Reliable Communication and Latency Bound Generation in Wireless Cyber-Physical Systems

SIRAJUM MUNIR, Bosch Research and Technology Center
HAO-TSUNG YANG and SHAN LIN, Stony Brook University
S. M. SHAHRIAR NIRJON, University of North Carolina at Chapel Hill
CHEN LIN, Sun Yat-sen University
ENAMUL HOQUE, JOHN A. STANKOVIC, and KAMIN WHITEHOUSE,
University of Virginia

Low-power wireless communication has been widely used in cyber-physical systems that require time-critical data delivery. Achieving this goal is challenging because of link burstiness and interference. Based on significant empirical evidence of 21 days and over 3.6 M packet transmissions per link, we propose both routing and scheduling algorithms that produce latency bounds of the real-time periodic streams and accounts for both link bursts and interference. The solution is achieved through the definition of a new metric $B_{\max}$ that characterizes links by their maximum burst length, and by choosing a novel least-burst-route that minimizes the sum of worst-case burst lengths over all links in the route. With extensive data-driven analysis, we show that our algorithms outperform existing solutions by achieving accurate latency bound with much less energy consumption. In addition, a testbed evaluation consisting of 48 nodes spread across a floor of a building shows that we obtain 100% reliable packet delivery within derived latency bounds. We also demonstrate how performance deteriorates and discuss its implications for wireless networks with insufficient high-quality links.

CCS Concepts: • **Networks** → **Network protocols**; **Network reliability**;

Additional Key Words and Phrases: Link burstiness, link interference, latency bound, reliable transmission, real-time applications

ACM Reference format:
Sirajum Munir, Hao-Tsung Yang, Shan Lin, S. M. Shahriar Nirjon, Chen Lin, Enamul Hoque, John A. Stankovic, and Kamin Whitehouse. 2019. Reliable Communication and Latency Bound Generation in Wireless Cyber-Physical Systems. *ACM Trans. Cyber-Phys. Syst.* 4, 2, Article 15 (November 2019), 26 pages.
https://doi.org/10.1145/3354917

This work was supported in part by U.S. National Science Foundation under grants CNS-1553273 (CAREER), CNS-1816213 and CNS-1704469.

Authors' addresses: S. Munir, Bosch Research and Technology Center; email: Sirajum.Munir@us.bosch.com; H.-T. Yang and S. Lin, Stony Brook University; emails: haotyang@cs.stonybrook.edu, shan.x.lin@stonybrook.edu; S. M. S. Nirjon, University of North Carolina at Chapel Hill; email: nirjon@cs.unc.edu; C. Lin, Sun Yat-sen University; email: Lin.Chen@lri.fr; E. Hoque, J. A. Stankovic, and K. Whitehouse, University of Virginia; emails: {eh6p, jas9f, whitehouse}@virginia.edu.

2378-962X/2019/11-ART15 $15.00
https://doi.org/10.1145/3354917

1 INTRODUCTION

More and more cyber-physical systems (CPS) have been applied to industrial processes [19, 28], structural health monitoring [4, 12], and smart cities [10, 23, 26]. In those applications, low-power wireless communication technology gains rapid adoption due to its better scalability, lower maintenance costs, and flexibility in installation points for either sensing or actuation [21, 39, 42]. To support CPS for controlling or monitoring physical processes, wireless communication in CPS usually requires high reliability and hard end-to-end deadline. However, achieving these goals is difficult, since wireless links are non-deterministic because of link burstiness and interference [11, 29, 37]. Link burstiness is a physical property that means transmissions on a wireless link do not have an independent probability of failure; instead, they have periods of continuous message loss, i.e., they fail in a burst. Link interference is another physical property of the communication environment that causes packet transmission between different links to interfere with each other, which results in packet loss. Due to these types of non-determinism, it is difficult to offer reliability and latency bounds for packet delivery over wireless networks.

In this article, we propose a systematic approach for CPS communication to achieve reliable packet transmission with bounded latency. Our solution includes the following steps: First, we characterize physical properties like interferences and burstiness of a particular network. Then, we compute the latency bound for reliable delivery of a certain number of packets on each link. Finally, we schedule packet transmissions of multiple streams in the network to achieve reliable end-to-end reliability within specified latency bound.

It is obvious that we cannot allocate only a single transmission time slot for a stream on each link, especially if we are dealing with bursty links. Because, if the transmission fails at that time slot due to a link burst, the node will need some additional time slots to transmit its packet. So, to provide the end-to-end latency bounds, we have to allocate more than one time slot per link for a stream. The number of time slots we need to allocate depends on the burstiness of the link. We do not want to allocate more time slots than we need, otherwise, we will increase the latency bound. In addition, because of interference, other streams cannot be scheduled in nearby links during that entire multiple slot allocation time, which may increase the overall latency bound of all streams. So, achieving reliable communication and minimizing latency bound by schedule is, therefore, a challenging goal.

The specific problem we are addressing assumes that we are given a network topology and a set of periodic streams. The route of each stream is either given or assigned by our routing algorithm. After the routing phase, our algorithm outputs a packet transmission schedule and estimates a latency bound for each stream by taking into account both link burstiness and interference. For each stream, if the estimated latency bound is lower than or equal to the period, we offer reliable end-to-end delivery. We confirmed that traditional approaches based on packet reception rate cannot bound latency, because they do not account for link bursts. Our 21-day-long empirical study shows the evidence that over 23% links having packet reception rate (PRR) as high as 0.99 lose more than 50 packets in a row—some lose even over 1 K packets in a row. Thus, we need to carefully design some other metrics to characterize the burstiness of the links that will allow us to allocate a sufficient number of time slots to produce a latency bound. The main contributions of this article are listed as the following:

- Based on 21 days of empirical study over an 802.15.4 network, we define a new metric *max burst length ($B_{\max}$)* that allows us to classify links and allocate a sufficient number of time slots to produce a latency bound of the streams. Empirical analysis indicates that within a period of measuring $B_{\max}$, the measured $B_{\max}$ values are consistent, robust, and can characterize link quality better than PRR.

- Our algorithm has two phases. In the routing phase, we use least-burst-routes to produce minimum latency bounds of the streams by taking into account link bursts and load-balancing. In the broadcast scenario, the algorithm generates a tree-structured route that takes the advantage of omnidirectional transmission. In the unicast scenario, the algorithm generates "least burst" route for each stream, which considers load-balancing of each link. In the scheduling phase, the scheduling algorithm is a greedy-based one with the consideration of internal interference.
- The simulation results suggest that the routes generated from our algorithms have lower latency bounds compared with other baselines. By using testbed evaluation, we also show that we can actually bound the latency by achieving 100% packet delivery ratio within the derived latency bounds. We also investigate how the performance degrades when we do not have sufficient high-quality links in the network.

An implication of these contributions is that if the transmission period of each stream is greater than or equal to its latency bound that we provide, then our scheduling algorithm allows reliable communication subject to our burstiness and interference assumptions. If the burstiness characterization used for creating the schedules is violated during the course of execution, then a packet might still be missed, but this is rare, because the link characterization is performed under realistic operating conditions and an adaptive solution will reduce such scenarios subsequently. Note that our approach does not minimize latency to maximize throughput. Instead, our average delivery latency is higher than most other techniques. However, we do offer a reliable communication and latency bound, which makes it easier to engineer predictable systems. We verify this claim by evaluating our approach on a 48-node wireless testbed with 10 simultaneous and periodic packet streams. The result shows that our scheme has a 100% on-time delivery ratio when all links are available and 90% ones when top 25% PRR links are removed.

A preliminary version of the results in this article [29] mainly focuses on the method of calculating end-to-end latency bounds with $B_{\max}$. In this article, we provided additional analysis on the robustness of $B_{\max}$ among links in different classes. Moreover, we proposed routing algorithms for unicast and broadcast scenarios that achieve minimum latency bounds. The evaluation of the routing algorithms and comparison with other baselines is provided in the simulation section.

The rest of this article is structured as follows: Section 2 formally defines the problem. Section 3 describes related work on scheduling streams with real-time constraints in low-power wireless communication. Section 4 describes our model parameters, assumptions, and link classification based on an empirical study. Section 5 describes the routing/scheduling algorithms to achieve minimum latency bounds for streams. Section 6 shows the simulation result of the algorithm performance and compares it with other baselines. Section 7 describes the experimental setup and the results of the experiments. We discuss the future work in Section 8 and conclude our work in Section 9.

2 PROBLEM DEFINITION

The problem that we are addressing is formally stated as follows: Given a number of periodic streams and a fixed network topology, output a transmission schedule with an estimated end-to-end latency bound of each stream. Applications have a constraint on period for each stream. If the provided latency bound is less than or equal to its period, then our approach offers reliable communication.

We define a Stream Set SS that contains n periodic streams, where a periodic stream S_i has a source node SRC_i, a destination node $DEST_i$, a starting time ST_i, a period P_i, and a route RT_i(optional), where $i = 1, 2, 3, \ldots, n$ and a stream is represented as a six-tuple (S_i, SRC_i, $DEST_i$,

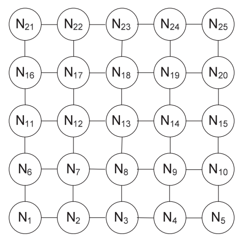


Fig. 1. An example topology.

Table 1. An Example Set of 4 Streams

Stream	Period	ST	Source	Dest.	Route
S_1	20	1	N_1	N_4	N_1, N_2, N_3, N_4
S_2	20	1	N_2	N_5	N_2, N_3, N_4, N_5
S_3	20	1	N_7	N_9	N_7, N_8, N_9
S_4	10	1	N_{17}	N_{19}	N_{17}, N_{18}, N_{19}

ST_i, P_i, RT_i). After the starting time is elapsed, at every period, the source node has a packet that has to be transmitted to the destination node by following that route. The route RT_i can be either fixed or assigned by our algorithms.

Network topology is specified as a set of nodes $N_1, N_2, N_3, \ldots, N_k$ along with their connectivity matrix and interference matrix. For each pair of connected nodes N_i and N_j, we denote the intermediate link as $L(i, j)$, which has burstiness parameters $B_{\max}$ and $B'_{\min}$, which are estimated based on empirical data. We will define these parameters formally in Section 4.

For example, consider the network topology shown in Figure 1. If two nodes are within their radio range, then they are connected by an edge. Suppose that we are given 4 streams to calculate their latency bounds. The details of the streams are shown in Table 1. The route means stream S_1 is going from node N_1 to N_4 using route $N_1 \to N_2 \to N_3 \to N_4$. So, we have

$$\begin{aligned} SS = \{&(S_1, N_1, N_4, 1, 20, RT_1),\ (S_2,\ N_2, N_5, 1, 20, RT_2),\\ &(S_3, N_7, N_9, 1, 20, RT_3),\ (S_4, N_{17}, N_{19}, 1, 10, , RT_4)\}, \end{aligned} \tag{1}$$

where $RT_1 = \{N_1, N_2, N_3, N_4\}$, $RT_2 = \{N_2, N_3, N_4, N_5\}$, $RT_3 = \{N_7, N_8, N_9\}$, $RT_4 = \{N_{17}, N_{18}, N_{19}\}$.

