computational capacity, and permissible energy consumption [16]. In our industrial setting, the number of nodes is usually limited, power is readily available throughout a workshop, and more powerful hardware can be used, rendering the faster but more expensive ad hoc mechanisms preferable.

III. System Model

We assume that sensor information transmitted by our protocol is acquired in form of time series data associated with individual production cycles, as is the case in most industrial production processes. Our network consists of u machines $m_1, \ldots, m_u$. Each machine is equipped with a number of sensors. The number of sensors can differ per machine, as can the duration of cycles and the sensors' sample rates. Therefore, we identify units of information to be transmitted as a series of sensor values, which is identified by a machine-sensor-cycle 3-tuple (i, j, k), with production cycle number k, machine number i, and sensor number j.

Typical sensor frequencies range between $1\,\mathrm{Hz}$ and $1000\,\mathrm{Hz}$, and typical cycle durations are in the order of $1\,\mathrm{s}$ to $60\,\mathrm{s}$, which gives between 1 and 30.000 samples per production cycle tuple. We identify those samples as:

$$C_{(i,j,k)} = \Big(x_1, x_2, \ldots, x_{|C_{(i,j,k)}|}\Big). \tag{1}$$

Sensor information is recorded for a cycle duration; between such production cycles, machines have a cool down period, during which no sensor information is recorded.

We assume that, on average, link rates suffice to transmit all machines' cycle information at the rate with which they are generated. Still, bottlenecks may occur temporarily when connectivity is challenged, e. g., due to wireless interference.

Each machine i is equipped with one wireless node n_i; we call those nodes *source nodes*. One additional wireless node, termed *sink node* n_s, which is connected to a centralized processing server, is installed in the network. In addition, we support an arbitrary number of *forwarding nodes* that are not attached to a machine. These nodes extend wireless coverage for larger factories. Wireless nodes are assumed to generally operate non-stop, but they may fail temporarily, for instance, due to power failures, maintenance system reboots, or factory personnel relocating nodes.

The objective is to continuously transmit all available information from the machines $m_1, \ldots, m_n$ towards sink n_s. In addition to raw sensor samples, meta-data is needed to allow correct identification and processing of acquired sensor information. This meta-data $M_{(i,j,k)}$ always includes cycle-identifying information i, j, k. Additional meta-data may include cycle timestamps, sensor sample rates, and mold description fields, for instance. As the size of the meta-data

can be considered small relative to the time series data size, we abstract from the application details and model meta-data as arbitrary information $M_{(i,j,k)}$ in the remainder of the paper. In the remainder of this paper, unless otherwise noted and w. l. o. g., we describe the protocol considering a single production cycle (i, j, k) and thus drop the index to improve readability.

IV. The TANDEM Protocol

The TANDEM protocol operates on production cycles as defined in Section III. Here, we present an overview of the protocol's operation before delving into its details in the following subsections.

After a production cycle is completed, the machine transfers the acquired sensor information C to its attached wireless source node n_i, where it is preprocessed and inserted into the transmission queue. Preprocessing will be discussed in more detail in Section IV-A. Both source nodes and forwarding nodes schedule their information transmissions using a sending queue with an associated prioritization function, which we detail in Section IV-B. Nodes determine the shortest path to reach the sink using link-state routing information (see Section IV-C).

To improve robustness, acknowledgments are not only sent by the sink, but by each next hop on the path to the sink. To reduce overhead, these acknowledgements are send periodically, acknowledging whole cycles instead of individual packets. Acknowledgments, further discussed in Section IV-D, also leverage the wireless medium's inherent broadcast nature by implementing an overhearing mechanism. Responsibility for packet delivery is repeatedly passed on to nodes closer to the sink until eventually the sink receives and acknowledges each packet.

A. Production cycle preprocessing

Prior to wireless transmission, each source node performs two major tasks: (1) frequency-domain transformation and (2) splitting the resulting coefficients into blocks suitable for wireless transmission. Frequency domain transformation uses the DCT to utilize its energy compaction property for later in-network prioritization: broadly speaking, *most* of the contained sensor information can often be approximated with *few* of the DCT's low-frequency coefficients; higher frequencies then help to provide more details; and in their totality coefficients represent the complete original information. In injection-molding, this approximate sensor information is highly precise for both flawless parts produced and for parts with defects, such as blistering, non-fills, voids, and so forth [1].

Applying the DCT, a cycle's time series samples $C = x_1, x_2, x_3, \ldots, x_{|C|}$ are transformed to a series of cosine coefficients:

$$\hat{C} = \mathrm{DCT}(C) = \big(X_1, X_2, X_3, \ldots, X_{|C|}\big). \tag{2}$$

The coefficient cycle $\hat{C}$ has the same size as C and is usually too large for efficient wireless transmission in a single packet [17]. We therefore partition $\hat{C}$ into blocks $B = \big(b_1, b_2, \ldots, b_{|B|}\big)$ that satisfy a maximum packet size constraint P. To save network capacity, our block splitting algorithm only includes relevant meta-data M with the first block of each cycle, which we term *head block*. The remaining blocks save space by only holding a reference to the head block and, therefore, can accommodate more coefficients.

