Report Number: WUCSE-2012-16

2012

Accounting for Failures in Delay Analysis for WirelessHART Networks

Authors: Abusayeed Saifullah, Paras Babu Tiwari, Bo Li, and Chenyang Lu lu@wustl.edu

WirelessHART networks are gaining ground as a real-time communication infrastructure in industrial wireless control systems. Because wireless communication is often susceptible to transmission failures in industrial environments, it is essential to account for failures in the delay analysis for realtime flows between sensors and actuators in process control. WirelessHART networks handle transmission failures through retransmissions using dedicated and shared time slots through different paths in the routing graphs. While these mechanisms for handling transmission failures are critical for process control requiring reliable communication, they introduce substantial challenges to worst-case end-to-end delay analysis for real-time flows. This paper presents the first worst-case end-to-end delay analysis for periodic real-time flows in a WirelessHART network that takes into account transmission failures. The delay bounds can be used to quickly assess the schedulability of real-time flows for industrial wireless control applications with stringent requirements on both high reliability and network latency. Simulations based on the topologies of a wireless sensor network testbed consisting of 69 TelosB motes indicate that our analysis provides safe upper bounds of the end-to-end... **Read complete abstract on page 2.**

Accounting for Failures in Delay Analysis for WirelessHART Networks

Complete Abstract:

WirelessHART networks are gaining ground as a real-time communication infrastructure in industrial wireless control systems. Because wireless communication is often susceptible to transmission failures in industrial environments, it is essential to account for failures in the delay analysis for realtime flows between sensors and actuators in process control. WirelessHART networks handle transmission failures through retransmissions using dedicated and shared time slots through different paths in the routing graphs. While these mechanisms for handling transmission failures are critical for process control requiring reliable communication, they introduce substantial challenges to worst-case end-to-end delay analysis for real-time flows. This paper presents the first worst-case end-to-end delay analysis for periodic real-time flows in a WirelessHART network that takes into account transmission failures. The delay bounds can be used to quickly assess the schedulability of real-time flows for industrial wireless control applications with stringent requirements on both high reliability and network latency. Simulations based on the topologies of a wireless sensor network testbed consisting of 69 TelosB motes indicate that our analysis provides safe upper bounds of the end-to-end delays of real-time flows at an acceptable level of pessimism.

2012-16

Accounting for Failures in Delay Analysis for WirelessHART Networks

Authors: Abusayeed Saifullah, Paras Babu Tiwari, Bo Li, Chenyang Lu, Yixin Chen

Corresponding Author: saifullah@wustl.edu

Web Page: http://www.cse.wustl.edu/~saifullaha/

Abstract: WirelessHART networks are gaining ground as a real-time communication infrastructure in industrial wireless control systems. Because wireless communication is often susceptible to transmission failures in industrial environments, it is essential to account for failures in the delay analysis for realtime flows between sensors and actuators in process control. WirelessHART networks handle transmission failures through retransmissions using dedicated and shared time slots through different paths in the routing graphs. While these mechanisms for handling transmission failures are critical for process control requiring reliable communication, they introduce substantial challenges to worst-case end-to-end delay analysis for real-time flows. This paper presents the first worst-case end-to-end delay analysis for periodic real-time flows in a WirelessHART network that takes into account transmission failures. The delay bounds can be used to quickly assess the schedulability of real-time flows for industrial wireless control applications with stringent requirements on both high reliability and network latency. Simulations based on the topologies of a wireless sensor network testbed consisting of 69 TelosB motes indicate that our analysis provides safe upper bounds of the end-to-end delays of real-time flows at an acceptable level of pessimism.

Type of Report: Other

Accounting for Failures in Delay Analysis for WirelessHART Networks

Abusayeed Saifullah, Paras Babu Tiwari, Bo Li, Chenyang Lu, Yixin Chen
Department of Computer Science and Engineering
Washington University in St. Louis
{saifullah, pbtiwari, b.li, lu, ychen25}@wustl.edu

***Abstract*—WirelessHART networks are gaining ground as a real-time communication infrastructure in industrial wireless control systems. Because wireless communication is often susceptible to transmission failures in industrial environments, it is essential to account for failures in the delay analysis for real-time flows between sensors and actuators in process control. WirelessHART networks handle transmission failures through retransmissions using dedicated and shared time slots through different paths in the routing graphs. While these mechanisms for handling transmission failures are critical for process control requiring reliable communication, they introduce substantial challenges to worst-case end-to-end delay analysis for real-time flows. This paper presents the first worst-case end-to-end delay analysis for periodic real-time flows in a WirelessHART network that takes into account transmission failures. The delay bounds can be used to quickly assess the schedulability of real-time flows for industrial wireless control applications with stringent requirements on both high reliability and network latency. Simulations based on the topologies of a wireless sensor network testbed consisting of 69 TelosB motes indicate that our analysis provides safe upper bounds of the end-to-end delays of real-time flows at an acceptable level of pessimism.**

I. Introduction

WirelessHART networks [1], [2] are gaining ground as a real-time communication infrastructure in industrial wireless control systems. Industrial process control applications demand both reliability and real-time guarantees in wireless communication [3]. Failures in wireless transmissions are prevalent in industrial environments due to channel noise, power failure, physical obstacle, multipath fading, and interference from co-existing wireless devices. WirelessHART deals with transmission failures through retransmissions, multi-path graph routing, and channel diversity [1]. While these mechanisms enable WirelessHART networks to achieve high reliability, they increase communication delay and complicate the end-to-end delay analysis for real-time flows in the network.

In this paper, we present the first end-to-end delay analysis for periodic real-time flows in a WirelessHART network that takes into account transmission failures. The analysis provides an upper bound of the communication delay of each data flow and therefore can be used to determine if the real-time flows can meet their deadlines. The delay bounds can be used to quickly assess the schedulability of real-time flows for the purpose of online admission control or workload adjustment in response to network dynamics for industrial wireless control applications with stringent requirements on both high reliability and network latency.

Specifically, WirelessHART employs the following mechanisms to recover from transmission failures. A WirelessHART network can support reliable graph routing [1], [2], [4] that provides redundant routes for real-time flows. For each flow, the network handles transmission failures by allocating a *dedicated time slot* (in which at most one transmission is scheduled to a receiver) for each en-route device starting from the source, followed by allocating a second dedicated slot on the same path for a retransmission, and then by allocating a third *shared slot* (i.e., a time slot when multiple nodes may contend to send to a common receiver) on a separate path to support another retransmission. While highly effective for achieving reliability, these mechanisms introduce significant challenges in delay analysis for WirelessHART networks. Existing delay analysis [5] for WirelessHART networks does not account for transmission failures and retransmissions, which can lead to an underestimation of the delays when these retransmission mechanisms are enabled in a WirelessHART network.