Given this input set, our algorithm schedules all the streams and outputs latency bound LB_i for each stream S_i. We claim that for any integer k, the kth packet of stream S_i will reach its destination no later than $ST_i + k \cdot LB_i$ if packet transmission is performed according to our schedule.

3 RELATED WORK

The challenge of providing real-time and reliable transmission in cyber-physical system has been addressed in Reference [18] and researched in recent years [8, 25, 31, 45, 46]. The general framework includes link quality estimation [5, 22, 36] and routing/ scheduling algorithm design for real-time streams [35, 38, 44]. Particularly, the quality of wireless links fluctuates due to link burstiness, which has been closely investigated by recent studies [1, 2, 32, 40]. The critical aspect in these works is to accurately capture the characteristic link burstiness and design algorithms to guarantee QoS.

3.1 Link Burstiness

The effort of dealing or coping with link burstiness can be divided into offline approaches, which focus on the long-term characteristic of link quality, and online approaches, which aim to detect short-term link quality fluctuation. For offline approaches, K. Srinivasan et al. [36] present the β metric to measure link burstiness. β is calculated by using a conditional probability packet delivery function (CPDF). By giving an n previous packets delivered trace, the function determines the probability of successfully delivering the next packet. His empirical study shows that link burstiness is avoidable by having an inter-packet delay of at most 500 ms. Yet, it does not specify the inter-packet delays for links with different characteristics of link burstiness. Moreover, as M.

Radi et al. [32] observe, measuring β requires a large amount of data to achieve a 95% confidence interval.

J. Wen et al. [40] propose an offline scheme that estimates the expected reliable transmission periods by using the conditional probability distribution function of the signal-to-noise ratio (SNR) of received acknowledgment packets. Their approach can detect link burstiness when the SNR of received packets is below a threshold. However, empirical studies show that the estimated burst periods might be over- or under-estimated, since the SNR varies for most intermediate links.

In advance, Z. Ansar et al. [1] propose a transmission scheme that applies a two-stage Markov model to characterize link quality fluctuation. They cluster link quality into k different levels based on the values of Acknowledgement Reception Ratio (ARR) and SNR from a large amount of packet transmission traces. After clustering, the transition probabilities between the levels is computable by using a first-order Markov chain. Moreover, the expected number of successfully received packets at each level can be acquired by using another first-order Markov chain. Their experiments show that their approach improved the packet delivery ratio of the links by up to 40%.

For the online approaches, Brown introduces BurstProbe [5], a mechanism to measure link burstiness ($B_{\max}$ and $B'_{\min}$) online. The mechanism has an additional probing slot in the transmission schedule such that each node can probe and estimate link burstiness online and share the information among neighbor nodes. This approach is more reactive for capturing short-term burstiness but requires additional time slots to measure link quality. Ansar [2] proposes another online transmission scheme extended from his previous offline approach [1]. His scheme is twofold. During the offline phase, the link quality has been categorized into three different stages (good, intermediate, bad) by applying a two-stage Markov model. In the online phase, the transmission scheme takes the short-term statistics of received acknowledgment packets to predict the most probable future state and the associated burst length. They implement the scheme on the TinyOS using the TelosB platform, and their experiments show that their scheme produces a higher throughput compared with the scheme β. However, it also shows that their scheme exhibits 4% to 10% higher packet losses than the β.

The methods of capturing the characteristics of link burstiness in these offline/online approaches are all based on the probabilistic functions. Different from their works, the $B_{\max}$ metric is focused on the characteristics of link burstiness in the worst-case scenario. It aims to provide end-to-end latency bound on hard real-time applications.

3.2 Routing/Scheduling Algorithm

However, there are a number of scheduling algorithms of stream transmission in CPS. Here, we only provide some representative works and one can refer to Reference [24] for a comprehensive survey. In early stage, SPEED [14] maintains a desired delivery speed across the network by a combination of feedback control and non-deterministic geographic forwarding. It is designed for soft real-time applications and it is not concerned with the reliability issues of individual streams.

For recent works, RDDS [15] provides a semantics-aware communication mechanism to guarantee QoS that mainly focus on the predictive sensor models. A multicast routing is provided in Reference [20] that can control the data flow in real-time CPS but it has no reliability guarantee. C.T. Sony et al. [35] provide a method to cluster hops and give a TDMA schedule with a reliable transmission for each stream. However, there is no reliability guarantee on these researches and it is only suitable in LEACH (Lower Energy Adaptive Clustering Hierarchy) protocol [27, 43]. Without the requirement of specific protocols, Reference [44] proposes a scheduling algorithm for reliable packet delivery with end-to-end delay constraint in the data-link layer. However, this work requires two steps: scheduling and extending, which makes it hardly scalable. In advance, Reference [8] provides SchedEx, a low-complexity generic extension for existing slot-based scheduling algorithms that improves the calculation time of re-scheduling and makes scheduling algorithms scalable.

H. T. Yang et al. [13] propose a cross-layer (MAC and network layer) scheduling algorithm to provide reliability guarantees for multi-stream. In their scenario, each stream has different reliability requirements, and they aim to schedule streams such that each stream can meet its requirement while minimizing end-to-end latency. They propose a routing algorithm that does not take the internal interference into account. Compared with their work, we present our complete algorithm for scheduling multi-stream that provides end-to-end latency bounds. We also implement the system with real-world experiments.

4 EMPIRICAL STUDY

To see how burstiness affects packet transmission, we conducted a 21-day experiment on a testbed with 48 Tmote nodes running TinyOS. For each pair of two nodes, a total of 3.6 M packets are transmitted to evaluate packet delivery traces. These nodes were deployed on the walls and ceilings of a building. The transmission rate was approximately 200 packets per second for all links. The transmissions of different links were scheduled at different time slots to avoid a collision. The receiving status of each packet at every node is recorded as a sequence of 0 and 1. These sequences are used for calculating $B_{\max}$, and further analysis is in the following subsections.

In Section 4.1, we show how to characterize link bursts with parameters $B_{\max}$ and $B'_{\min}$. In Section 4.2, we classify links based on the value of $B_{\max}$ and discuss the robustness of $B_{\max}$. We confirmed that the $B_{\max}$ value reflects the maximum number of retransmission times in a link with more than 99% correctness while the link is affected by the burstiness. This suggests $B_{\max}$ as a robust signature to characterize link bursts.

4.1 Model Parameters and Assumptions

To characterize link bursts, we define five parameters: B, $B_{\max}$, B', $B'_{\min}$, and W for every link. We define W as a window of outcomes of packet transmission. As an example, when $|W|$ is 3, we consider a sliding window of a size 3 over the outcomes of packet transmission. We define B as the number of time slots where packet transmission failed due to link bursts for a window W. We define B' as the number of time slots where packet transmission was successful for a window W. We define $B_{\max}$ as the maximum value of B for all possible windows of size $|W|$ and $B'_{\min}$ as the minimum value of B' for all possible windows of size $|W|$.

The way these parameters are computed is as follows: After transmitting 3.6 M packets over every link, we have a long sequence of data trace of 0s and 1s per link where 1 at ith index of the sequence means that packet with ith sequence number was successfully transmitted and 0 at that place means it failed. As an example, consider a sample data trace 0110010011 of length 10. We would like to estimate the $B_{\max}$ of this link. To do that, we start with choosing $B'_{\min}$. Let us assume, we choose $B'_{\min} = 1$. We start with a window slightly bigger than $B'_{\min}$, which is $|W| = 2$. Now, for every window of size 2, we check if there is at least one good slot within the window, since $B'_{\min} = 1$. Here, we have nine different windows of size 2. The first window spans from index 1 to 2, the second one spans from 2 to 3, and so on. In the first window containing 01, where we have $B = 1$, $B' = 1$. In the second window containing 11, we have $B = 0$, $B' = 2$. In the third window containing 10, we have $B = 1$, $B' = 1$. However, in the fourth window containing 00, we do not have a good slot as $B = 2$, $B' = 0$. So, we increase the window to 3. Now, we have eight different windows to consider, each of length 3. We have $B = 1$, $B' = 2$ for the first window (011), $B = 1$, $B' = 2$ for the second window(110), $B = 2$, $B' = 1$ for the third window(100), and so on. At the end, we get $B_{\max} = 2$ for this link for $B'_{\min} = 1$. Note that it means that there is at least one good slot for every window of size 3. In the rest of the article, we use the terms $W, B_{\max}, B'_{\min}$ when we discuss the general burstiness influence and link quality and $W(i,j), B_{\max}(i,j), B'_{\min}(i,j)$ when we want to emphasize its calculated values on link $L(i,j)$.

Our proposed solution uses the same $B'_{\min}$ for all the links. Choosing a higher $B'_{\min}$ of a link can reduce latency bounds of the streams that use this link due to spatio-temporal overlapping as mentioned in Section 5.3. However, choosing a higher $B'_{\min}$ of a link when such spatio-temporal overlapping does not exist can hurt latency bounds of the streams that use the link. We leave the selection of optimal $B'_{\min}$ of each link for minimizing latency bound of each stream as future work. More details on the selection of $B'_{\min}$ is described in Section 8.2.

ALGORITHM 1: ComputeBmax(D, $B'_{\min}$)

1: **for** $i \leftarrow B'_{\min} + 1$ to $|D|$ **do**
2:   $isSatisfied \leftarrow TRUE$
3:   **for** $j \leftarrow 1$ to $|D| - i$ **do**
4:     $B' \leftarrow \sum_{k=j}^{j+i-1} D[k]$
5:     **if** $B' < B'_{\min}$ **then**
6:       $isSatisfied \leftarrow FALSE$
7:       **break**
8:     **end if**
9:   **end for**
10:   **if** $isSatisfied = TRUE$ **then**
11:     $B_{\max} \leftarrow i - B'_{\min}$
12:     **return** $B_{\max}$
13:   **end if**
14: **end for**
15: **return** -1

Algorithm 1 computes $B_{\max}$ given a data trace D of 0s, 1s, and $B'_{\min}$. It returns -1 if the data trace does not have a $B_{\max}$ that satisfies the condition specified by the arbitrary value of $B'_{\min}$. The running time of the algorithm is $O(|D|^2)$, which is large for long data trace, although B' at line 4 can be computed in $O(1)$ time by using a dynamic programming-based memorization approach. But if a link has a very high $B_{\max}$, it indicates that the link is prone to heavy bursts, and we avoid this link for real-time applications. So, for practical purposes, we limit the maximum $B_{\max}$ to be $C = 1{,}200$ and constrain the loop at line 1 to run from 1 to C. Then, the running time of the algorithm becomes $O(C|D|)$, which is linear with respect to the size of the data trace.