Formally, head block b_1 for cycle C consists of

$$b_1 = \big(M, \quad (X_1, X_2, \ldots, X_{\phi_1})\big), \tag{3}$$

where ϕ_1 (with $1 \leq \phi_1 \leq |\hat{C}|$) is an index chosen such that b_1 contains the maximum possible amount of coefficients while still satisfying the packet size constraint P. Consecutive blocks are built analogously, but with a reference – in the form of a cryptographic hash $H(\cdot)$ – to their head block:

$$b_2 = \big(H(b_1), (X_{\phi_1+1}, X_{\phi_1+2}, \ldots, X_{\phi_2})\big), \tag{4}$$

$$b_3 = \big(H(b_1), (X_{\phi_2+1}, X_{\phi_2+2}, \ldots, X_{\phi_3})\big), \tag{5}$$

$$\vdots$$

$$b_{|B|} = \big(H(b_1), (X_{\phi_{|\Phi|-1}+1}, X_{\phi_{|\Phi|-1}+2}, \ldots, X_{\phi_{|\Phi|}})\big). \tag{6}$$

Specifically, the index set $\Phi = \{\, \phi_1, \ldots, \phi_{|\Phi|} \,\}$ must satisfy

$$\forall\, \phi_i \in \Phi\colon \texttt{BinSize}(b_i) \leq P \quad \text{and} \quad \forall\, i < j\colon \phi_i < \phi_j, \tag{7}$$

where $\texttt{BinSize}(\cdot)$ is a coefficient block's binary, serialized size. In addition, we require that for all $i < |\Phi|$ that ϕ_i is the largest number that satisfies Equation (7).

After transforming cycle information with the DCT and splitting sensor readings into blocks, each block is inserted into the sending queue.

B. Information prioritization

As it is likely that the available sensor information per time unit may temporarily over-saturate available wireless capacity, an important component of our protocol is a suitable information prioritization metric. We use this metric to locally prioritize the sending queue of all wireless nodes, which may contain information from different machines, different sensors, and different production cycles due to the network's multi-hop nature. Rather than implementing a complex function that prioritizes using these individual parameters, we argue that a simple yet effective prioritization can be achieved by leveraging the DCT's energy compaction. Namely, we sort network packets in the sending queue such that low-frequency components are transmitted prior to high-frequency components, independent of their associated cycle id, machine id, and sensor id.

As a result, our wireless nodes first transmit information that enables the sink to approximate with similar precision all cycles that are currently transmitted. If the wireless capacity is not saturated, it is likely that the current production cycle's data is completely transmitted before the next cycle completes and is added to the transmission queue. If the channel is temporarily over-saturated, the next cycle's low-frequency coefficients will be added to the sending queue with higher priority than the current cycle's lower-frequency coefficients.

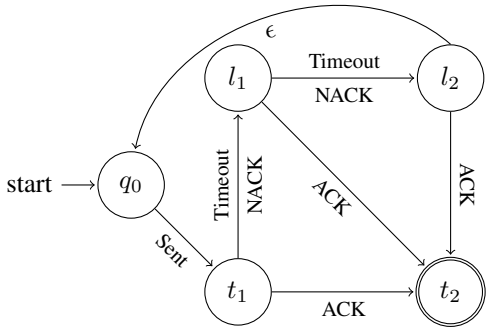


Fig. 1. State machine used for re-transmissions.

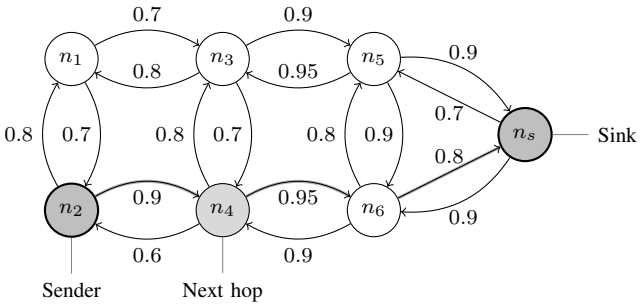


Fig. 2. Toplogy management and next-hop calculation.

Thereby, we ensure that all cycles can be approximated with low delay, and detailed representations are only transmitted when the current network capacity is sufficient.

The prioritization mechanism can be implemented efficiently using a priority queue of coefficient blocks, which can be implemented as a heap. Taking the highest priority coefficient block from the priority queue only requires constant time. Insertions take logarithmic time with respect to the size of the queue, which is sufficient for our use case where queues are expected to be temporary.

Besides prioritizing early approximations of sensor information, our transmission queue management also provides fairness for all machines. That is, it that guarantees that important information is delivered irrespective of a node's distance to the sink – an important property that we discuss as part of our protocol property analysis in Section V.

C. Topology management and transmission

Whenever the sending queue is non-empty, the most highly prioritized coefficient block is forwarded towards the sink. To determine the next forwarding hop, each wireless node maintains the current network topology in form of a a directed, weighted graph that represents link quality information. As we assume wireless nodes to remain at mostly static positions, we can obtain this topology information with tolerable communication overhead using existing link-state approaches.