In this paper, we incorporate these reliability specifications into real-time scheduling analysis, and propose the first end-to-end delay analysis for real-time flows that takes into account transmission failures in a WirelessHART network. Our analysis computes upper bounds of the end-to-end delays of real-time flows in pseudo polynomial time. Simulation studies based on the topologies of a 69-node wireless sensor network testbed demonstrate that our analysis provides safe upper bounds of the end-to-end delays of real-time flows, and hence enables effective schedulability tests for WirelessHART networks. Furthermore, we extend the analysis to a polynomial time analysis that can be used to compute a looser delay bound within a shorter execution time.

In the rest of this paper, Section II reviews related works. Section III describes the network model and the flow model. Section IV presents the delay analysis. Section V shows how the delay bounds can be extended to a polynomial time method. Section VI presents evaluation results. Section VII offers conclusions.

II. Related Works

Real-time scheduling for wireless networks has been explored in many early [6] and recent works [7]–[17]. As explained in our previous study [5], none of these works is

suitable for WirelessHART networks. In addition, these works do not focus on delay analysis in the network.

A number of works [17]–[21] have researched delay analysis in wireless sensor networks. Most of these works concentrate to data collection at the base station through a routing tree [18], [21] and/or do not address multi-channel communication [18]–[20]. In contrast, communication in WirelessHART is based on multiple channels and reliable graph routing. Besides, our analysis is targeted for real-time flows between sensors and actuators for process control purposes, and is not limited to just data collection at one node.

Real-time scheduling for WirelessHART networks has received considerable attention in recent works [4], [5], [22]–[27]. Some of these just focus on data collection at one node and limit the applicability to simplified network models such as linear [22] and tree networks [25], [27]. The rest address more general model of the network, but mostly focus on either scheduling [23], [24], routing [4], or rate assignment algorithms [26]. The delay analysis for real-time flows between sensors and actuators has been studied only in [5]. However, it does not consider reliable real-time scheduling, and its delay bounds do not hold under communication failures.

In contrast to the results in [5], we address delay analysis under communication failures for real-time flows in process control applications. These applications are mission critical and delay sensitive, and require a high degree of reliability with real-time communication [3]. But wireless communication in industrial environments is susceptible to transmission failures due to channel noise, power failure, physical obstacle, multipath fading, and interference. To ensure reliability, communications in WirelessHART networks are scheduled based on reliable graph routing [1], [2], [4]. Transmission failures are handled through retransmissions using dedicated and shared time slots through different paths in the routing graphs. While these mechanisms are critical to mitigate wireless deficiencies in unreliable industrial environments, they add substantial challenges to delay analysis. We integrate these specifications into real-time scheduling analysis, and propose the first delay analysis that takes into account transmission failures. The delay bounds can quickly assess the schedulability of real-time flows for industrial control applications having stringent requirements on both high reliability and network latency.

III. System Model

A. WirelessHART Architecture

The WirelessHART standard [1] is built upon very conservative design rules that ensure reliable communication despite wireless deficiencies in unreliable industrial environments. Characterized by a centralized *Network Manager*, WirelessHART forms a mesh network consisting of a Gateway, a set of field devices, and several access points. The network manager and the controllers are installed in the *Gateway*. The *field devices* are wirelessly networked sensors and actuators. *Access Points* are wired to the Gateway to provide redundant paths between the wireless network and the Gateway. For process control, the sensor devices deliver sample data to the controllers, and the control messages are then delivered to the actuators. The network manager schedules their transmissions, and distributes the schedule among devices.

Transmissions are scheduled based on TDMA. A transmission and its acknowledgement requires one time slot. For transmission between a receiver and its senders, a time slot can be either dedicated or shared. In a *dedicated slot*, only one sender is allowed to transmit to the receiver. In a *shared slot*, more than one sender can attempt to transmit to the same receiver. The network uses 16 channels defined in IEEE $802.15.4$, and adopts channel hopping in every time slot. Each receiver uses a distinct channel for reception in any time slot. Any excessively noisy channel is blacklisted not to be used.

To offset transmission failures, communications are scheduled based on reliable graph routing. A *routing graph* is a directed list of paths that connect two devices. Packets from all sensor nodes are routed to the Gateway through the *uplink graph*. For every actuator, there is a *downlink graph* from the Gateway through which the Gateway delivers the control messages. The end-to-end communication between a source (sensor) and destination (actuator) pair happens in two phases. In the sensing phase, on one path from the source to the Gateway in the uplink graph the scheduler allocates a dedicated slot for each en-route device starting from the source, followed by allocating a second dedicated slot on the same path (starting from the source) to handle a retransmission. The links on this path are called *dedicated links*. Then, to offset failure of both transmissions along a primary link, the scheduler again allocates a third shared slot on a separate path to handle another retry. The links on these paths are called *shared links*. Then, in the *control phase*, using the same way, the dedicated links and shared links are scheduled in the downlink graph of the destination.

B. Flow Model

A periodic end-to-end communication between a source (sensor) and a destination (actuator) is called a *flow*. We consider there are n flows: $F_1, F_2, \cdots, F_n$ for process monitoring and control purposes. The source and the destination of F_i are denoted by s_i and d_i, respectively. Each flow $F_i,\ 1 \leq i \leq n$, has a fixed priority. Transmissions of a flow are scheduled based on the priority of the flow. We assume that flows are ordered by priorities. Flow F_h has higher priority than flow F_i if and only if $h < i$.

The subgraph of the uplink graph that s_i (the source of F_i) uses to deliver sensor data to the Gateway is denoted by UG_i. The downlink graph for the destination of F_i is denoted by DG_i. The graph consisting of UG_i and DG_i is the *routing graph* of F_i, and is denoted by G_i. Associated with each flow F_i are a period T_i and a deadline D_i, where $D_i \leq T_i$. Time slots are used as time units. The sensor periodically generates data at a period of T_i which has to be delivered to the Gateway in the sensing phase, and then control message has to be delivered to the actuator in the control phase. The total communication delay in two phases, called the *end-to-end delay*, must not exceed D_i. Our objective is to determine

an upper bound R_i of the end-to-end delay of each flow F_i. The end-to-end delay analysis will determine the flows to be *schedulable* if $R_i \leq D_i, \ \forall 1 \leq i \leq n$.

C. Fixed Priority Scheduling

We consider that the flows are scheduled based on fixed priority scheduling policy. In a *fixed priority scheduling policy*, each flow has a fixed priority, and its transmissions are scheduled based on this priority. Starting from the highest priority flow F_1, the following procedure is repeated for every flow F_i in decreasing order of priority. For current priority flow F_i, the network manager schedules its dedicated links and shared links on UG_i in its sensing phase on earliest available time slots and on available channels. It then schedules the dedicated links and shared links on DG_i in the control phase following the same way. F_i is scheduled up to the hyper-period of flows $F_1, \cdots, F_i$, after which its schedule repeats.

A transmission is *conflicting* with a transmission of a higher priority flow, if they involve a common node. Hence a time slot is *available* if no conflicting transmission is already scheduled in that slot except the case that transmissions along the shared links having the same receiver are allowed to schedule in the same slot. Each receiver uses a separate channel for reception in any time slot. We use m to denote the number of available channels. For any flow F_i there are multiple paths both in UG_i, and in DG_i. But only one path will be chosen by its packet based on link condition (further detailed in Subsection IV-A).