Our model assumes that after network characterization, i.e., after we compute $B_{\max}(i,j)$ and $B'_{\min}(i,j)$ of every link $L(i,j)$, if we consider $B_{\max}(i,j) + B'_{\min}(i,j)$ time slots for packet transmission, we have at least $B'_{\min}(i,j)$ time slots to transmit packet successfully. Although the assumption looks questionable, we have some strong arguments in favor of it. The assumption may not hold in a battlefield where link behavior may change drastically, but it seems to hold in a regular working environment such as offices, universities, and industrial plants if we can characterize the links for a long period of time under all possible working environments. Obviously, this assumption will not hold for all the wireless links. We classify the links (in the next section) for which the assumption seems to be true. If the wireless links are not good enough to meet the assumption, then we can move some nodes or add additional nodes in the network to create the "right" topology having good burst properties that satisfy the model assumption. Also, another work [33] has explored that bursts in the wireless link have scaling properties, meaning that the bursts show self-similarity or

HFLB : High burstiness frequency and low $B_{\max}$
HFHB : High burstiness frequency and high $B_{\max}$
LFLB : Low burstiness frequency and low $B_{\max}$
LFHB : Low burstiness frequency and high $B_{\max}$

HFLB	HFHB	LFLB	LFHB
12.2%	37.8%	38%	12%

Fig. 2. Link classes distribution.

other coherent structure over many time scales without having long-range dependence. The article specifies an onset point of 640 ms where random variations stop affecting the wireless link and self-similarity starts to dominate. It clearly indicates the possibility of capturing the burstiness characteristics of the wireless links if we investigate the burstiness behavior of the wireless links over a long period of time.

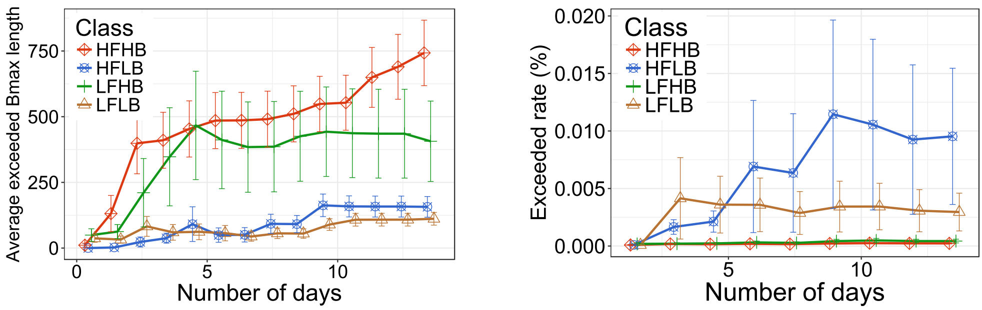
4.2 Robustness of $B_{\max}$

Since the whole characteristics of link burstiness may not be observable in a short period of time, one realistic solution is to collect packet transmission traces in a period to calculate $B_{\max}$ and only use links with robust $B_{\max}$ value that can stand for a long time. In this subsection, we discuss the robustness of $B_{\max}$ among links under two factors: the $B_{\max}$ value and burstiness frequency. The first one would influence the robustness of $B_{\max}$, since a link with low $B_{\max}$ value may imply that the burstiness event with the longest length is not observed yet in the collected trace. Burstiness frequency would also impact the robustness of $B_{\max}$, since a robust $B_{\max}$ value might not be derivable with only an observation from few burstiness events. To explore the relationship of $B_{\max}$ robustness with these two factors, we divided our collected traces into measuring set and testing set. In the measuring set, we calculated the $B_{\max}$ value and burstiness frequency for each link and classified them into different categories based on the values of these two factors. Then, in the testing set, we compared the changes of $B_{\max}$ values among different classes by observing the exceeded rate and exceeded $B_{\max}$ lengths. The exceeded rate is defined as the ratio of the burstiness events in the testing set whose lengths are longer than the calculated $B_{\max}$ value from the measuring set. The measuring and testing set is from the 21-day-long traces, which we mentioned earlier. We used our first 7 days of collected traces as measuring set and the remaining 14 days as the testing set.

For demonstration, we set $B'_{\min} = 1$ for all links and we only used links whose PRRs are higher than 90%. It is because there are 48×47 links in the traces, and some of them have very low reception rates due to long transmission range or across multiple obstacles.

4.2.1 Burstiness Frequency v.s. $B_{\max}$. The links are classified by the following methods: First, we calculated $B_{\max}$ and burstiness frequency for each link in the measuring set. We defined a burstiness event as a number of consecutive packets lost in the data trace. The burstiness frequency for a link is the average number of the burstiness events per hour in its data trace of the measuring set that is collected over the period of 7 days. Then, we divide links into four classes based on its $B_{\max}$ values and burstiness frequency. We define links whose $B_{\max}$ value is in the top 50% of all links as high $B_{\max}$ (HB), low $B_{\max}$ (LB) otherwise. Links whose burstiness frequency is in the top 50% are high burstiness frequency (HF), low burstiness frequency (LF) otherwise. Thus, there are a total of four different classes and the distribution is shown in Figure 2.

This table suggests that most of the links are either in HFHB or LFLB class, which is reasonable from the testbed of our experiment. For example, most of HFHB links are near the stairs or hallways, where people's movements are frequent and varied. However, many LFLB links connect nodes in the same office room or storage, which is relatively less interfered.



(a) Average exceeded $B_{\max}$ vs. the number of days.

(b) Exceeded rate v.s. the number of days. Exceeded rate is defined as the ratio of the burstiness lengths which are longer than the calculated $B_{\max}$ value.

Fig. 3. The exceeded lengths and times of $B_{\max}$ value among links from different classes in the testing set. The $B_{\max}$ values are calculated from the measuring dataset.

4.2.2 Exceeded Values and Times of $B_{\max}$. To address $B_{\max}$ robustness, we first focused on the exceeded $B_{\max}$ values among links in the testing set. We aimed to find links with low exceeded value, because it means the calculated $B_{\max}$ of these links are still robust in the testing set. This result is shown in Figure 3(a). The error bars are the standard error bounds for each day in the testing set. In the figure, HFHB and LFHB suffer high exceeded values rather than HFHB and HFLB. It suggests that the calculated $B_{\max}$ values on links in HFHB and LFHB are unstable and cannot stand for long periods.

Our second interest is the exceeded rate of calculated $B_{\max}$ value in the testing set, which is measured by the ratio of the burstiness events with the lengths more than the calculated $B_{\max}$ value. This result is shown in Figure 3(b). Different from the previous figure, links with low $B_{\max}$ value have higher chances that the burstiness lengths would exceed the $B_{\max}$ value. However, by referring the result of Figure 3(a), since the exceeded $B_{\max}$ values under these links are small, the exceeded rate of these links could possibly decrease in reality by setting the latency bound slightly higher than the calculated $B_{\max}$ value. In addition, we would like to emphasize that the exceeded rate of $B_{\max}$ value among all links is pretty low (at most 0.020%) according to Figure 3(b), which suggests that the $B_{\max}$ metric is a robust signature for real-time applications.

5 ALGORITHM

In this section, we show how to use $B_{\max}$ information to design a schedule with minimum latency bound for each stream. We divide our algorithm into routing and scheduling parts. In Section 5.1, we explain how to choose routes with minimum latency bound for either unicast or broadcast transmission. In Section 5.2, we show how to deal with end-to-end latency in the presence of burstiness for a single stream. In Section 5.3, we then expand the schedule of a single stream to multiple streams where both burstiness and internal interference must be handled. In Section 5.5, we present our complete algorithm for scheduling multi-stream and provide end-to-end latency bounds.

5.1 Unicast/ Broadcast Routing with $B_{\max}$

From Section 2, we assume every stream is assigned to a single route before scheduling. However, this might not be true for different systems. Also, a stream might derive high latency if the selected route for this stream goes through some links with high $B_{\max}$ values. To avoid these issues, in this subsection, we provide routing algorithms for both unicast and broadcast scenarios. In the unicast

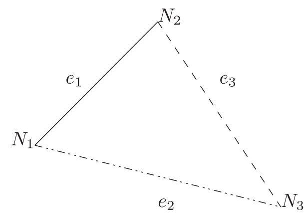


Fig. 4. The advantage of the omnidirectional transmission to provide lower latency bound.

scenario, there are multiple source-destinition pairs in the network and the algorithm is aiming to find a route for each pair such that the maximum latency bound is minimized. In the broadcast scenario, there is exactly one source that wants to broadcast the packets to all other nodes. The objective is to minimize the maximum latency bound among all other nodes. The general idea of the routing algorithms is to refer $B_{\max}$ information and select a route with links having the lowest $B_{\max}$ values. In practice, these algorithms can be implemented in Network layer to provide routing decisions for multiple streams. Consider a weighted graph $G = (V, E)$ that represents a sensor network, where sensors are represented by nodes and each edge represents a link between sensors. The weight of an edge $(i, j) \in E$ is $B_{\max}(i, j)$, where $i, j \in V$. For the unicast scenario, we simply calculate the shortest path from source to destination with respect to the $B_{\max}(i, j)$ value at each edge (i, j). This path ensures the maximum retransmission times, i.e., latency bound, is minimal. In addition, if there are multiple unicast streams in a network, how to choose routes to achieve minimum latency bounds among all streams is considered as an NP-hard problem [45]. To provide an online routing algorithm for multi-stream, here, we applied the load-balancing algorithm from J. Aspnes [3], which is to minimize the maximum traffic load while satisfying all the requests. The main idea is to introduce an exponential weight function on each edge and run the shortest path algorithm. In our case, for every assigned route, each edge e of the route would add an exponential cost. That is, $w'(e) \leftarrow w(e) + a^{B_{\max}(e)}$ for some integer a, where $B_{\max}(e)$ is the $B_{\max}$ value of edge e, $w(e)$ is the current weight, and $w'(e)$ is the updated weight of edge e. Then, the algorithm would schedule the route for next stream based on the updated weight in G. In this way, our unicast routing algorithm can provide $O(\log n)$ approximation factor to achieve minimum latency bound for multi-stream.