The expected topology format is a graph that includes directed edge weights $w_i \in [0, 1]$ representing the expected packet delivery ratios. We obtain the ETX [7] for each directed edge – that is, the expected amount of total transmissions required to successfully transmit information along the edge – by taking the multiplicative inverse of the delivery ratio, $1/w_i$. Using the ETX-annotated graph as input, each wireless node then uses Dijkstra's weighted shortest path algorithm to determine the path towards the sink that has the lowest cumulative ETX. From the calculation results, the wireless node obtains its next hop towards the sink, which is determined by selecting the closest neighbor node on the calculated shortest path. In addition, the wireless node stores its own ETX-distance to the sink:

$$D_{n_i} = \sum_i \frac{1}{w_{p_i}}, \tag{8}$$

where p_i are the edges along the node's shortest path towards the sink. Figure 2 shows n_2 as an example sending node, which obtains the marked shortest path towards the sink, a distance $D_{n_2} \approx 3.41$, and n_4 as its next hop. Note that all shortest path calculations are only used locally to determine the next hop; subsequent nodes perform the same calculations to determine their next hop, and so forth. No routing information is transmitted along with the coefficient blocks, because topology information is more recent in a node's proximity and wireless nodes can accommodate changing interference in the environment during transmissions along a path.

Once the wireless node determined its next hop, it serializes the next coefficient block b_i from its sending queue to a binary representation and transmits the packet to its next hop. Block b_i is then temporarily removed from the sending queue until it is either acknowledged by a wireless node closer to the sink (in which case it is removed permanently) or until it is considered lost (in which case it is re-scheduled for transmission). Figure 1 shows the state machine that is used for re-transmissions, which incorporates timeouts for lost acknowledgment packets. Different from end-to-end mechanisms, such as TCP acknowledgments, we implement reliability in a transitive manner. That is, packets that are being forwarded along a shortest-path segment only cause retransmissions on this segment and not, as it is the case for TCP, along the whole path.

D. Periodic broadcast acknowledgments

Each wireless node periodically transmits information until it receives an acknowledgment from another wireless node that is closer towards the sink. We combine two optimizations to avoid unnecessary transmission overhead: first, each node accumulates acknowledgments for a whole cycle and sends acknowledgements periodically. Second, nodes farther than one hop away may overhear acknowledgments to reduce retransmissions.

An accumulated acknowledgment packet consists of the identifying machine-sensor-cycle tuple and a bit vector v. A bit 1 at position l in the bit vector indicates that coefficient block b_l of the cycle has been received. Conversely, $v_l = 0$ indicates that block b_l has not yet been received or was lost during transmission.

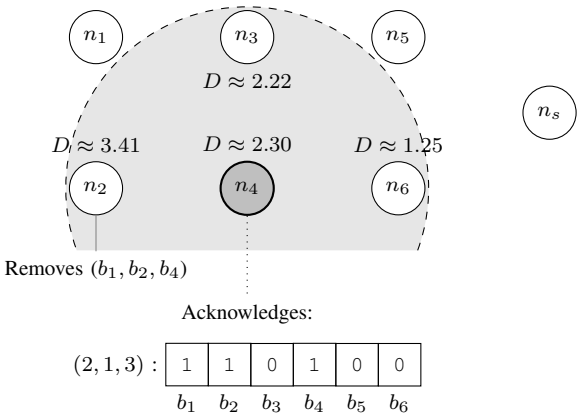


Fig. 3. Opportunistic acknowledgments.

Acknowledgment periods are fixed time intervals. We say a machine-sensor-cycle is *fresh* if a node has received at least one packet with some of its coefficients during the current acknowledgment period. In every period, each node broadcasts an acknowledgment packet for every fresh cycle. The accumulated transmission of bit vectors saves bandwidth, because fewer, accumulated packets reduce the transmission overhead due to packet headers. Since acknowledgments are implemented as link-layer broadcast, they can be overhead by all nodes within communication range, possibly skipping hops along the shortest path.

When a nodes receive a broadcast acknowledgment, they determine its meaningfulness based on their own sending state machine, which is shown in Figure 1, and their local topology knowledge: positive acknowledgements are processed by all nodes farther away from the sink, irrespective of their next-hop relationships. That is, when a node receives a positive acknowledgement in form if a 1-bit, it compares its own cumulative ETX towards the sink to that of the acknowledging node. If its own distance is perceived as larger than that of the node that sent the acknowledgement, the respective coefficient block is considered to be successfully transmitted and is removed from the transmission queue. In the example shown in Figure 3, n_2 removes b_1, b_2, and b_4 from its sending queue, because $D_{n_4} \approx 2.3 < D_{n_2} \approx 3.41$. The other receivers, n_3 and n_6, discard the acknowledgement, because $D_{n_4} > D_{n_3}$ and $D_{n_4} > D_{n_6}$, respectively.

In contrast to positive acknowledgments, negative acknowledgement bits only impact the sending state machine if the acknowledging node had been selected as next hop for the particular block b_i. This restriction avoids superfluous retransmissions, which otherwise occur whenever negative acknowledgements are overheard from nodes that are two or more hops downstream towards the sink and have not yet received the block in question. In Figure 3, n_3 and n_6 ignore all negative acknowledgements in the bit vector, because they never selected n_4 as next forwarding hop, whereas n_2 counts b_3, b_5, b_6 as negatively acknowledged.