The complete schedule is split into superframes. A *superframe* represents transmissions in a series of time slots that repeat infinitely and represent the communication pattern of a group of devices. The schedule can be mapped to superframes as follows. For any i and j such that $1 \leq i \leq j \leq n$, the schedule for flows $F_1, F_2, \cdots, F_i$ is repeated after their hyper-period. Therefore, the schedule for flows $F_1, F_2, \cdots, F_i$ can be assigned to a superframe of length (i.e., total time slots in the superframe) equal to their hyper-period. Similarly, the schedule for flows $F_i, \ F_{i+1}, \cdots, F_j$ is repeated after the hyper-period of first j flows (i.e., flows $F_1, \ F_2, \cdots, F_j$), and hence can be assigned to a superframe of length equal to that hyper-period. For example, when $D_i = T_i$ for each F_i, flows having the same period are assigned in the superframe of length equal to that period under rate monotonic scheduling.

IV. Delay Analysis under Transmission Failures

We now present an efficient end-to-end delay analysis under communication failures for real-time flows that are prioritized by a given fixed priority policy. Due to its simplicity and reduced overhead, fixed-priority scheduling is a commonly adopted policy in practice for real-time CPU scheduling, control-area networks, and also for WirelessHART networks. Our analysis determines an upper bound of the end-to-end delay of every flow considering the worst-case successful end-to-end communication. In other words, we bound the delay considering that at least one path in the routing graph survives through which the packet can be delivered to the destination.

In the scheduling, a lower priority flow can be delayed by higher priority flows (a) due to *channel contention* (when all channels are assigned to transmissions of higher priority flows in a time slot), and (b) due to *transmission conflicts* (when a transmission of the flow and a transmission of a higher priority flow involve a common node). Like [5], we first analyze each delay separately. We then incorporate both types of delays into our analysis and end up with an upper bound of the end-to-end delay for every flow. All notations used in the analysis are summarized in Table I.

m	total number of available channels in the network
n	total number of flows
F_i	a flow with priority i
s_i	source (sensor) of F_i
d_i	destination (actuator) of F_i
T_i	period of F_i
D_i	deadline of F_i
R_i	an upper bound of end-to-end delay of F_i
UG_i	subgraph of the uplink graph used by F_i
DG_i	downlink graph of F_i
G_i	routing graph of F_i (consists of UG_i and DG_i)
L_i^{sen}	worst-case time requirement of F_i in sensing phase
L_i^{con}	worst-case time requirement of F_i in control phase
L_i	worst-case time requirement of F_i in 2 phases, i.e., $L_i^{\text{sen}} + L_i^{\text{con}}$
$\Omega_i(x)$	channel contention delay suffered by F_i in an interval of x slots
$\lambda_i^{h,\text{sen}}$	maximum conflict delay caused by one instance of F_h along the bottleneck sensing path of F_i
$\lambda_i^{h,\text{con}}$	maximum conflict delay caused by one instance of F_h along the bottleneck control path of F_i
γ_i^h	maximum conflict delay caused by one instance of F_h along the bottleneck link of F_i
δ_i^h	maximum conflict delay caused on F_i by the first dedicated link of F_h
Δ_i^h	maximum conflict delay that one instance of a higher priority flow F_h can cause on the bottleneck path of F_i

TABLE I
NOTATIONS

A. Channel Contention Delay

We will first determine the channel contention delay caused by one higher priority flow F_h to a lower priority one F_i. Note that here we will determine only channel contention delay (while the delay due to transmission conflict is analyzed in the next subsection). Hence, the term 'delay' used in this subsection will refer to 'only channel contention delay'.

To determine the channel contention delay in the existing delay analysis [5] that does not account for transmission failures in WirelessHART networks, the real-time scheduling problem in a WirelessHART network was mapped to global multiprocessor scheduling where each 'channel' was mapped to a 'processor', and each real-time 'flow' was mapped to a multiprocessor 'task'. Then a state-of-the-art response time analysis [28] for multiprocessor scheduling was adopted to determine the channel contention delay experienced by a flow. Specifically, in the flow model considered in [5], a 'flow' was a chain of transmissions through a single route (i.e., one path from the source to destination), and hence was mapped to a 'sequential multiprocessor task'. Since each transmission needs one time unit (i.e., one time slot), for each flow F_h the total number of transmissions along its single route (i.e., the total number of links on that route) was considered as its

(a) UG_h for F_h (b) UG_i and UG_h

Fig. 1. Routing in the sensing phase of F_i and F_h

'worst-case time requirement' C_h in the mapping. Since every transmission in a time slot happens on a separate channel, each 'channel' was mapped to a 'processor'.

Unlike that in [5], in this work a flow is *not* scheduled simply as a sequential multiprocessor task. Figure 1(a) shows UG_h (the subgraph of the uplink graph used by F_h) for flow F_h. In the figure, the dedicated links used by F_h in the sensing phase are shown in solid lines while the dotted lines indicate the shared links used by F_h. Considering that F_h is not delayed by any other flow, the time slots in which a link is activated are shown beside the links (starting from slot 1). First, starting from its source node s_h, the dedicated links $s_h \rightarrow u$, $u \rightarrow v$, and $v \rightarrow a$ are scheduled on dedicated slots 1, 2, and 3, respectively. Hence, if no transmission fails then the packet will reach the access point a in 3 time slots. To handle transmission failure, each of these links are scheduled on a second dedicated slot again (i.e., on slots 2, 3, and 4 for $s_h \rightarrow u$, $u \rightarrow v$, and $v \rightarrow a$, respectively). Then, to handle failure of both transmissions along a link, for every link the scheduler again allocates a third shared slot on a separate path to handle another retry (i.e., on slots 3, 4, and 5 along shared links $s_h \rightarrow y$, $u \rightarrow x$, and $v \rightarrow w$, respectively). For example, if the transmission along link $s_h \rightarrow u$ fails both at slot 1 and at slot 2, then the packet is delivered through the shared links along path $s_h \rightarrow y \rightarrow z \rightarrow w \rightarrow a$ that takes 6 time slots.

Thus any flow F_h is scheduled on multiple paths in UG_h. But only one route will be chosen by its packet based on link conditions. Considering no delay from higher priority flows, the *worst-case time requirement* of F_h in the **sen**sing phase is denoted by L_h^{sen}. In other words, considering no delay from higher priority flows, L_h^{sen} is the maximum number of time slots required by F_h in the sensing phase when at least one path in UG_h survives. For example, in Figure 1(a), $L_h^{\text{sen}} = 6$ slots. A similar scheduling model is followed in the control phase also. Similarly, considering no delay from higher priority flows the *worst-case time requirement* of F_h in the **con**trol phase is denoted by L_h^{con}. Thus, considering no delay from higher priority flows, the *worst-case time requirement*, denoted by L_i, of any flow F_i in two phases is

$$L_i = L_i^{\text{sen}} + L_i^{\text{con}}$$

To determine channel contention delay in the above model of flow scheduling, we make some important observations as noted below. To do so, we consider the higher priority flow F_h and the lower priority flow F_i together in the network. Hence, in Figure 1(b), with flow F_h, we also show UG_i for flow F_i. The figure shows links $s_i \rightarrow z$, $z \rightarrow v$, and $v \rightarrow a$ as dedicated links in UG_i while the corresponding shared links are $s_i \rightarrow y$, $z \rightarrow w$, and $v \rightarrow w$, respectively.