For the routing in the broadcast scenario, there is only one source in the network. In addition, the end-to-end latency for a job is the time that all of the nodes in the graph receive the broadcast packet. Since each node only needs to receive the packet once, the broadcast route can be represented by a tree structure. The key idea to achieve minimum latency bound is to compute a minimum spanning tree with respect to the $B_{\max}$ value. In this research, we modify BIP [41], one of the minimum energy broadcast (MEB) routing algorithms, to achieve the minimum latency bound for a job. The main idea of BIP is to take the advantage of omnidirectional transmission to achieve broadcast mission with minimum energy. We notice that this advantage is also suitable to calculate the latency bound. We use Figure 4 as an example to explain this in detail. In this figure, there is a graph G with nodes N_1, N_2, N_3, links e_1, e_2, e_3, and the weight function w. Assume N_1 is the source and it wants to broadcast a packet to N_2 and N_3. It can either send a packet to N_2 first and ask N_2 to forward the packet to N_3 or send the packet simultaneously to N_2 and N_3. Now, suppose all links are suffering the burstiness at the same time, the first way would take at most $w(e_1) + w(e_3)$ number of time slots to broadcast the packet but the second way would only cost at most $\max\{w(e_1), w(e_2)\}$ number of time slots because of the omnidirectional transmission. Thus, the minimum end-to-end latency bound in this case is the minimum of $\{w(e_1) + w(e_3), \max\{w(e_1), w(e_2)\}$. Here, we provide a broadcast algorithm that takes advantage of omnidirectional transmission.

Table 2. $B_{\max}$ and $B'_{\min}$ of Links

Link	$B_{\max}$	$B'_{\min}$
$L(1,2)$	2	2
$L(2,3)$	3	2
$L(3,4)$	3	3
$L(4,5)$	3	2
$L(7,8)$	2	2
$L(17,18)$	2	3
$L(18,19)$	1	4

Table 3. Scheduling Table with a Single Stream

Link/Time Slot	1	2	3	4	5	6	7	8	9	10	11
$L(1,2)$	S_1	S_1	S_1								
$L(2,3)$				S_1	S_1	S_1	S_1				
$L(3,4)$								S_1	S_1	S_1	S_1

Our broadcast algorithm can be simplified into two steps: *select* and *add*. The basic idea is to iterate these two steps until the spanning tree is constructed. Starting from the source node, in *select* step, it selects an edge (i, j) with the minimum weight, where i is one of the nodes in the tree and j is one of the nodes not in the tree. In *add* step, our algorithm adds the selected link (i, j) into the tree and the weight of all links whose head is i would decrease by the weight of (i, j) due to the advantage of the omnidirectional transmission. Thus, by iterating these two steps, the algorithm would construct a spanning tree for broadcasting.

The time complexity of both algorithms is under $O(|V|^2)$. For the unicast algorithm, the running time of each stream is the time of finding the shortest path, which is $O(|V|^2)$ by Dijkstra's algorithm, and the time of updating the weights, which is $O(|V|)$. For the broadcast algorithm, the running time is $O(|V|d)$, where d is the maximum degree in the graph.

5.2 Scheduling with a Single Stream

If there is only one stream in the network and there is no packet loss, then the end-to-end latency bound is the sum of per-link latencies in all the intermediate links. This is the theoretical lower bound of end-to-end latency for a single stream. But in reality, links are not ideal and packet losses occur in a burst. If a stream has n intermediate links from source to destination and kth intermediate link $L(i_k, j_k)$ has burstiness parameters $B_{\max} = b_k$ and $B'_{\min} = b'_k$ $(k = 1, 2, 3, \ldots, n)$, then on kth link, we have to allocate $b_k + 1$ time slots for the stream. So the end-to-end latency bound LB is $LB = \sum_{k=1}^{n}(b_k + 1)$.

Consider the topology in Figure 1. Assume that the links have $B_{\max}$ and $B'_{\min}$ as shown in Table 2. Now, assume that our SS consists of a single stream S_1 from Table 1 having period 20, starting time slot 1, and it goes from N_1 to N_4 using route $N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow N_4$. So, $SS = \{(S_1, N_1, N_4, RT_1, 1, 20)\}$ where $RT_1 = \{N_1, N_2, N_3, N_4\}$.

Since $B_{\max}(1, 2)$ is 2 for $L(1, 2)$, if we allocate $B_{\max}(1, 2) + 1$, i.e., 3 time slots for S_1 at node N_1, then the packet will reach to N_2 even if there is a burst. Similarly, if we allocate 4 time slots at N_2 to transmit over $L(2, 3)$ and 4 time slots at N_3 to transmit over $L(3, 4)$, then it takes only $3 + 4 + 4 = 11$ time slots to ensure that every packet of S_1 will be delivered to its destination within that time even if there is a burst in a number of links. Hence, $LB_1 = 11$. The corresponding scheduling table is shown in Table 3 below where the column represents time slots and the row represents links. It shows which stream will be transmitted at which time slot using which link. For example, S_1 will be transmitted at time slots 1, 2, and 3 using link $L(1, 2)$.

Note that multiple time slots are reserved to ensure reliability. It does not mean that N_1 is transmitting three packets of S_1 in time slots 1, 2, and 3. Rather, it means that N_1 will try in these time slots to transmit a packet of S_1 and it will stop as soon as the packet gets into the next hop node, i.e., N_2. We assume that the radio transceiver supports hardware/software acknowledgment so the receiver can acknowledge its packet reception immediately to the sender. This is a reasonable assumption based on the radio transceivers available in the market.

Notice that to provide the latency bound, each transmission on each link is required to allocate $B_{\max} + 1$ time slots, which has a potential drawback, since the burstiness may not happen in every transmission and there would be a waste of additional allocated time slots. However, the routing algorithm is prone to select links with low $B_{\max}$ value such that the waste of time slots is minimal. In general, it is a dilemma to achieve high reliability and low latency for the scheduling algorithm.

5.3 Scheduling with Multiple Streams

When we have only one stream in the network, even in the presence of burstiness, the scheduling is not complex, since the interference is not an issue. No two links can interfere with the transmission of each other in this scenario. But when multiple streams co-exist in the network, we have to take interference into account and ensure that no two interfering links transmit packets at the same time slot causing a packet loss. Note that finding the minimum average latency bound of all the streams by scheduling packet transmission considering all possible routes for every stream subject to link interference and link burstiness is an NP-Hard problem, since it is analogous to the bin packing problem [6]. Hence, we use a greedy solution based on the following principles:

(1) Schedule packet transmission up to the least common multiple (LCM) of the periods of all the streams. If the estimated end-to-end latency bounds of all streams are lower or equal to their periods, then we conclude that the stream set is schedulable.
(2) Address packet loss due to link interference as follows: First, figure out the interference pattern of the network, represented by *IM* empirically. Then, schedule packet transmission in a way to make sure that no two interfering links "transmit" packets at the same time slot.
(3) Address packet loss due to link burst as follows:
    (a) Allocate $B_{\max} + 1$ contiguous time slots for packet transmission over a link. Note that different links have different $B_{\max}$. We are assuming that the route is *least-burst-route*, as the route is the shortest path having the minimum sum of $B_{\max}$ from the source node to the destination node.
    (b) While allocating $B_{\max} + 1$ time slots, overlap at most $B'_{\min}$ streams' time slot allocation. The reason is, within a window of $B_{\max} + B'_{\min}$, there are at least $B'_{\min}$ good slots for packet transmission, and so, we should be able to transmit at least $B'_{\min}$ number of streams. Hence, we allow slots of at most $B'_{\min}$ streams to overlap. There are two conditions for it: (i) This overlapping is allowed only when packets are being transmitted over the same link. Note that we cannot overlap time slots for neighboring nodes due to link interference; (ii) While overlapping time slots of multiple streams, no two streams are allowed to be allocated the same $B_{\max} + 1$ time slots, i.e., complete overlapping is not allowed. The reason is, if two streams S_1, S_2 are allocated the same $B_{\max} + 1$ time slots, and if we lose $B_{\max}$ time slots due to a burst, then we cannot transmit packets of both S_1 and S_2 in the remaining time slot.

To explain why overlapping is important and how it is done, consider the following example: Assume that Stream Set SS consists of two streams, S_1 and S_2, both going from node N_1 to N_2 using route $N_1 \rightarrow N_2$ through the link L(1, 2). Assume that both streams have period 20 and starting time slot 1. Also assume that $B_{\max}(1, 2) = 3$ and $B'_{\min}(1, 2) = 2$ for the link L(1, 2). A schedule without overlap is shown at Table 4 where average latency bound per stream is $(4 + 8)/2 = 6$ time slots. Compare it with a schedule with overlapping in Table 5 where average latency bound per stream is

Table 4. Schedule with Non-overlapping

Link/Time Slot	1	2	3	4	5	6	7	8
$L(1,2)$	S_1	S_1	S_1	S_1				
$L(1,2)$					S_2	S_2	S_2	S_2

Table 5. Schedule with Overlapping

Link/Time Slot	1	2	3	4	5
$L(1,2)$	S_1	S_1, S_2	S_1, S_2	S_1, S_2	S_2

Table 6. Schedule with Complete Overlapping

Link/Time Slot	1	2	3	4
$L(1,2)$	S_1, S_2	S_1, S_2	S_1, S_2	S_1, S_2

Table 7. Schedule with Maximum Overlapping

Link/Time Slot	1	2	3	4	5	6
$L(1,2)$	S_1	S_1, S_2	S_1, S_2, S_3	S_2, S_3, S_4	S_3, S_4	S_4

$(4 + 5)/2 = 4.5$ time slots. Note that we could not do complete overlapping as in Table 6. Because, as it is mentioned earlier, if we lose $B_{\max} = 3$ time slots due to a burst, then we cannot transmit packets of both S_1 and S_2 in the remaining time slot.

Overlapping time slots raises another issue. Since we can transmit only one packet at a time and the packet transmission process is completely deterministic, we have to prioritize the streams to be transmitted in the overlapped time slots. Our *prioritizing rule* works as follows: If multiple streams are scheduled to be transmitted at the same time slot, transmit the not-yet-transmitted stream that has the closest ending time slot. We will see its use in the next example.

To illustrate how many streams can be overlapped, consider the following example: Assume that Stream Set SS consists of four streams S_1, S_2, S_3, and S_4 all going from node N_1 to N_2 using route $N_1 \rightarrow N_2$ through the link L(1, 2). Assume that all the streams have period 20 and starting time slot 1. Also assume that $B_{\max}(1,2) = 2$ and $B'_{\min}(1,2) = 4$ for the link L(1, 2). Table 7 demonstrates a schedule that shows within a window of size $B_{\max}(1,2) + B'_{\min}(1,2) = 2 + 4 = 6$, we can overlap at most $B'_{\min}(1,2) = 4$ streams. This schedule will always work if nodes transmit packets according to the prioritizing rule. For example, if packet transmission fails during time slots 1 and 2, packets of streams S_1, S_2, S_3, and S_4 will be transmitted at time slots 3, 4, 5, and 6, respectively. If packet transmission fails at time slots 2 and 4, then packets of streams S_1, S_2, S_3, and S_4 will be transmitted at time slots 1, 3, 5, and 6. So, even if packet transmission of any combination of size $B_{\max}(1,2)$ fails within a window of $B_{\max}(1,2) + B'_{\min}(1,2)$, all the packets of all the streams will get through to the next node if we allocate time slots according to the principles mentioned earlier and nodes transmit packets according to the prioritizing rule as it is proved in the next subsection.