V. Protocol Properties

This section discusses two protocol properties: reliability and fairness. With reliability we refer to guarantees that acknowledged information is delivered to the sink eventually. The fairness aspect is important to our use-case as we want to make sure that all machines' information is prioritized equally, i. e., machines farther from sink are not at a disadvantage. Reliability is best evaluated analytically, since we are interested in worst case behavior; we assess fairness both analytically in this section and via network simulations in the next section.

A. Reliability

In Section IV-D, we argued that our hop-by-hop acknowledgment mechanism adds reliability to the protocol, i. e., all confirmed information is delivered to the sink eventually. As each node indefinitely re-transmits blocks to its respective next hop until an acknowledgment is received, with the final next-hop being the sink, the sink receives all information eventually when all nodes continuously execute TANDEM. In practice, however, node failure renders it impossible to guarantee reliability to the same degree as end-to-end protocols, i. e., that all acknowledged information has been delivered, with hop-by-hop protocol designs [18]. In protocols that acknowledge per-hop, already confirmed packets may be lost when (a) buffer overflows occur or (b) nodes exhibit failure.

In our system model, we assume that bottlenecks are temporary; here, we model model the duration of a node being a bottleneck with random variable $X : \mathbb{R} \to \mathbb{R}$. Since bottlenecks are temporary, $\lim_{k->\infty} \mathbb{P}(X > k) = 0$, and therefore,

$$\forall\, p \in (0, 1]\, \exists\, k \in \mathbb{R} \colon \mathbb{P}(X > k) < p. \tag{9}$$

In other words, for any failure probability p, however small, we can find an upper bound k on the bottleneck duration. Even when assuming worst case behavior, in which case a bottleneck has a sending rate of zero, buffer space required to compensate for bottlenecks scales linearly with their duration. Therefore, for each node, an upper bound $L \in \mathcal{O}(k)$ on buffer space requirements can be found with negligible error p. As a result, when network size, average link rates, and each machine's sensor information rates are known, wireless nodes' storage size can be fitted large enough to guarantee sufficient buffering space for the use case, which guarantees eventual information delivery if nodes are functional at all times.

Also, we assume in our system model that node failure is temporary. While new information is routed around failed nodes via a different path, if available, some information may become unavailable upon node failure: when a node that uniquely stores acknowledged yet not forwarded blocks in its sending queue exhibits temporary failure, its blocks would be lost permanently. In order to guarantee eventual information delivery, TANDEM nodes are therefore required to persistently store blocks in their sending queue and re-send them after recovering from temporary failure.[1]

[1]Our protocol implementation also utilizes existing, volatile main memory without impeding reliability, which we describe in Section VI-E.

As an example, when nodes have 1 GB of persistent storage available for coefficient block storage, assuming a typical 500 Hz sample rate, 30 s cycle duration, four sensors per machine, and up to 15 machines in the workshop, TANDEM nodes can compensate bottlenecks and temporary node failures for at least one hour.

B. Fairness

Machines' sensor information is equally important regardless of whether a machine is located next to the sink or at the other end of the production floor. Most end-to-end window-based protocols, such as TCP, however, result in fairness behavior that gives more distant nodes a smaller proportion of bottleneck link capacity [18]. What we strive for with TANDEM is to guarantee an equal share of bandwidth to all machines. In network literature, the term *max-min fairness* [19, pp. 524] has been used to to describe packet schedulers that only increase a packet flow's bandwith when no packet flow with lower bandwidth can be increased instead. Hahne [20] showed that by employing a local round-robin scheduler at each forwarding node, the network eventually achieves global max-min fairness for every flow in the network.

Let $R(m_i)$ be the data rate from machine m_i's associated source node to sink n_s. Then, a feasible rate allocation vector $r = \big(R(m_1), R(m_2), ...\big)$, i. e., a rate allocation vector for which no node exceeds its maximum sending rate, is defined as max-min fair if for each machine m_i, $R(m_i)$ cannot be increased while maintaining feasibility without decreasing $R(m_j)$ for some $j \neq i$, $R(m_j) < R(m_i)$.

Without any bottleneck nodes, i. e., no node's packet receive rate being greater than its maximum sending rate, TANDEM equals a round-robin scheduler and therefore is max-min fair. Moreover, without bottlenecks, all rates in the rate allocation vector r must be equal.

Since our system model allows for temporary bottlenecks, next we consider the *aftermath* of such a congestion event, a non-equal rate allocation vector $\hat{r}$, that is,

$$\exists\, i, j \colon R(m_i) < R(m_j). \tag{10}$$

We argue that after congestion events, our prioritization results in faster convergence to a max-min fair rate allocation than round-robin: forwarding nodes prioritize packets based on coefficient frequencies. We assume w. l. o. g. that forwarding node n_k lies on the shortest path of both flow $m_i \rightarrow n_s$ and flow $m_j \rightarrow n_s$, in other words, n_k schedules packets that pertain both to rate $R(m_i)$ and rate $R(m_j)$. After the congestion event, $R(m_i) < R(m_j)$ holds, therefore forwarder n_k will on average have more packets with low frequency components of m_i in its sending queue, which leads to a prioritized rate $R(m_i)$ for low-frequency components, fastening convergence to max-min fair rates in the network. The same is less true for higher frequency components, since low-frequency components pertaining to $R(m_j)$ will be prioritized over high frequency components of $R(m_i)$, so restoration of max-min fair data rates takes proportionally longer for higher frequency coefficients.