First, when F_i and F_h contend for a channel access, F_i is delayed. Note that F_h is scheduled in at most L_h time slots during which some links of F_i and F_h may be scheduled on shared slots using the same channel when the two links involve the same receiver. Hence, such links do not cause any delay. For example, in Figure 1(b), the shared link $s_i \rightarrow y$ of F_i and the shared link $s_h \rightarrow y$ of F_h have the same receiver (i.e., y) and they can be scheduled at the same time slot using the same channel[1]. Except those along shared links (having the same receiver), each other transmission in the same slot is scheduled on a separate channel. That is, between F_i and F_h there is no channel contention for the scenario when their shared links have the same receiver.

Second, as can be seen in Figure 1, more than one link of F_h may be scheduled (in parallel) at the same time. For example, links $u \rightarrow v$ and $v \rightarrow a$ are scheduled for time slot 3. While each transmission needs to be scheduled in a separate channel, such transmissions on the same slot do not need to be scheduled using separate channels since they belong to the same packet and only one of them will happen in real scheduling. For example, although links $u \rightarrow v$ and $v \rightarrow a$ are scheduled for time slot 3 for F_h (Figure 1), only one of these two schedules will happen. Hence they are assigned the same channel at slot 3. Similarly, although links $z \rightarrow w$ and link $x \rightarrow a$ are scheduled for time slot 5 for F_h, only one of these two schedules will happen. Hence they are assigned the same channel at slot 5. In each other time slot also F_h needs at most one channel.

Based on the above observations, the worst-case time re-

[1]Note that collisions may occur in a shared slot when both flows' primary links fail. This is an worst-case scenario where a packet may not be finally delivered to the destination.

quirement of F_h is L_h slots, and it needs at most one channel in every time slot. Hence, in the delay expression derived in [5] we can simply replace 'worst-case time requirements' C_h and C_i with L_h and L_i, respectively, to find the channel contention delay caused by F_h on F_i. According to [5], in any time interval of x slots, there are at most $m-1$ higher priority flows each flow F_h among which can cause at most $I_i^h(x)$ delay on F_i as expressed below

$$I_i^h(x) = \min\left(x - L_i + 1,\ \left\lfloor \frac{x - L_h}{T_h} \right\rfloor L_h + L_h + \min\left(L_h - 1, \max\left((x - L_h) \bmod T_h - (T_h - R_h), 0\right)\right)\right)$$

where R_h is the worst-case end-to-end delay of F_h. Each other higher priority flow F_h can cause at most $J_i^h(x)$ delay on F_i

$$J_i^h(x) = \min\left(x - L_i + 1,\ \left\lfloor \frac{x}{T_h} \right\rfloor L_h + \min\left(x \bmod T_h,\ L_h\right)\right)$$

Thus, considering a total of m channels, an upper bound $\Omega_i(x)$ of the channel contention delay caused by all higher priority flows on F_i in any time interval of x slots is derived as follows.

$$\Omega_i(x) = \left\lfloor \frac{1}{m}\left(Z_i(x) + \sum_{h<i} J_i^h(x)\right)\right\rfloor \tag{1}$$

with $Z_i(x)$ being the sum of the $\min(i-1, m-1)$ largest values of the differences $I_i^h(x) - J_i^h(x)$ among the higher priority flows $F_h,\ h < i$.

Effect of Channel Hopping. To every transmission, the scheduler assigns a channel offset between 0 and $m-1$ instead of an actual channel, where m is the total number of channels. Any channel offset c (i.e., $1,\ 2,\ \cdots, m-1$) is mapped to different channels at different time slots t as follows.

$$channel = (c + t) \bmod m$$

That is, although the physical channels used along a link changes (hops) in every time slot, the total number m of available channels is fixed. The scheduler only assigns a fixed channel index to a transmission which maps to different physical channels in different time slots, keeping the total number of available channels at m always, and scheduling each flow on at most one channel at any time. Hence, channel hopping does not have effect on channel contention delay.

B. Transmission Conflict Delay

Now we analyze the delay that a flow can experience due to transmission conflicts. Whenever two transmissions conflict, the transmission that belongs to the lower priority flow must be delayed, no matter how many channels are available. Note that here we will determine the delay only due to transmission conflict. Hence, the term 'delay' used in this subsection will refer to 'only transmission conflict delay'.

First we determine the conflict delay that one higher priority flow F_h may cause on a lower priority flow F_i. A transmission of F_h along a link ℓ_h and a transmission of F_i along a link ℓ_i may be conflicting in 4 ways as follows when these two links involve a common node:

1) *Type 1:* ℓ_h is a dedicated link and ℓ_i is a shared or dedicated link.
2) *Type 2:* ℓ_h is a shared link and ℓ_i is a dedicated link.
3) *Type 3:* ℓ_h is a shared link and ℓ_i is a shared link, and the receiver nodes of the two links are different.
4) *Type 4:* ℓ_h is a shared link and ℓ_i is a shared link, and the receiver nodes of the two links are the same. In this case, the transmission of F_i is not delayed.

In the first 3 cases the transmission of F_i is delayed while for Type 4 conflict it will not be delayed. Therefore, the total delay caused by F_h on F_i depends on how their dedicated and shared links intersect in the routing graphs. Now we will first upperbound the conflict delay that one instance of a higher priority flow F_h may cause on F_i. To determine this, in the next discussion we limit our attention only to F_h and F_i.

In the routing graph G_i (consisting of UG_i and DG_i) of flow F_i there can be too many directed end-to-end paths from its source s_i to destination d_i. Among these end-to-end paths, the one who experiences the maximum conflict delay from F_h is called the *bottleneck path* with respect to F_h. Although the packet is scheduled on each of these end-to-end directed paths, only one path will be chosen by the packet based on link conditions (as already explained in the previous subsection). Hence, the conflict delay caused by F_h along F_i's bottleneck path represents the upper bound of the conflict delay that F_h may cause on F_i. Let Δ_i^h be an *upper bound of conflict delay* that one instance of F_h may cause along the bottleneck path of F_i. To determine Δ_i^h, we *do not* need to find the bottleneck path or do not need to enumerate all end-to-end paths in G_i (as there can be too many end-to-end paths in G_i). Instead, we find several bounds that can be efficiently calculated and their minimum value represents a value of Δ_i^h.