5.4 Correctness Proof

The correctness of our algorithm depends on the following theorem:

Theorem: *If we overlap packet transmissions of at most b' streams in $(b + b')$ time slots, all going through the same link having $B_{\max} = b$, $B'_{\min} = b'$, each stream having $(b + 1)$ contiguous time slots allocated without complete overlapping with any other stream, then all of the streams will get through even if there is a burst of at most b time slots (not necessarily contiguous) if we transmit packets according to our prioritizing scheme.*

Proof: The proof is by contradiction. For a contradiction, assume that stream S_i could not be transmitted due to a burst. Since we can lose at most b time slots due to a burst, there has to be a time slot (we call a "good slot") when there was no burst, but still S_i was not transmitted. The reason for that can only be some other stream S_j was transmitted at that time slot. Note that the good slot cannot be the last sending time slot of S_i. Because, at that time slot, stream S_i has the highest priority for transmitting a packet. So, no other stream S_j can be transmitted at that time slot. So, now we have to consider one remaining possibility. There was a burst at the last sending time slot of S_i and among the previous b slots of S_i, $(b-1)$ slots are gone due to a burst leaving one good slot and at that time slot some other stream S_j was transmitted for the priority scheme. Note that this cannot happen either, because as S_j has higher priority than S_i, S_j has started time slots before S_i did, and hence it should be transmitted even before the burst happens. Because we are overlapping at most b' streams in any time window of $(b + b')$ slots and there has to be at most b' good slots between any consecutive bursts. So, S_j will be transmitted in the good time slots of the previous window and S_i will be transmitted within this time window. So, there is no way S_i will not be transmitted due to a burst.

5.5 Scheduling Algorithm

Algorithm 2 schedules packet transmission up to the LCM of the periods of streams in a central sensor network based on the principles and conditions stated in Section 5.2. It takes the Stream Set SS of n streams (as defined in Section 2), topology, Interference Matrix IM, burstiness parameters $B_{\max}$, $B'_{\min}$ of every link as inputs, and returns either $true$ if all the streams are schedulable or $false$ if they are not. If all the streams are schedulable, then LB_i holds the latency bound of stream S_i.

Let L_i be the last allocated time slot for S_i in the scheduling algorithm. So, initially, we have $L_i = (ST_i$ - 1) (at line 4) for every stream S_i. We need to adjust the starting time ST_i of stream S_i when its period P_i is over (at line 40) to correctly compute the latency bound LB_i (at line 29). Let N_i be the node that has received the last transmitted packet of S_i. So, initially $N_i = SRC_i$ (at line 5), which is the source node of stream S_i. Assume that N_{i+1} is the next node to which N_i has to transmit a packet of S_i. Note that N_{i+1} can be easily determined from the route of S_i.

We put *DeferThreshold* = 2 (used at line 20). If you use a higher value for it, then the scheduler runs faster, but the generated latency bound may also rise. We are deferring time slot allocation to only those streams that cannot take advantage of overlapping time slots with other streams. The reason for deferring time slot allocation is to increase parallelism over the link from N_i to N_{i+1}. This is a kind of lazy approach for allocating, because we are hoping that it is possible that some other streams may show up and can be allocated in these time slots that can exploit parallelism.

Table 8 shows the whole scheduling for the example problem we are working with from Section 2 and B, B' of Table 2. Here S_1 and S_2 share time slots at links $L(2,3)$ and $L(3,4)$. The latency bound of the streams is shown in Table 9.

Our algorithm exploits parallelism in two ways. The first one is, multiple streams can be transmitted at the same time if there is no interference in their transmission as it is seen in S_1 and S_4 at time slots 1, 2, and 3 in Table 8. The second way is, when two streams are going to the same link, they can share at most $B_{\max}$ time slots, as it is observed between S_1 and S_2 in time slots 5, 6, and 7 in Table 8.

The running time of the algorithm is $O(LP)$ where L is the LCM of the periods of the streams and P is the sum of the periods of the streams. The reason is, the for loop at line 8 takes $O(L)$ time and the for loops at lines 12 and 17 together takes $O(P)$ time. Note that $O(LP)$ depends on mainly the periods of the streams, and it can be exponential if the streams have periods that are prime numbers.

ALGORITHM 2: Centralscheduler(SS, topology, IM, $B_{\max}$, $B'_{\min}$ of every link) - Part 1

```
n ← |SS|
lcm ← LCM(P_1, P_2, P_3, ... P_n)
for i ← 1 to n do
    L_i ← ST_i − 1,
    N_i ← SRC_i,
    LB_i ← 0
end for
for t_1 ← 1 to lcm do
    if All the streams are scheduled then
        break
    end if
    for i ← 1 to n do
        if S_i is already scheduled then
            continue
        end if
        if t_1 = L_i then
            for t_2 ← t_1 + 1 to ST_i + P_i do
                (b, b′) ← (B_max, B′_min) of link L(N_i, N_{i+1})
                if it is possible to allocate (b + 1) contiguous time slots starting from t_2 by following the principles and conditions then
                    if S_i is not making any overlap with any streams in these time slots AND (t_2 − t_1) > DeferThreshold then
                        L_i ← t_2 − 1
                        break
                    else
                        Allocate these (b+1) time slots for S_i in N_i
                        L_i ← t_2,
                        N_i ← N_{i+1}
                        if N_{i+1} is the destination node for S_i then
                            Consider that S_i is scheduled
                            LB_i ← max(LB_i, t_2 + b + 1 − ST_i)
                        end if
                        break
                    end if
                end if
            end for
        end if
```

6 SIMULATION

In this section, we simulated the routing algorithms for broadcast and unicast scenarios, which we proposed in Section 5.1, and compared the performance with other baseline algorithms. This comparison includes end-to-end deadline miss ratio, energy cost, and the estimated latency bound. For each transmitted packet, there is a deadline, which is the value of the estimated end-to-end latency bound. During the transmission, if a packet is not delivered to its destination node within its latency bound, it is considered to miss its deadline. We use BIP [41] as the baseline for the broadcast scenario and ETX [7] for the unicast scenario. BIP is the algorithm we introduced in

Table 8. Scheduling with Multiple Streams

Link/Time Slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$L(1, 2)$	S_1	S_1	S_1																	
$L(2, 3)$				S_1	S_1, S_2	S_1, S_2	S_1, S_2	S_2												
$L(3, 4)$									S_1	S_1, S_2	S_1, S_2	S_1, S_2	S_2							
$L(4, 5)$														S_2	S_2	S_2	S_2			
$L(7, 8)$														S_3	S_3	S_3				
$L(8, 9)$																		S_3	S_3	S_3
$L(17, 18)$	S_4	S_4	S_4								S_4	S_4	S_4							
$L(18, 19)$				S_4	S_4									S_4	S_4					

ALGORITHM 3: Centralscheduler(SS, topology, IM, $B_{\max}$, $B'_{\min}$ of every link) - Part 2

        if $mod(t_1, P_i) = 0$ **then**
            if S_i is not scheduled yet **then**
                return $false$
            else
                $ST_i \leftarrow ST_i + P_i$,
                $N_i \leftarrow SRC_i$
            end if
        end if
    end for
end for
if all the streams are not scheduled within the lcm **then**
    return $false$
end if
return $true$

Section 5.1 and ETX is basically finding the shortest path in respect to the expected number of transmission counts. The internal interference between nodes is measured by interference matrix and we give a detail explanation in Section 7.2. In addition, all the nodes are using a TDMA-based schedule that has both sleep and awake modes.

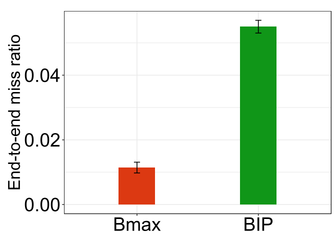
The simulation dataset we used is the 21-day-long transmission traces we collected in Section 4. We set the first 7 days of the data as the measuring set that is used for measuring $B_{\max}$ values (with $B'_{\min}$ selected to 1) and ETX values. The rest of data is used for the testing set that cannot be observed by the algorithms.

For the broadcast scenario, we set the source as one of the nodes in the testbed and used the routes that are generated from either our algorithm or BIP. Then, we simulated the broadcast transmission 1 K times and used the testing set to determine whether the delivered packet is received at each node. During each transmission, every link is capable of retransmitting the packet multiple times until it meets the $B_{\max}$ value of the link. If a node cannot receive the packet before the deadline, then we count this packet as a loss, and this node cannot forward this packet to other nodes as well.

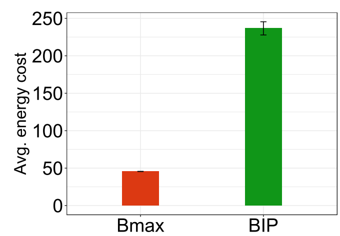
Figure 5 shows the performance comparison between our algorithm (which we labeled it as $B_{\max}$ in the figure) and BIP. In Figure 5(a), it shows the average end-to-end miss ratio per broadcast transmission in all 47 sources. Overall, our algorithm has 5.02 times lower miss ratio than BIP. In addition, the routes chosen by our algorithm are more energy-efficient than BIP as well. This result

Table 9. Latency Bound for Each Stream

Stream	Latency Bound
S_1	12, i.e., $\max(12-1+1,0)$
S_2	17, i.e., $\max(17-1+1,0)$
S_3	20, i.e., $\max(20-1+1,0)$
S_4	5, i.e., $\max((5-1+1),(15-11+1),0)$



(a) Average ratio of non-received nodes for broadcasting a packet in all 47 different sources.



(b) Average energy cost (retransmission times) for broadcasting a packet in all 47 different sources.

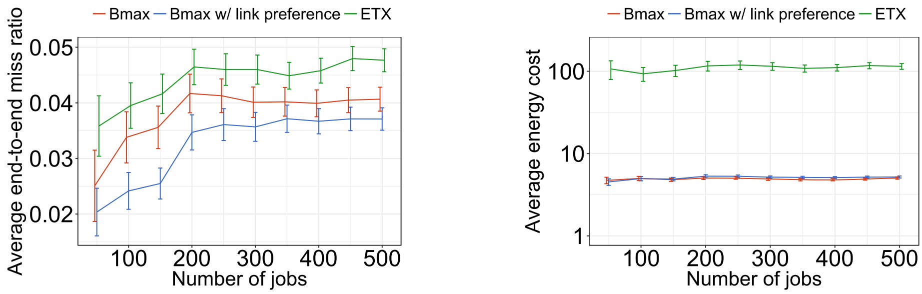
Fig. 5. Compare the deadline miss ratio and energy cost under $B_{\max}$ and BIP in broadcast scenario.

is shown in Figure 5(b), where the average energy cost is defined as the number of (re)transmission times per broadcast transmission in all 47 sources. The average energy cost from the routes chosen by our algorithm is 48 transmission times per broadcast but 232 transmission times from the routes chosen by BIP.