VI. Evaluation

To validate TANDEM's performance, we performed extensive discrete event network simulations. In addition and as a proof of concept, we implemented TANDEM on industrial grade wireless hardware and measured performance. Unless otherwise noted, data points in plots of this section show the arithmetic means and 95% confidence intervals. Error bars might not always be visible in the figures when the error is very small.

A. Simulation methodology

We base our simulations on the discrete event network simulator ns-3 [21] and model wireless transmission via YANS Wifi model [22] with IEEE 802.11g Media Access Control (MAC) and 2.4 GHz physical layer. We consider obstructed line of sight and account for the effects of multipath propagation and large-scale path loss by employing Rayleigh and log-distance propagation loss models, one superimposed on another, as Hashemi [23] suggests. To obtain a meaningful sample size, we repeat each simulation using 40 independent sub-streams of NS-3's MRG32k3a pseudo-random number generator [24]. In addition to the packet loss probability, we add randomness to simulations by varying machines' cycle start times and by adding a random delay before sending each packet.

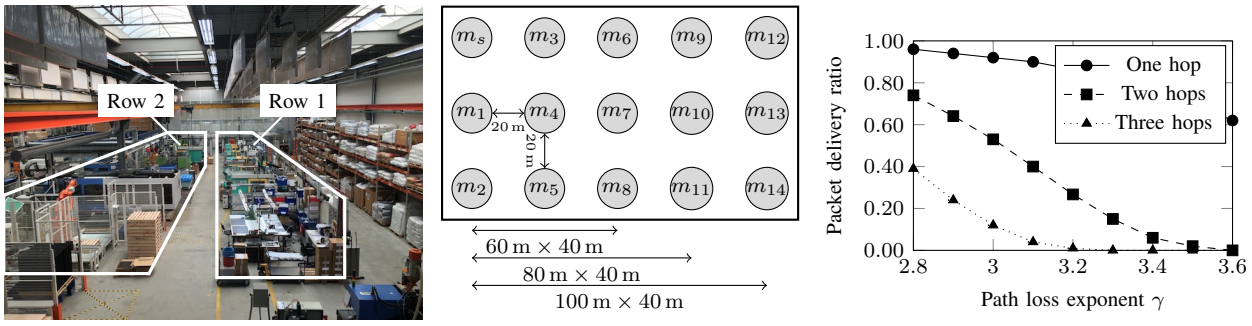
For network topology information, each node runs an OLSR implementation. To obtain ETX values, we implemented the OLSR link-quality extensions, introduced by "OLSRd" [5] and defined in RFC 7181 [6], for ns-3, which currently only implements the original OLSR standard [4].

As time series data, we replay pre-recorded sensor readings from real injection-molding machines and use packet sizes derived for a maximum packet size $P = 1024\,\mathrm{B}$. Each machine has four sensors, two cavity temperature sensors and two pressure sensors. Each sensor records samples at 500 Hz rate over 25 s production cycle duration. After each production cycle, machines have a cool-down period twice as long as the cycle.

The simulated factory topology is based on typical industrial factory layouts. Figure 4a shows an example workshop that is approximately 50 m long and consists of two rows of injection-molding machines. To show scalability, we simulate a similar but larger three-row topology, as shown in Figure 4b and vary the factory length between 40 m to 100 m. Node distances are fixed to 20 m while network conditions are varied via the log-distance path loss exponent γ, using values sensible for factory environments [25]; Figure 4c maps γ-values to packet delivery ratio for this topology configuration.

B. Prioritization effects

First, we evaluated how long it takes to receive a production cycle's prioritized low-frequency components. To this end, we simulated 15 machines in the $100\,\mathrm{m} \times 40\,\mathrm{m}$ factory. Each machine transmits 8 production cycles over 600 s simulated time. Figure 5 shows the cumulative distribution function of the relative time until the most prioritized 10%, 20%, 50%, and 100% of frequency coefficients were received. It can be seen



(a) A two-machine-row injection-molding factory.

(b) Simulation topology.

(c) Expected delivery probabilities.

Fig. 4. Simulation topology.