First, since F_h is scheduled on all paths in G_h, its schedules along all of these paths may cause delay on the bottleneck path of F_i. But F_h will finish in L_h time slots. Hence, Δ_i^h cannot exceed L_h. Note that this upper bound of Δ_i^h may not be very loose in most cases since the value of L_h is very small compared to the total number of transmissions scheduled for F_h (as can be seen for the sensing phase in Figure 1). Besides, when the (outgoing) links of s_h (source of F_h) in G_h do not have any node in G_i, this upper bound can be further decreased since these links will never cause conflict delay on F_i. The same holds for the (incoming) links of d_h (destination of F_h) in G_h. Considering β_i^h as the number of nodes in the set $\{s_h, d_h\}$ whose incident links in G_h do not have any node that is also in G_i, Lemma 1 provides a bound of Δ_i^h.

Lemma 1: For a higher priority flow F_h and a lower priority flow F_i, $\Delta_i^h \le L_h - \beta_i^h$.

Proof: When there are only two flows F_h and F_i, F_h completes in L_h time slots in which F_h must not conflict with F_i for at least β_i^h time slots. Hence, F_h conflicts with F_i for at most $L_h - \beta_i^h$ time slots. ■

Second, let us call the bottleneck path (with respect to F_h) in UG_i the *bottleneck sensing path* of F_i. Let an upper bound

of conflict delay caused by F_h on F_i's bottleneck **sen**sing path be $\lambda_i^{h,\text{sen}}$. A value of $\lambda_i^{h,\text{sen}}$ can be efficiently calculated without enumerating all paths in UG_i as explained below. Let us consider a particular path p in UG_i. The total number of transmissions of (one instance of) F_h that may have Type 1, 2, or 3 conflict on p represents a value of conflict delay along p caused by one instance of F_h. For example, in Figure 1(b), F_h has 9 transmissions that may cause delay along $p = s_i \to z \to w \to a$ of F_i. (Note that this is the delay along p considering links $s_i \to z$, $z \to v$, $z \to w$, and $w \to a$ of F_i. Link $z \to v$ is considered because $z \to w$ is scheduled only after scheduling $z \to v$.) Now the path in UG_i whose delay (calculated using the above method) is maximum is the bottleneck sensing path, and its delay represents $\lambda_i^{h,\text{sen}}$. Such a value of $\lambda_i^{h,\text{sen}}$ can be found quickly by exploring each link on UG_i once, for example, using a depth-first search on UG_i.

Similarly, let $\lambda_i^{h,\text{con}}$ be the conflict delay along the bottleneck **con**trol path. The value of $\lambda_i^{h,\text{con}}$ also can be calculated using the way explained above. Now based on these values, Lemma 2 provides another bound of Δ_i^h. Theorem 3 then finally establishes the bound Δ_i^h.

Lemma 2: For a higher priority flow F_h and a lower priority flow F_i, $\Delta_i^h \le \lambda_i^{h,\text{sen}} + \lambda_i^{h,\text{con}}$.

Proof: Since the control phase of F_i starts after its sensing phase is complete, the bottleneck path between s_i and d_i consists of its bottleneck sensing path and the bottleneck control path. Hence, $\lambda_i^{h,\text{sen}} + \lambda_i^{h,\text{con}}$ is an upper bound of conflict delay caused by one instance of F_h along F_i's bottleneck path. ■

Theorem 3: For a higher priority flow F_h and a lower priority flow F_i,

$$\Delta_i^h = \min\left(\lambda_i^{h,\text{sen}} + \lambda_i^{h,\text{con}},\ L_h - \beta_i^h\right) \tag{2}$$

Proof: Follows from Lemma 1 and Lemma 2. ■

Now we consider a special case where Δ_i^h can be further decreased. If a directed path in G_h and a directed path in G_i overlap, then on that overlap F_i can be delayed at most once by the transmissions of F_h on that overlap [5]. Note that in most cases DG_i and DG_h may overlap. Similarly, UG_i and UG_h may overlap. In the overlapping subgraph of G_i and G_h, the routing of F_i and F_h are the same. In such cases, if G_i and G_h do not intersect anywhere outside the overlapping subgraph, then along a path p in G_i, F_i can be delayed by F_h at most once. That is, on p once F_i is delayed along a link by F_h, after the delay causing transmissions are scheduled, both F_i and F_h can be scheduled along p in parallel (see [5] for details). Let the link on G_i that may have maximum conflict delay of Type 1, 2, or 3 with F_h be called the *bottleneck link* of F_i (with respect to F_h). That is, a transmission of F_i along this link may face the highest conflict with F_h. Let γ_i^h denote the *maximum conflict delay* along the bottleneck link. (For example, considering only UG_i in Figure 1(b), we can see that $\gamma_i^h = 7$, since a link of F_i can have conflict with at most 7 transmissions of F_h. Here, $z \to v$ is F_i's bottleneck link.) Then for the above routing structure, we can update Δ_i^h further as follows

$$\Delta_i^h = \min\left(\lambda_i^{h,\text{sen}} + \lambda_i^{h,\text{con}},\ L_h - \beta_i^h,\ \gamma_i^h\right)$$

So far we have derived Δ_i^h, an upper bound of delay that one instance of F_h can cause along F_i's bottleneck path. Now we will upperbound the total delay caused by all instances of F_h. In considering the delay caused by multiple instances, we observe that at the time when a transmission on a directed path p in G_i conflicts with some transmission of F_h, the preceding transmissions on p are already scheduled. These already scheduled transmissions on p are no more subject to delay by the subsequent instances of F_h. For example, in Figure 1(b) let us consider the path $s_i \to y \to z \to v \to w \to a$ in UG_i of F_i. If some instance of F_h conflicts and causes delay on F_i's transmission along $v \to w$, the next instance of F_h must not delay F_i's transmissions along links $s_i \to y$, $y \to z$, and $z \to v$ on this path since these transmissions are already scheduled. Thus only the transmissions that are not yet scheduled along path p will be considered for conflict delay by the subsequent instances of F_h. These observations lead to Lemma 4 and Lemma 5.

Lemma 4: Let $k > 1$ instances of F_h cause delay on any directed path p in G_i such that there is no common link on p along which F_i's transmission is delayed by more than 1 instance of F_h. Then the total delay caused along p by these k instances is at most Δ_i^h.

Proof: Let the set of links of G_h along which F_h's transmissions can conflict and cause delay on F_i along path p be denoted by Q. Note that the total delay caused by these links on F_i's transmissions along p is at most Δ_i^h. When one instance $F_{h,1}$ of F_h causes delay along p, its transmissions along a subset Q_1 of links Q cause delay on p. Now consider a second instance $F_{h,2}$ of F_h. Transmissions of $F_{h,2}$ along another subset Q_2 of Q cause delay on p. Since there is no link of p along which F_i's transmission is delayed by both $F_{h,1}$ and $F_{h,2}$, the subsets Q_1 and Q_2 must be disjoint. Similarly, for any k, these subsets are disjoint. Hence, the total delay on p caused by k, $k \ge 2$, instances of F_h cannot exceed Δ_i^h. ■

Let δ_i^h denote the *maximum conflict delay* caused on F_i by the first dedicated link of F_h. Note the *first dedicated link* of F_h is the dedicated link incident on s_i in UG_i. For example, in Figure 1(b), $s_h \to u$ is the first dedicated link of F_h. Specifically, if u lies on G_i, then $\delta_i^h = 4$ (since node u involves 4 time slots as can be seen in Figure 1(b)). If s_h lies on G_i but u does not, then $\delta_i^h = 3$. If none of of these two nodes lies on G_i, then $\delta_i^h = 0$. The value of δ_i^h plays a critical role in determining the total delay caused along a path by all instances of F_h as shown in Lemma 5.