However, for the unicast setting, we simulated a number of jobs from 50 to 500 with random sources and sinks. Similar to the broadcast simulation, for each job we simulated the stream transmissions 1 K times from the testing set to determine whether a packet transmission is successful. The setting of link retransmission ability is the same as the broadcast setting. However, unlike broadcast scenario that broadcasts one source at a time, all the jobs would transmit at the same time in unicast scenario. Therefore, the algorithms need to consider the load-balance of the network to avoid congestion. Last, besides comparing $B_{\max}$ and ETX, we also add one more algorithm "$B_{\max}$ with link preference," which tries routing in graph $G' = (V, E')$ first, where edge set E' includes LFLB links only. If it cannot find a route for a job in G', it then tries routing in the original graph G. The reason to do so is because links in this class have both low and robust $B_{\max}$ value according to our analysis in Section 4.2.

Figures 6 and 7 show the performances with different routing algorithms. Figure 6(a) reports the end-to-end miss ratio. $B_{\max}$ with link preference shows most reliable transmission, but it is not significant, since the miss-ratio difference is only 1.5% compared with ETX. However, the routes ETX uses are not all energy-efficient, which is reported in Figure 6(b). For each job, routes from ETX take 110 transmission times on average, where the routes from our algorithms take only 5 transmission times.

The scalability of our algorithms is also evaluated in this simulation. Figure 7 reports the maximum latency with 1 K samples after scheduling hundreds of streams. We have two observations in this figure. First, the routes of our algorithms have low maximum end-to-end latency (from 6 to 40 seconds) compared with the routes of ETX (from 200 to 2 K seconds). Second, our algorithms are scalable, since the maximum latency increases little when the number of scheduled jobs increase to 500.



(a) Average end-to-end deadline miss ratio per job from 50 to 500 jobs.

(b) Average energy cost (retransmission times) per job from 50 to 500 jobs.

Fig. 6. Compare the deadline miss ratio and energy cost under $B_{\max}$, $B_{\max}$ with link preference on LFLB and HFLB links, and ETX metric.

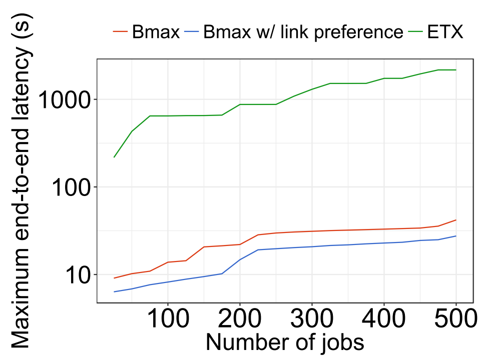


Fig. 7. The maximum latency (second) on streams after scheduling jobs from 50 to 500.

Table 10. Link Classes Distribution of the Routes from Different Algorithms

Routing Algo.	HFLB	HFHB	LFLB	LFHB
$B_{\max}$	24.2%	14%	54.3%	7.5%
$B_{\max}$ w/ preference	18%	6.7%	70.1%	5.2%
ETX	14.1%	42.4%	31.2%	12.3%

To give another aspect of the difference in routing decisions between these algorithms, we provide the composition of links for routes in Table 10. We find out that there are 42.4% of HFHB links being chosen in the routes from ETX, whereas there are only 14% of HFHB links chosen from $B_{\max}$ and 6.7% from $B_{\max}$ with link preference. According to our previous analysis, the latency bound of HFHB links is high and causes high retransmission rate when these links are in the burst. Since our algorithms take account $B_{\max}$ values when finding the routes, we avoid using these high latency bound links such that each scheduled job can have a lower energy cost and latency bound.

7 EXPERIMENT

In this section, we evaluate our algorithm in terms of end-to-end deadline miss ratio (which is defined in Section 6) and latency bound. At first, we describe the experimental setup. Then, we

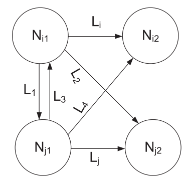


Fig. 8. Measuring *IM*.

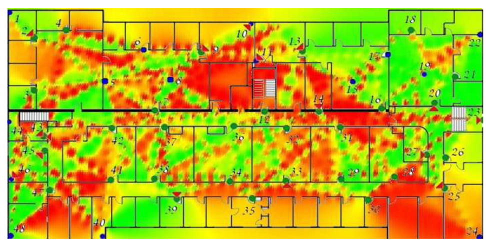


Fig. 9. Spatial Distribution of $B_{\max}$.

describe how we measure burstiness parameters $B_{\max}$ and $B'_{\min}$ of every link and the interference matrix, IM. Then the effect of $B_{\max}$ and $B'_{\min}$ to estimated latency bound and end-to-end deadline miss ratio are evaluated.

7.1 Measuring Burstiness

To understand the effect of link burstiness on packet transmission and to calculate the model parameters $B_{\max}$ and $B'_{\min}$ of every link, we transmit 3.6 M packets per link. The packet transmission rate is around 200 packets per second where every node tries to transmit the next packet as soon as possible with a zero inter-packet interval. To avoid possible collisions and interference, we allow only one node to transmit packets at a time. There are 48 nodes, each node takes turn for packet transmission and at each turn a node transmits around 1,200 packets continuously. We are supposed to transmit $21 \times 24 \times 60 \times 60 \times 200/48 = 7{,}560{,}000$ packets per link in 21 days with the specified packet transmission rate. But we could only transmit 3.6 M packets per link, because it takes time to fetch the sequence number of the received packets from all other nodes when a node finishes its turn of packet transmission. After collecting sequence numbers of received packets on every link, we have a long data trace that is used to calculate model parameters $B_{\max}$ and $B'_{\min}$ of every link using algorithm 1. The spatial distribution of $B_{\max}$ of the links at the testbed is shown in Figure 9, where green zones represent links having low $B_{\max}$, and red zones represent links having high $B_{\max}$. We find that most of the red zones fall within places where peoples' movement varies a lot, e.g., stairs, restrooms, copy-room, conference rooms, and kitchen.

7.2 Measuring Interference

$$IM(i,j) = \begin{cases} 1 & \text{if link } L_i \text{ and link } L_j \text{ are in interference range,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The external interference, e.g., interference caused by Wi-Fi networks, is captured through the $B_{\max}$ and $B'_{\min}$ parameters. To address internal interference, i.e., interference caused by packet transmission through neighboring links at the same time, we define a $k \times k$ interference matrix, IM for a network of k links that specifies which links potentially interfere with others:

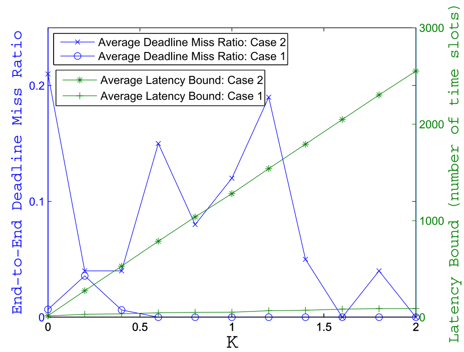


Fig. 10. Effect of $B_{\max}$ on end-to-end deadline miss ratio and latency bound.

If $IM(i,j) = 1$, then scheduler will make sure that two packets are not scheduled for transmission over links L_i and L_j at the same time. The computation of IM is based on Packet Reception Ratio (PRR), and PRR is computed from the same data trace that has been used to measure the burstiness parameters. We explain how IM is computed by using Figure 8. Assume that link L_i spans from node N_{i1} to node N_{i2} and link L_j spans from node N_{j1} to node N_{j2}. We compute PRR of every link and then set $IM(i,j)$ to 1 if any of the links L_1, L_2, L_3, or L_4 have a PRR greater than PRR_t, a threshold. We set PRR_t to 0.3 in our experiment. This is a conservative model that hurts the latency bound, but improves the miss ratio.

7.3 Effect of $B_{\max}$

In this section, we describe the effect of $B_{\max}$ on end-to-end deadline miss ratio and latency bound. In our problem definition described in Section 2, the streams did not have any deadline associated with them and we define the deadline of the streams to be their generated latency bound. We evaluate it on the testbed. We compute the burstiness parameters $B_{\max}$, $B'_{\min}$ and the interference matrix IM of the actual testbed network exactly the way specified in Sections 7.1 and 7.2. To ensure packet transmission experiences the same burst-behavior, at this experiment, we maintain the same packet transmission rate of around 200 packets per second with a zero inter-packet interval that we use to measure link burstiness as described in Section 7.1.

To disable the effect of $B'_{\min}$, we select $B'_{\min}$ to 1 for all links. We define a multiplying factor K to demonstrate a trade-off between latency bound and end-to-end deadline miss ratio and instead of allocating $B_{\max} + 1$ time slots per link, we are allocating $B_{\max} \times K + 1$ time slots for different values of K ranging from 0 to 2.

We run the experiment by considering two cases. In case 1, we use the whole testbed for evaluation. This case represents an actual deployed system. But in case 2, we remove the links that have the top 25% of lowest $B_{\max}$. This case represents another system where links are not as good and we are forced to use some non-stationary links. Since we are not using the top 25% of the links in case 2, the workloads for these two cases are different. But for each case, for each value of K, we generate 10 different workloads and the average values are plotted in Figure 10.

To generate a workload, we randomly select 10 pairs of nodes as sources and destinations to generate 10 random streams, and choose the route of those streams as the shortest path from the source node to the destination node having minimum sum of $B_{\max} \times K + 1$ time slots. The starting time for every stream is set to its first time slot, and period is randomly selected from a pool of even numbers up to 800 time slots for case 1 and 6 K time slots for case 2. If we consider a packet transmission rate of 200 packets per second, and allocate 5 msec per time slot, then the generated latency bound at K = 1 is $51 \times 5 = 255$ msec for case 1, which is practical for the implementation of control loops in factory automation.

To time-synchronize the nodes, we use an RBS-style synchronization technique [9]. Since nodes may experience clock drift, to keep the nodes synchronized over time, we need to allocate time slots for periodic broadcasting of the time-synchronization message.

Figure 10 shows the effect of $B_{\max}$ on latency bound. From the figure, we observe that as K increases, the average latency bound increases linearly for both cases. The reason is, as K increases, $B_{\max}$ of all the links increases linearly and, since $B'_{\min}$ is one for all the links, there is no overlapping of time slots to reduce latency bound.