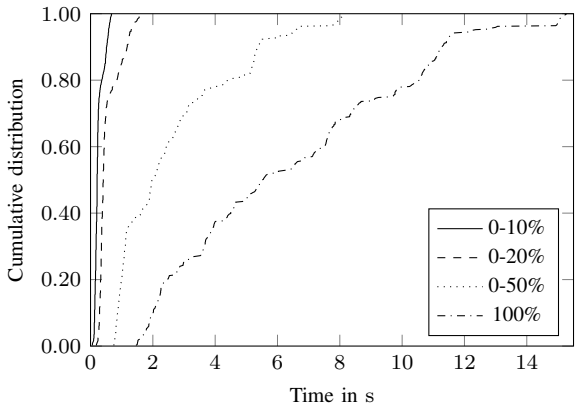
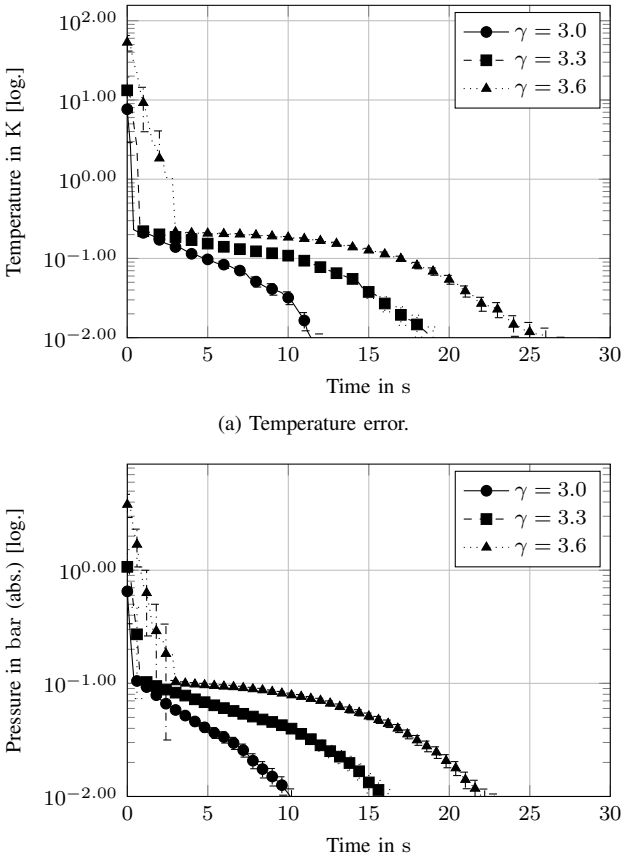


Fig. 5. Prioritization effects on 40 m × 100 m factory floor. ($\gamma = 3.0$)



(a) Temperature error.

(b) Pressure error.

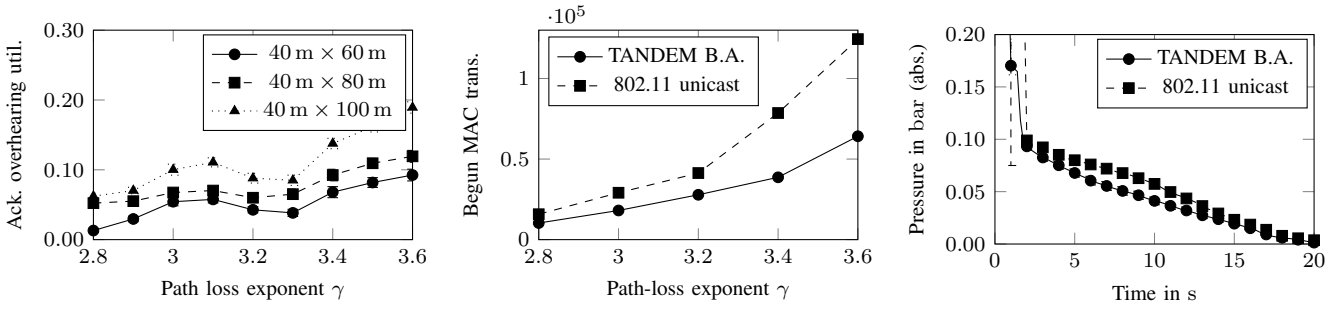
Fig. 6. Average error over time for 100 m × 40 m workshop.

that in-network prioritization has a drastic, non-linear effect: the most prioritized 10% and 20% of the lowest frequencies are available after a median time of only 210 ms and 410 ms, respectively, whereas median transmission duration of all the frequency coefficients is 5.52 s, which is one order of magnitude slower. The steepness of the low frequency components (10% and 20%) implies a high degree of fairness, which supports our fairness results in Section V-B.

C. Prioritization and Error

Next, we examined the effect of an incomplete set of coefficients on the sink's estimation quality. Pressure is given in bars (absolute) and temperature in degrees Kelvin. For each point in time we plot the average error based on the sink's information reconstruction. Here, we do not require that consecutive frequency coefficients were received, but instead also allow for gaps in the spectrum: analogous to missing high-frequency components, missing low-frequency coefficients are set to zero in order to reconstruct an approximation of the signal [1]. Figures 6a and 6b show how the average error in the signal, that is, the average over all reconstructed samples' absolute error, decreases over time for pressure and temperature, respectively.

In our pre-recorded sensor data, pressure samples assume values between between 1 bar and 331 bar. For γ-values 3.0, 3.3, and 3.6, it took on average less than 0.1 s, 0.2 s, and 1 s until the error of the preview drops below 1 bar, which is less than 0.5% of the value range. Similarly, temperature values range from 306 K to 355 K and the mean error drops below one Kelvin after 0.4 s, 0.8 s, and 3 s for γ of 3.0, 3.3, and 3.6, respectively. The error drops slower for cavity temperature because samples are more distorted by noise, which results in less efficient DCT compression, necessitating more coefficients for precise decoding.



(a) Acknowledgment overhearing utilization. (b) MAC-layer transmissions of acknowledgments. (c) Precision comparison to unicast ($\gamma = 3.4$).

Fig. 7. Broadcast acknowledgment mechanism.