Lemma 5: (i) If G_i does not share a node with the first dedicated link of F_h, then along each link on any path p in G_i at most one instance of F_h can cause delay.
(ii) Let some link ℓ^* of G_i share a node with F_h's first dedicated link. F_i's transmission along link ℓ^* can be delayed by at most two instances of F_h. For other links that do not share node with F_h's first dedicated link, along each link on any path p in G_i at most one instance of F_h can cause delay.

Proof: (i) Suppose to the contrary, along a link ℓ on p two instances of F_h can cause delay. Let these two instances of F_h be $F_{h,1}$ and $F_{h,2}$, where $F_{h,1}$ is released before $F_{h,2}$. During the time when $F_{h,1}$ causes delay on a transmission along ℓ, the delay causing transmissions are scheduled. Immediately after scheduling them, if $F_{h,2}$ is not released then the delayed transmission along ℓ can be scheduled. Otherwise, if $F_{h,2}$ is released immediately then its first dedicated slot will be scheduled. Since G_i does not intersect with this dedicated link, it cannot delay the transmission under consideration along ℓ. Thus it contradicts with our hypothesis.
(ii) This proof follows from the above proof for (i). Specifically, the above proof says that two instances $F_{h,1}$ and $F_{h,2}$ can cause delay along link ℓ^*. Now the fact that no third instance of F_h can delay along ℓ^* immediately follows from the above proof for (i). Similarly, follows from (i) that, for other links of G_i that do not share node with F_h's first dedicated link, each link on p can be delayed by at most one instance of F_h. ■

Theorem 6 now establishes an upper bound of conflict delay caused by all instances of F_h on F_i.

Theorem 6: The worst-case delay caused by a higher priority flow F_h on a lower priority F_i due to transmission conflict is upper bounded by

$$\Delta_i^h + \min\left(\left\lceil \frac{D_i}{T_h} \right\rceil - 1,\ 1\right)\delta_i^h$$

Proof: By Lemma 4 and Lemma 5, for any path p in G_i, if no common link on p is delayed by more than one instance of F_h, then the total delay caused along p is at most Δ_i^h. Now let there be a common link on p that is delayed by more than one instance. By Lemma 4, in this case at most one more instance can cause δ_i^h additional delay, no matter how many instances of F_h are released until deadline D_i is over. Hence, this additional delay is at most $\min(\lceil \frac{D_i}{T_h} \rceil - 1,\ 1)\delta_i^h$. That is, along any path p in G_i, the delay caused by F_h is at most $\Delta_i^h + \min(\lceil \frac{D_i}{T_h} \rceil - 1,\ 1)\delta_i^h$. Since this bound is true for any path in G_i, it is true for the bottleneck path in G_i. Since the conflict delay along the bottleneck path represents the conflict delay caused on F_i by F_h, the theorem follows. ■

From Theorem 6, now an upper bound of the total delay caused on flow F_i by all higher priority flows due to transmission conflicts is

$$\sum_{h<i}\left(\Delta_i^h + \min\left(\left\lceil \frac{D_i}{T_h} \right\rceil - 1,\ 1\right)\delta_i^h\right) \tag{3}$$

C. End-to-End Delay Bound

Now both types of delays are incorporated together to develop an upper bound of the end-to-end delay of every flow using the approach used in [5]. This is done for every flow in decreasing order of priority starting with the highest priority flow. Theorem 7 provides an upper bound R_i of end-to-end delay for every flow F_i.

Theorem 7: Let x_i^* be the minimum value of $x \geq L_i$ that solves Equation 4 using a fixed-point algorithm.

$$x = \Omega_i(x) + L_i \tag{4}$$

Then the end-to-end delay bound R_i of flow F_i is

$$R_i = x_i^* + \sum_{h<i}\left(\Delta_i^h + \min\left(\left\lceil \frac{D_i}{T_h} \right\rceil - 1,\ 1\right)\delta_i^h\right) \tag{5}$$

Proof: Note that according to Equation 1 x_i^* is calculated considering R_h (i.e., the end-to-end delay bound of F_h considering both channel contention delay and conflict delay) of each higher priority flow F_h. According to Equation 1, $\Omega_i(x)$ is the channel contention delay caused by all higher priority flows on F_i in any time interval of x slots. Hence x_i^* is the bound of the end-to-end delay of F_i when it suffers only from channel contention delay caused by higher priority flows (and no conflict delay). Equation 3 provides the bound of transmission conflict delay of F_i. Hence, adding this value to x_i^* must be an upper bound of F_i's end-to-end delay considering both channel contention delay and transmission conflict delay. ■

Thus we can determine R_i for every flow F_i in decreasing order of priority starting with the highest priority flow using Theorem 7. In solving Equation 4, if x exceeds D_i, then F_i is decided to be "unschedulable". Similarly, if $R_i > D_i$, then F_i is unschedulable. Since determining the minimum value of x to solve Equation 4 takes pseudo polynomial time, R_i can be calculated in pseudo polynomial time for every F_i.

V. Delay Analysis in Polynomial Time

Here we extend the analysis to compute the delay bounds in polynomial time. This bound is not as precise as the bound that is computed in pseudo polynomial time in the previous section, but is often preferred due to its faster execution time.

Using the same mapping proposed in [5] of transmission scheduling in a WirelessHART network to the global multiprocessor scheduling, we can follow the same approach used in Subsection IV-A to determine channel contention delay based on the response time analysis for global multiprocessor scheduling proposed in [29]. The response time bound (in multiprocessor scheduling) using this analysis can be computed in polynomial time. According to Subsection IV-A, any flow F_i needs at most one channel in every time slot, and its worst-case time requirement is L_i. Hence, similar to Subsection IV-A, we can replace 'worst-case execution requirement' of a task with 'worst-case time requirement' L_i for each flow F_i in the said response time analysis to calculate the channel contention delays of the flows. In particular, using this analysis, the maximum channel contention delay, denoted by Ω_i^h, that a flow F_i can experience during its lifetime from a higher priority flow F_h can be expressed as follows.

$$\begin{aligned}\Omega_i^h = &\min\Big(D_i - L_i + 1, \left\lfloor \frac{D_i + D_h - L_h}{T_h} \right\rfloor .L_h \\ &+ \min\left(L_h, D_i + D_h - L_h - \left\lfloor \frac{D_i + D_h - L_h}{T_h} \right\rfloor .T_h\right)\Big)\end{aligned}$$

The maximum channel contention delay caused by all higher priority flows, denoted by Ω_i, is as follows.

$$\Omega_i = \left\lfloor \frac{1}{m}\sum_{h<i}\Omega_i^h \right\rfloor \tag{6}$$

Now using Equations 3, 5, and 6, the end-to-end delay R_i of F_i is determined as follows.

$$R_i = L_i + \Omega_i + \sum_{h<i} \left(\Delta_i^h + \min \left(\left\lceil \frac{D_i}{T_h} \right\rceil - 1,\ 1 \right) \delta_i^h \right) \quad (7)$$

Each R_i thus can be calculated in $O(n)$ time when all Δ and δ values are known.