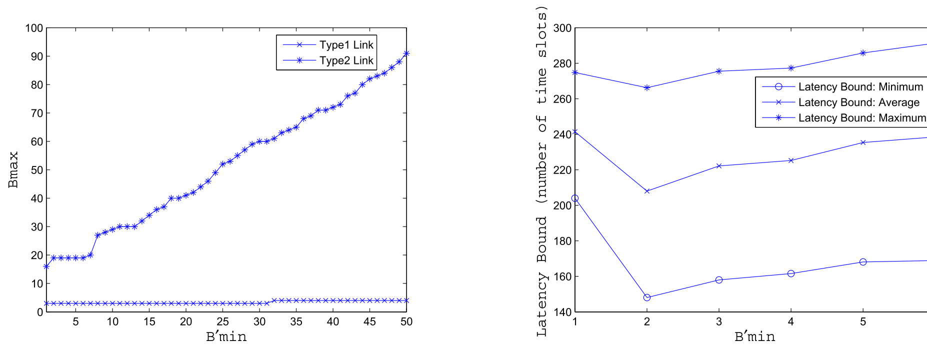
Figure 10 also shows the effect of $B_{\max}$ on end-to-end deadline miss ratio in the testbed. For case 1, with the full testbed, we observe that although there are some fluctuations in the end-to-end deadline miss ratio for different values of K up to 0.6, the end-to-end deadline miss ratio becomes 0 after $K = 0.6$. This implies that when there are many good links in a network, our solution obtains 100% packet delivery within the latency bounds; surprisingly, we observe that even before $K =$ 1.0. Note that our generated latency bound (at $K = 1$) is within 14.2% of the minimum latency (at $K = 0.6$) for which we observe zero end-to-end deadline miss ratio. So, the generated latency bound is relatively tight. Also, if we allocate $0.6 * B_{\max} + 1$ time slots instead of $B_{\max} + 1$ time slots, we can save 12.4% of average latency and can still make all the deadlines. We may not want to set $K < 1$ for hard real-time applications, but it can be very useful to control the trade-off between end-to-end deadline miss ratio and latency bound for soft real-time applications.

For case 2, with the top 25% links removed, we observe a different result. Here, missing of deadlines continues beyond $K \geq 1$ even though we are allocating $B_{\max} \times K + 1$ time slots. The reason behind this is, links having high $B_{\max}$ are typically susceptible to the changes in the physical environment (as shown in Figure 9) and to accurately characterize these links, empirical data should be collected over more than 21 days. Since we are choosing least-burst-route for packet transmission, in case 1, we have sufficient number of links having low $B_{\max}$ for packet transmission; while in case 2, we are forced to choose some links having high $B_{\max}$. The burst size may become bigger than the measured one no matter how long the empirical characterization is. In this case, we can address the problem with two different approaches: packet recovery and link adaptation, as discussed in Section 8.

7.4 Effect of $B'_{\min}$

In this section, we discuss the effect of $B'_{\min}$ on the latency bound. For a particular stream, its latency bound will either increase or remain same if $B'_{\min}$ is increased for every link that the stream goes through. The effect depends on the quality of the link. For high-quality links, either stationary or asymptote stationary, the response to the increase of $B'_{\min}$ is of two types. So, we call these links as Type1 and Type2 links. The response of the two types of links to different values of $B'_{\min}$ is shown in Figure 11(a).

In a Type1 link, as we see from Figure 11(a), $B_{\max}$ increases very slowly with the increase of $B'_{\min}$. As a result, the latency bound remains the same for some time and increases very slowly for the increase of $B'_{\min}$. In the case of a Type2 link, $B_{\max}$ increases rapidly with the increase of $B'_{\min}$, and that is why the latency bound for a particular stream also increases rapidly. The reason behind this behavior depends on the link quality. After transmitting 3.6 M packets over every link, we compute $B_{\max}$ for different values of $B'_{\min}$ by using Algorithm 1. We observe that Type1 links are so good that even if we increase $B'_{\min}$, $B_{\max}$ remains almost the same and increases very slowly with keeping at least $B'_{\min}$ number of good slots for packet transmission in every possible window of size $B_{\max} + B'_{\min}$. In the case of Type2 links, as $B'_{\min}$ increases, to keep at least $B'_{\min}$ number of



(a) Variation of $B_{\max}$ for Different $B'_{\min}$ (b) Effect of $B'_{\min}$ on Latency Bound

Fig. 11. The effect of $B'_{\min}$ on the latency bound.

good slots in every possible window of size $B_{\max} + B'_{\min}$, we need to increase the window size by increasing $B_{\max}$ rapidly.

However, for a set of streams, the average latency bound may increase, decrease, or remain same as a result of the increase of $B'_{\min}$ of every link depending on quality of the links, either Type1 or Type2, and spatio-temporal overlapping of the streams. If a large number of streams pass through a dense network with some spatial and temporal overlapping in transmission, then we observe a decrease in average latency as $B'_{\min}$ increases, but after all the overlapping advantage is exploited, the average latency bound starts to rise again. We observe a similar result with 10 randomly generated streams for the testbed for which average latency bound decreases up to $B'_{min} = 2$ and then it starts to rise again as is shown in Figure 11(b). We run the experiment five times for each value of $B'_{\min}$ and the minimum and maximum values of the latency bound of five runs are also plotted in the figure. It clearly indicates that we can minimize the average latency bound of a set of streams by an intelligent selection of $B'_{\min}$ of the links.

8 FUTURE WORK

In this section, we discuss the possible improvements of our mechanism and routing/scheduling algorithm. We consider the evaluation of these following approaches as future work.

8.1 Update $B_{\max}$ at Runtime

Although $B_{\max}$ and $B'_{\min}$ are adaptive to capture worst-case link reliability for networks that have to support time-critical data delivery, it cannot be guaranteed that link characteristics are invariant. For example, links may have interference due to an interferer that appears temporarily in some areas of the network. The burst-behavior of the wireless links may change due to a change in the physical environment, e.g., node failure, node replacement, or unexpected obstacles. Therefore, it is necessary to measure and update link quality after a period of time. There are some works such as Burstprobe [5] that measure link burstiness online. However, their work needs additional time slots to remeasure max burst in the schedule, which increase redundancy of the network. Here, we propose a method to update $B_{\max}$ according to the transmission results of each link. We argue that $B_{\max}$ values can keep being updated and it is not necessary to allocate large and long data traces every time to calculate new $B_{\max}$ values.

Specifically, assume all streams are scheduled according to our proposed algorithm—each sensor would monitor the transmission results at its adjacent links. If a transmission failure happened at a specific link, then we update its $B_{\max}$ to $B_{\max} + t$, where t is an arbitrary integer, and reschedule the stream for its updated $B_{\max}$. We can also apply the doubling technique [17] to find the updated $B_{\max}$ sooner. Although it is costly to reschedule streams, we argue there are only a few links that need to update their $B_{\max}$ in one time if we only select links with robustness $B_{\max}$, as shown in Section 4.2.

8.2 Selection of $B'_{\min}$

In this article, we set $B'_{\min}$ to 1 for all the links during our evaluation in Section 7.3. In Section 7.4, we set $B'_{\min}$ of all the links to 1, 2, 3, 4, 5, 6 to show how $B'_{\min}$ affects latency bound. However, we did not set different $B'_{\min}$ to different links. The minimum value of $B'_{\min}$ is 1 and the maximum value of $B'_{\min}$ is n, which is total number of streams in the system. One could compute $B_{\max}$ values of all possible $B'_{\min}$ values $(1, 2, 3, \ldots n)$ of all the links and call Algorithm 2 with different input configurations repeatedly to find a better latency bound. However, this approach is computationally very expensive, and we leave the selection of optimal $B'_{\min}$ of each link to provide a better latency bound as future work.

8.3 Energy Efficiency

Although the scheduler allocates redundant time slots for packet transmission, since we use least-burst-routing, most of the selected links are very good and hence single packet transmission suffices in most of the cases. Consider the case of a sensor network using a TDMA-based schedule with sleep and active modes; the radio needs to be used only for single packet transmission time in most of the cases and then goes to sleep mode to save the energy. In other cases of packet loss, the transmitter and receiver stay on until the packet is received.

However, there are also other approaches that use statistical models such as Markov chain to predict burstiness. The transmitter would turn off the transmission temporarily when they foresee the incoming burst [16, 36]. Their opportunistic approaches can save energy but do not provide any reliability guarantee and energy bound. Our solution can also potentially explore duty-cycle-based transmission schedules for more energy savings.

8.4 Specific Reliability with Latency Bound

$B_{\max}$ metric considers stream latency when links are all in the worst-case scenario, which provides a latency bound for reliability guarantee. In addition, if we combine with the probability theory, it is possible to provide the latency bound with probability reliability guarantee. For example, we can consider a stream that only requires 90% success rate and derive a new minimum window $B'_{\max}$ of each link on its route that has an equal or lower value compared with the original $B_{\max}$ of each link. To calculate the new $B'_{\max}$, we can use the algorithm that is provided in Reference [13]. For the scheduling, we can keep using our algorithm, since the input is still a route with a number of allocated time slots on each link. In this way, we can also provide routing and scheduling algorithms among streams with specific reliability guarantee with lower latency bounds.

8.5 System Integration

$B_{\max}$ metric and our proposed algorithms can be applied to any systems based on TDMA transmission. For example, WirelessHART [30, 34] is one of the systems using time slotted channel hopping (TSCH). It is adaptive to integrate with our scheduling algorithm under data-link layer. In addition, the routing schema in WirelessHART supports both broadcast and unicast routings that can be applied to our routing algorithm with a little modification. In graph routing of WirelessHart, the edges are created by the network manager and downloaded to each device. Thus, each device has full knowledge of the connectivity of the graph. To integrate with our routing algorithm, each device needs to provide $B_{\max}$ information of its links to the network manager such that all devices can have the $B_{\max}$ information of links in the graph. Therefore, each device can further generate the routes based on our broadcast/unicast algorithm.

9 CONCLUSIONS

We have presented a new analysis technique that provides exact characterization and classification of the network links subject to link burstiness and interference. By considering link burstiness before scheduling, our routing algorithm provides reliable routes with minimum latency bounds for streams that are suitable for CPS. The scheduling algorithm is also novel, since it accounts for both link burstiness and interference and offers a schedule for packet transmission that produces an upper bound on the latencies of the streams. We expect that this approach will improve the use of wireless networks in CPS.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their insightful comments.

REFERENCES

[1] Zeeshan Ansar, Jianjun Wen, Eyuel Debebe Ayele, and Waltenegus Dargie. 2015. An efficient burst transmission scheme for wireless sensor networks. In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. ACM, 151–155.

[2] Zeeshan Ansar, Jianjun Wen, and Waltenegus Dargie. 2016. Efficient online burst transmission scheme for wireless sensor networks. In *Proceedings of the 25th International Conference on Computer Communication and Networks (ICCCN'16)*. IEEE, 1–9.