D. Broadcast acknowledgment impact

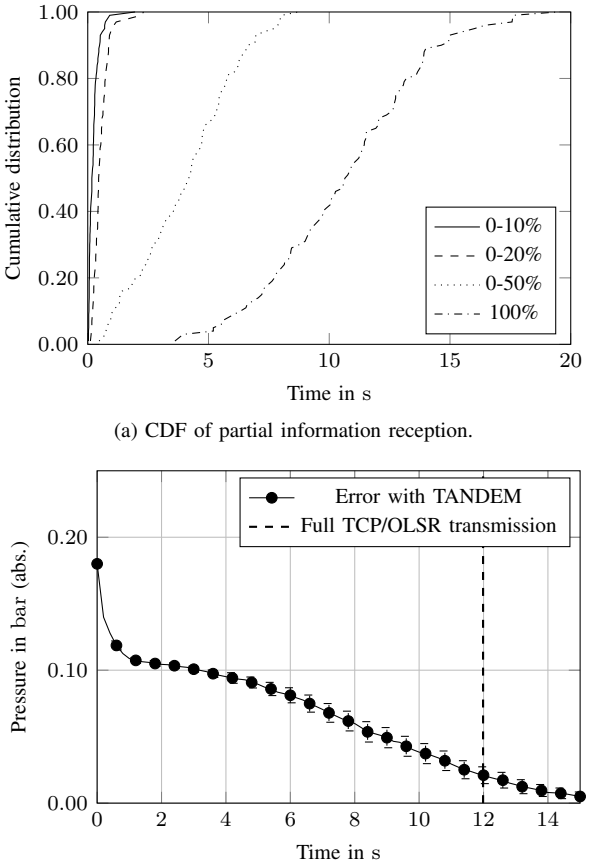
Finally, we examined the extent to which TANDEM's broadcast acknowledgment algorithm helps in confirming packets based on overhearing. To this end, we let each node track whether blocks sent were confirmed via overhearing or by its designated next hop. Let h be the number of blocks that were confirmed by each node's next hop and o be the number of blocks that were confirmed by other nodes through overhearing, then we define the metric acknowledgment overhearing utilization as $o/(o+h)$. Figure 7a shows this average ratio for all nodes in the network as a function of path loss exponent γ. The acknowledgment overhearing utilization is shown for all three factory size configurations; it can be seen that the protocol benefits most from acknowledgment overhearing when packet loss rates are high. Since only paths that involve three or more nodes can add to overhearing utilization, larger topologies ($80\,\mathrm{m} \times 40\,\mathrm{m}$ and $100\,\mathrm{m} \times 40\,\mathrm{m}$) benefit more from the mechanism.

In addition, we evaluated how many transmissions were saved by the overhearing mechanism in comparison to standard 802.11 unicast, which involves MAC layer retransmissions. Figure 7b shows that TANDEM's overhearing mechanism consistently saves between 34% and 48% of MAC-layer transmissions for good ($\gamma = 2.8$) and bad connectivity ($\gamma = 3.6$).

Figure 7c shows that using TANDEM's acknowledgment mechanism, precision improves slightly faster at the sink for $\gamma = 3.4$. For different γ-values, both acknowledgment approaches provide very similar performance. This means that the saving in transmissions through TANDEM's acknowledgment mechanism, which means less energy used and less occupation of the wireless medium, usually comes without a cost or even offers benefits in terms of preview precision.

E. Real-world measurements

As a proof of concept and to validate that our simulation results correspond to real world scenarios, we implemented TANDEM on industrial grade IP68-certified hardware and evaluated the protocol in a four-node diamond shape topology with three source nodes and one sink node. Table I summarizes the main hardware components used.



(a) CDF of partial information reception.

(b) Mean error in pressure over time.

Fig. 8. Real-world measurements.

To support reliability, our implementation employs a *persistent* database to store coefficient blocks that are recorded from a machine or received over the wireless link. Only an indexing key to the persistent database is written to the memory-backed prioritization queue that we described in Section IV-B. As a result, temporary node failure can be resolved on the next start up of the wireless node by iterating over all entries in the persistent storage and restoring the prioritization queue. The other two data structures, namely sender state and confirmation set, cannot be restored – which causes temporary redundant

TABLE I
HARDWARE COMPONENTS

Component	Description
CPU	500 MHz AMD Geode LX800
RAM	256 MB DDR RAM
Storage	8 GB CompactFlash
Expansion	2 miniPCI slots
Ethernet	Via VT6105M
Antenna Ports	U.FL (Mini-SMT/I-PEX)
Wireless	Atheros XSPAN-Family AR9220

transmissions in case of node failures, but nonetheless ensures reliability. Otherwise, the implementation follows Section IV.

Prioritization results are shown in Figure 8a and confirm our simulation results: the prioritized information is transmitted quickly in all cases, whereas high frequency components take longer and are distributed less evenly. We further obtained a total of 20 comparison measurements using reliable TCP transmissions with equivalent data sizes, that is, including the relevant production cycle's meta-data. Shortest paths were provided by OLSR with link quality extensions enabled. TCP transmission took an average of 11.99 s, which is shown by the vertical line in fig. 8b. While TCP took less time for delivering all information, our in-network prioritization gives a close approximation much faster. As soon as a preview was available, the mean error we observed was lower than 180 mbar, which are less than 0.05% of the value range. Similarly, as soon as an approximate cavity temperature is available, its the mean error is less than 0.06% of the range.