VI. Evaluation

We evaluate the proposed end-to-end delay analysis through simulations based on the topologies of a wireless sensor network testbed deployed in two buildings (Bryan Hall and Jolley Hall) of Washington University in St Louis [30]. The testbed[2] consists of 69 TelosB motes each equipped with Chipcon CC2420 radios which are compliant with the IEEE 802.15.4 standard. Note that the physical layer of WirelessHART is also based on IEEE 802.15.4. We collect the topologies for 3 different channels: 26, 20, and 15. In every case, the transmission power of each node is set to 0 dBm. Setting the same channel at every node, every node broadcasts 50 packets in a round-robin fashion. The neighbors record the sequence numbers of the packets they receive. This cycle is repeated for 10 rounds. Then every link with a higher than 60% PRR (packet reception rate) is considered reliable and considered as a link in the topology. For Channel 26, the topology is shown in Figure 2 (embedded on the floor plan of two buildings).

We consider two fixed priority assignment policies to demonstrate the performance of the analysis. The first one is Deadline Monotonic (DM), a widely used fixed priority scheduling policy in CPU scheduling and in control area networks. DM assigns priorities to flows according to their relative deadlines; the flow with the shortest deadline being assigned the highest priority. The second one is a Delay-Based (DB) algorithm devised by modifying Audsley's [31] schedulability test based priority assignment algorithm. In DB, we use the delay bound derived in Equation 7 that can be computed for each flow in polynomial time and without considering different priority ordering among its higher priority flows. In particular, DB first prioritizes the flows according to DM. Then starting from the lowest priority level n, at each level k we check if the current flow F_k can meet the deadline at this level. If it can, then we go to the next level. Otherwise, starting from level 1 to $k-1$, the first higher priority flow F_h (currently at level $h \leq k-1$) who can meet deadline at this lower level k (considering F_k in its higher priority list) is assigned at level k. Then we increase the priority by one level of each flow from level $h+1$ to $k-2$, and F_k who cannot meet deadline at level k is put at level $k-1$. If no such F_h is found, then we simply go to next priority level, and repeat the same procedure.

[2]Our testbed in fact consists of 79 TelosB motes while, in this evaluation, we use only 69 nodes as the remaining nodes were under maintenance during our evaluation.

A. Simulation Setup

In the testbed topologies, we generate flows by randomly selecting sources and destinations. In every case, two nodes in the topology are selected as access points. The uplink and downlink graphs are generated using the algorithms presented in [4]. The periods of the flows are considered harmonic and are randomly generated in the range $2^{5\sim12}$ time slots. The deadlines are considered equal to periods. In all cases, we use 12 channels for scheduling.

B. Performance Analysis

We evaluate our analysis in terms of the following metrics. (a) *Acceptance ratio:* This is defined as the proportion of the number of test cases deemed to be schedulable to the total number of test cases. (b) *Pessimism ratio:* For a flow, this metric is defined as the proportion of the analyzed theoretical upper bound to its maximum end-to-end delay observed in simulations. Since there exists no prior work on reliability integrated delay analysis, we analyze the effectiveness of our delay analysis by simulating the complete schedule of transmissions of all flows released within the hyper-period. The results are shown for our analysis where delay bounds are computed in pseudo polynomial time.

Figure 3 shows the acceptance ratios for DM and DB priority assignment algorithms of 1000 test cases under varying number of flows. Every test case is simulated by scheduling all the instances of the flows released within their hyper-period. In the figure, "DB-Simulation" and "DM-Simulation" indicate the fraction of test cases that have no deadline misses in the simulations under DB and DM, respectively. "DB-Acceptance" and "DM-Acceptance" indicate the acceptance ratio of our delay analysis when flows are prioritized under DB and DM, respectively. The observations in these tests are noted below.

Figure 3(a) shows the results for the topology collected under Channel 26. In this topology, for 10 flows 988 test cases among 1000 are schedulable through simulations when priorities are assigned based on DB. Our analysis has determined 839 cases as schedulable. That is, almost 85% of the schedulable cases were deemed schedulable. When priorities are assigned based on DM, 988 test cases among 1000 are schedulable through simulations while our analysis has determined 821 cases as schedulable showing its acceptance ratio at 0.82. Similarly, for 20 flows 963 cases are schedulable through simulations under DB. Our analysis has determined 617 cases as schedulable which is almost 64% of the total schedulable cases. Under DM, 962 cases are schedulable through simulations among which 593 cases as deemed schedulable by our analysis, which is almost 62% of the total schedulable cases.

Figure 3(b) shows the results for the topology collected under Channel 20. In this topology, for 10 flows 916 test cases among 1000 are schedulable through simulations when priorities are assigned based on DB. Our analysis has determined 650 cases as schedulable. That is almost 71% of the schedulable cases were deemed schedulable by our analysis. For DM, 915 test cases are schedulable through simulations