[3] James Aspnes, Yossi Azar, Amos Fiat, Serge Plotkin, and Orli Waarts. 1997. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. ACM* 44, 3 (1997), 486–504.

[4] Md Zakirul Alam Bhuiyan, Guojun Wang, Jiannong Cao, and Jie Wu. 2015. Deploying wireless sensor networks with fault-tolerance for structural health monitoring. *IEEE Trans. Comput.* 64, 2 (2015), 382–395.

[5] James Brown, Ben McCarthy, Utz Roedig, Thiemo Voigt, and Cormac J. Sreenan. 2011. Burstprobe: Debugging time-critical data delivery in wireless sensor networks. In *Proceedings of the European Conference on Wireless Sensor Networks*. Springer, 195–210.

[6] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. 1997. Approximation algorithms for bin packing: A survey. In *Approximation Algorithms for NP-hard Problems*. PWS Publishing, Boston, MA.

[7] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. 2003. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'03)*.

[8] Felix Dobslaw, Tingting Zhang, and Mikael Gidlund. 2016. End-to-end reliability-aware scheduling for wireless sensor networks. *IEEE Trans. Industr. Inform.* 12, 2 (2016), 758–767.

[9] Jeremy Elson, Lewis Girod, and Deborah Estrin. 2002. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI'02)*.

[10] Ali Asghar Ghaemi. 2017. A cyber-physical system approach to smart city development. In *Proceedings of the IEEE International Conference on Smart Grid and Smart Cities (ICSGSC'17)*. IEEE, 257–262.

[11] Antonio Gonga, Olaf Landsiedel, Pablo Soldati, and Mikael Johansson. 2012. Revisiting multi-channel communication to mitigate interference and link dynamics in wireless sensor networks. In *Proceedings of the IEEE 8th International Conference on Distributed Computing in Sensor Systems (DCOSS'12)*. IEEE, 186–193.

[12] Gregory Hackmann, Weijun Guo, Guirong Yan, Zhuoxiong Sun, Chenyang Lu, and Shirley Dyke. 2014. Cyber-physical codesign of distributed structural health monitoring with wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.* 25, 1 (2014), 63–72.

[13] Yang Hao-Tsung, Liu Kin Sum, Gao Jie, Lin Shan, Munir Sirajum, Whitehouse Kamin, and Stankovic John. 2017. Reliable stream scheduling with minimum latency for wireless sensor networks. In *Proceedings of the 14th IEEE International Conference on Sensing, Communication, and Networking (SECON'17)*. IEEE.

[14] Tian He, John A. Stankovic, Chenyang Lu, and Tarek F. Abdelzaher. 2003. SPEED: A stateless protocol for real-time communication in sensor networks. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS'03)*.

[15] Woochul Kang, Krasimira Kapitanova, and Sang Hyuk Son. 2012. RDDS: A real-time data distribution service for cyber-physical systems. *IEEE Trans. Industr. Inform.* 8, 2 (2012), 393–405.

[16] Song Min Kim, Shuai Wang, and Tian He. 2015. cETX: Incorporating spatiotemporal correlation for better wireless networking. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems.* ACM, 323–336.

[17] Peter M. Kogge and Harold S. Stone. 1973. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Trans. Comput.* 100, 8 (1973), 786–793.

[18] Edward A. Lee. 2008. Cyber physical systems: Design challenges. In *Proceedings of the 11th IEEE Symposium on Object Oriented Real-time Distributed Computing (ISORC'08).* IEEE, 363–369.

[19] Jay Lee, Behrad Bagheri, and Hung-An Kao. 2015. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufact. Lett.* 3 (2015), 18–23.

[20] Husheng Li, Lifeng Lai, and H. Vincent Poor. 2012. Multicast routing for decentralized control of cyber physical systems with an application in smart grid. *IEEE J. Select. Areas Commun.* 30, 6 (2012), 1097–1107.

[21] Xiaomin Li, Di Li, Jiafu Wan, Athanasios V. Vasilakos, Chin-Feng Lai, and Shiyong Wang. 2017. A review of industrial wireless networks in the context of industry 4.0. *Wirel. Netw.* 23, 1 (2017), 23–41.

[22] Shan Lin, Gang Zhou, Kamin Whitehouse, Yafeng Wu, John A. Stankovic, and Tian He. 2009. Towards stable network performance for wireless sensor networks. In *Proceedings of the IEEE Real-Time Systems Symposium (RTSS'09).*

[23] Kin Sum Liu, Tyler Mayer, Hao Tsung Yang, Esther Arkin, Jie Gao, Mayank Goswami, Matthew P. Johnson, Nirman Kumar, and Shan Lin. 2017. Joint sensing duty cycle scheduling for heterogeneous coverage guarantee. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'17).* IEEE, 1–9.

[24] Chenyang Lu, Abusayeed Saifullah, Bo Li, Mo Sha, Humberto Gonzalez, Dolvara Gunatilaka, Chengjie Wu, Lanshun Nie, and Yixin Chen. 2016. Real-time wireless sensor-actuator networks for industrial cyber-physical systems. *Proc. IEEE* 104, 5 (2016), 1013–1024.

[25] Muhammad Adeel Mahmood, Winston K. G. Seah, and Ian Welch. 2015. Reliability in wireless sensor networks: A survey and challenges ahead. *Comput. Netw.* 79 (2015), 166–187.

[26] Michael Maiwald, Patrick Gräßer, Lukas Wander, Nicolai Zientek, Svetlana Guhl, Klas Meyer, and Simon Kern. 2017. Strangers in the night—Smart process sensors in our current automation landscape. In *Multidisc. Dig. Pub. Inst. Proc.*, Vol. 1. 628.

[27] Meena Malik, Dr. Yudhvir Singh, and Anshu Arora. 2013. Analysis of LEACH protocol in wireless sensor networks. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* 3, 2 (2013).

[28] László Monostori, Botond Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda. 2016. Cyber-physical systems in manufacturing. *CIRP Ann.* 65, 2 (2016), 621–641.

[29] Sirajum Munir, Shan Lin, Enamul Hoque, S. M. Nirjon, John A. Stankovic, and Kamin Whitehouse. 2010. Addressing burstiness for reliable communication and latency bound generation in wireless sensor networks. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks.* ACM, 303–314.

[30] Marcelo Nobre, Ivanovitch Silva, and Luiz Affonso Guedes. 2015. Routing and scheduling algorithms for WirelessHARTNetworks: A survey. *Sensors* 15, 5 (2015), 9703–9740.

[31] Wolf-Bastian Pöttner, Hans Seidel, James Brown, Utz Roedig, and Lars Wolf. 2014. Constructing schedules for time-critical data delivery in wireless sensor networks. *ACM Trans. Sens. Netw.* 10, 3 (2014), 44.

[32] Marjan Radi, Behnam Dezfouli, Kamalrulnizam Abu Bakar, Shukor Abd Razak, and Malrey Lee. 2013. Network initialization in low-power wireless networks: A comprehensive study. *Comput. J.* 57, 8 (2013), 1238–1261.

[33] Tal Rusak and Philip Levis. 2008. Burstiness and scaling in low power wireless simulation. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'08).*

[34] Jianping Song, Song Han, Al Mok, Deji Chen, Mike Lucas, Mark Nixon, and Wally Pratt. 2008. WirelessHART: Applying wireless technology in real-time industrial process control. In *Proceedings of the Real-time and Embedded Technology and Applications Symposium (RTAS'08).* IEEE, 377–386.

[35] Chinchu T. Sony, C. P. Sangeetha, and C. D. Suriyakala. 2015. Multi-hop LEACH protocol with modified cluster head selection and TDMA schedule for wireless sensor networks. In *Proceedings of the Global Conference on Communication Technologies (GCCT'15).* IEEE, 539–543.

[36] Kannan Srinivasan, Maria A. Kazandjieva, Saatvik Agarwal, and Philip Levis. 2008. The β-factor: Measuring wireless link burstiness. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys'08).*

[37] Rodrigo Teles Hermeto, Antoine Gallais, and Fabrice Theoleyre. 2017. Scheduling for IEEE802. 15.4-TSCH and slow channel hopping MAC in low power industrial wireless networks. *Comput. Commun.* 114, C (2017), 84–105.

[38] Sudhanshu Tyagi and Neeraj Kumar. 2013. A systematic review on clustering and routing techniques based upon LEACH protocol for wireless sensor networks. *J. Netw. Comput. Appl.* 36, 2 (2013), 623–645.

[39] Jiafu Wan, Min Chen, Feng Xia, Li Di, and Keliang Zhou. 2013. From machine-to-machine communications towards cyber-physical systems. *Comput. Sci. Inform. Syst.* 10, 3 (2013), 1105–1128.

[40] Jianjun Wen, Zeeshan Ansar, and Waltenegus Dargie. 2015. A link quality estimation model for energy-efficient wireless sensor networks. In *Proceedings of the IEEE International Conference on Communications (ICC'15).* IEEE, 6694–6700.

[41] Jeffrey E. Wieselthier, Gam D. Nguyen, and Anthony Ephremides. 2000. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *Proceedings of the 19th Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'00)*, Vol. 2. IEEE, 585–594.

[42] Fang-Jing Wu, Yu-Fen Kao, and Yu-Chee Tseng. 2011. From wireless sensor networks towards cyber physical systems. *Perv. Mob. Comput.* 7, 4 (2011), 397–413.

[43] Fan Xiangning and Song Yulin. 2007. Improvement on LEACH protocol of wireless sensor network. In *Proceedings of the International Conference on Sensor Technologies and Applications (SensorComm'07).* IEEE, 260–264.

[44] Mao Yan, Kam-Yiu Lam, Song Han, Edward Chan, Qingchun Chen, Pingzhi Fan, Deji Chen, and Mark Nixon. 2014. Hypergraph-based data link layer scheduling for reliable packet delivery in wireless sensing and control networks with end-to-end delay constraints. *Inform. Sci.* 278 (2014), 34–55.

[45] Hao-Tsung Yang, Kin Sum Liu, Jie Gao, Shan Lin, Sirajum Munir, Kamin Whitehouse, and John Stankovic. 2017. Reliable stream scheduling with minimum latency for wireless sensor networks. In *Proceedings of the 14th IEEE International Conference on Sensing, Communication, and Networking (SECON'17).* IEEE, 1–9.

[46] Zhiwei Zhao, Wei Dong, Gonglong Chen, Geyong Min, Tao Gu, and Jiajun Bu. 2017. Embracing corruption burstiness: Fast error recovery for ZigBee under Wi-Fi interference. *IEEE Trans. Mob. Comput.* 16, 9 (2017), 2518–2530.

Received May 2018; revised May 2019; accepted June 2019