In the real-world measurements, we observed an average acknowledgment overhearing utilization of 9.91% on the distant source node, which is the only node that can overhear acknowledgments due to the simple topology. Again, this result is in line with simulation results in Section VI-D.

VII. CONCLUSION

To facilitate reliable wireless communication in industrial environments, we contribute a wireless multi-hop transmission protocol, TANDEM, that is tailored towards the use case of delivering production processes' sensor information in a timely manner. TANDEM utilizes the DCT's high energy compaction to prioritize important information in the network and uses a novel hop-by-hop reliability mechanism that leverages the wireless medium's inherent broadcast nature.

Our simulative evaluation results, which are supported by a real-world implementation, shows that our protocols' in-network prioritization ensures timely delivery of prioritized frequency components despite challenging network conditions. Using real sensor readings from sensorized injection-molding machines for evaluation, our evaluation shows that the error of an approximation based on our in-network prioritization is low and that this information is available much faster than delivery of all information would allow. We also observed that our acknowledgment approach saves transmissions by overhearing normally discarded, distant acknowledgment information.

ACKNOWLEDGMENTS

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 636892. Website: www.PREVIEW-project.eu.

REFERENCES

[1] R. Naumann, S. Dietzel, and B. Scheuermann, "INFLATE: Incremental wireless transmission for sensor information in industrial environments," in *2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, Dec. 2015.

[2] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: A survey," Mar. 2005.

[3] C. Perkins, E. Belding-Royer, and S. Das, *Ad hoc On-Demand Distance Vector (AODV) Routing*, RFC 3561 (Experimental), Internet Engineering Task Force, Jul. 2003.

[4] T. Clausen and P. Jacquet, *Optimized Link State Routing Protocol (OLSR)*, RFC 3626 (Experimental), Internet Engineering Task Force, Oct. 2003.

[5] (2016). OLSR.org Wiki, [Online]. Available: http://www.olsr.org/mediawiki/index.php/Main_Page (visited on 06/26/2016).

[6] T. Clausen, C. Dearlove, P. Jacquet, *et al.*, *The Optimized Link State Routing Protocol Version 2*, RFC 7181 (Proposed Standard), Internet Engineering Task Force, Apr. 2014.

[7] D. S. J. De Couto, D. Aguayo, J. Bicket, *et al.*, "A High-throughput Path Metric for Multi-hop Wireless Routing," Jul. 2005.

[8] R. Draves, J. Padhye, and B. Zill, "Comparison of Routing Metrics for Static Multi-hop Wireless Networks," in *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '04, ACM, 2004.

[9] S. Biswas and R. Morris, "ExOR: Opportunistic multi-hop routing for wireless networks," in *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM 2005, ACM, 2005.

[10] S. Chachulski, M. Jennings, S. Katti, *et al.*, "MORE: A network coding approach to opportunistic routing," 2006.

[11] D. Koutsonikolas, Y. C. Hu, and C.-C. Wang, "Pacifier: High-Throughput, Reliable Multicast without"Crying Babies"in Wireless Mesh Networks," in *INFOCOM 2009, IEEE*, IEEE, 2009.

[12] R. Ahlswede, N. Cai, S. Y. R. Li, *et al.*, "Network information flow," Jul. 2000.

[13] T. Ho, M. Médard, R. Koetter, *et al.*, "A random linear network coding approach to multicast," 2006.

[14] K. Nguyen, T. Nguyen, and S. c Cheung, "Peer-to-peer streaming with hierarchical network coding," in *2007 IEEE International Conference on Multimedia and Expo*, Jul. 2007.

[15] J. N. Al-Karaki and A. E. Kamal, "Routing Techniques in Wireless Sensor Networks: A Survey," Dec. 2004.

[16] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, *et al.*, "Wireless sensor networks: A survey," 2002.

[17] J. Korhonen and Y. Wang, "Effect of packet size on loss rate and delay in wireless links," in *Wireless Communications and Networking Conference, 2005 IEEE*, IEEE, 2005.

[18] S. Kopparty, S. V. Krishnamurthy, M. Faloutsos, *et al.*, "Split TCP for mobile ad hoc networks," in *IEEE Global Telecommunications Conference, 2002. GLOBECOM '02*, Nov. 2002.

[19] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data Networks*. Prentice-Hall International New Jersey, 1992.

[20] E. L. Hahne, "Round-robin scheduling for max-min fairness in data networks," Sep. 1991.

[21] T. R. Henderson, M. Lacage, G. F. Riley, *et al.*, "Network simulations with the ns-3 simulator," 2008.

[22] M. Lacage and T. R. Henderson, "Yet another network simulator," in *Proceeding from the 2006 Workshop on Ns-2: The IP Network Simulator*, ACM, 2006.

[23] H. Hashemi, "The indoor radio propagation channel," 1993.

[24] P. L'Ecuyer, R. Simard, E. J. Chen, *et al.*, "An Object-Oriented Random-Number Package with Many Long Streams and Substreams," Dec. 2002.

[25] S. Phaiboon, "Space Diversity Path Loss in a Modern Factory at frequency of 2.4 GHz," 2014.