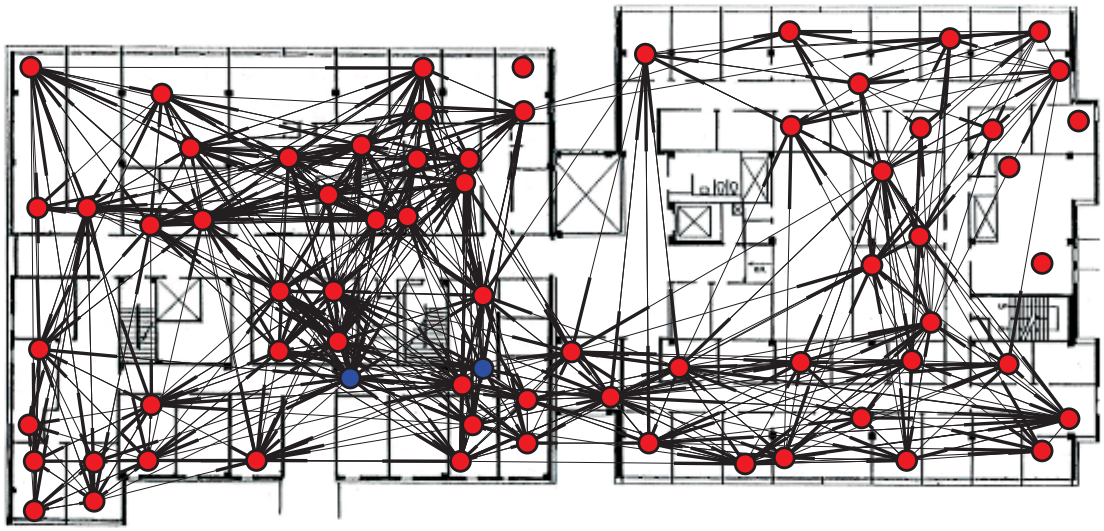
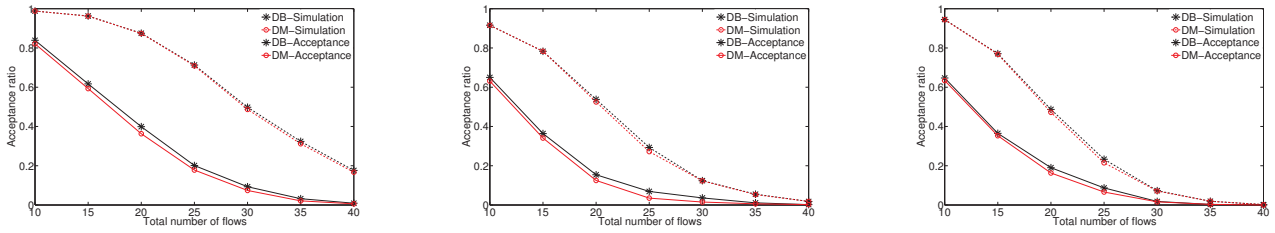
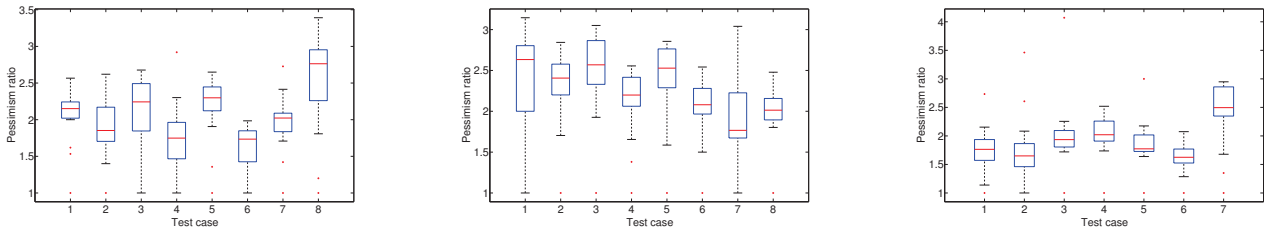


Fig. 2. Testbed topology using Channel 26 (access points are colored in blue)



(a) Topopology using Channel 26 (b) Topopology using Channel 20 (c) Topopology using Channel 15

Fig. 3. Acceptance ratios of the delay analysis under varying number of flows



(a) Topopology using Channel 26 (b) Topopology using Channel 20 (c) Topopology using Channel 15

Fig. 4. Pessimism ratios in delay bounds

while our analysis has determined 631 cases as schedulable. That is almost 69% of the schedulable cases were deemed schedulable by our analysis. Similarly, for 20 flows 783 test cases are schedulable through simulations under DB while our analysis has determined 364 cases as schedulable. For DM, 783 test cases are schedulable through simulations while our analysis has determined 341 cases as schedulable. Due to less connectivity on Channel 20 we have less schedulable cases compared to the topology on Channel 26. Figure 3(c) shows similar results for the topology collected under Channel 15.

In all cases in the above experiments, the acceptance ratios decrease sharply with the increase in the number of flows. This happens because with the increase of the number of flows in 69 nodes, the network becomes highly congested, thereby sharply degrading their schedulability. Since we generate up to 40 flows, and each flow involves a source node and a destination node, some nodes are selected both as a source of some flow and as a destination of another flow. Hence the overestimate in computing the conflict delay in our analysis sharply increases with the increase of the number of flows, thereby sharply decreasing the acceptance ratios of the analysis also. Since for every flow we have to allocate many retransmissions to handle failures, it is extremely difficult to establish very tight delay bounds. Yet, for a moderate number of flows our analysis always performs very well. Note that DB is more effective than DM in priority assignment in this scheduling. Also the

acceptance ratios of our analysis under DB is consistently higher than that under DM. This is quite reasonable because DB priority assignment policy is based on delay bounds.

Figure 4 plots the pessimism ratios of the flows under our analysis for randomly selected 8 test cases each consisting of 20 flows. Here we show the results when priorities of the flows are assigned using DB. Figure 4(a) shows the results for the topology at Channel 26. It indicates that the 75th percentile of the pessimism ratios is less than 2.5 in first 7 test cases. For the 8th case, the 75th percentile is below 3 and the median is below 2.75. Figure 4(b) shows the results for the topology at Channel 20. It indicates that the 75th percentile of the pessimism ratios in all cases remains within 2.75 and the median is below 2.5 for all but the first case. For the first test case, the median is below 2.6. Figure 4(c) shows the results for the topology at Channel 15. It indicates that the 75th percentile of the pessimism ratios in first 7 cases remains within 2.5 while in the 8 the case it is below 2.65. These results indicate that our delay bounds are not overly pessimistic. The delay bounds are mostly overestimated by a factor of 2.5. Also, we have observed similar statistics of pessimism ratios when priorities of the flows are assigned under DM, and hence those results are not shown here.

The results indicate that our analysis can be used as an effective schedulability test for real-time flows in WirelessHART networks. The pessimism ratios of the flows indicate that the analytical upper bounds of the end-to-end delays of real-time flows are at an acceptable level of pessimism. In every setup, we have observed that the acceptance ratios of our analysis are close to those of simulation for a moderate number of flows. In addition, all test cases accepted by our analysis meet their deadlines in the simulations which demonstrates that the estimated bounds are safe.

VII. Conclusion

Industrial wireless sensor networks must support reliable and real-time communication in hash environments. It is therefore critical to account for transmission failures in the delay analysis for these networks. In this work, we have proposed an efficient end-to-end delay analysis for real-time data flows in WirelessHART networks. A key feature of our analysis is that it incorporates the reliability features of the WirelessHART standard into the end-to-end delay bounds. To our knowledge, this is the first delay analysis cognizant of transmission failures in WirelessHART networks. Specifically, we have considered reliable real-time scheduling based on graph routing where communication failures are handled through retransmissions using dedicated and shared time slots through different paths in the routing graphs. Simulation studies based on the topologies of a wireless sensor network testbed consisting of 69 TelosB motes demonstrate that the proposed analysis provides safe upper bounds of the end-to-end delays of real-time flows at an acceptable level of pessimism.

References

[1] "WirelessHART specification," 2007, http://www.hartcomm2.org.

[2] D. Chen, M. Nixon, and A. Mok, *WirelessHART™ Real-Time Mesh Network for Industrial Automation*. Springer, 2010.

[3] J. Song, A. K. Mok, D. Chen, and M. Nixon, "Challenges of wireless control in process industry," in *Workshop on Research Directions for Security and Net. in Critical Real-Time and Embed. Sys.*, 2006.

[4] S. Han, X. Zhu, and A. K. Mok, "Reliable and real-time communication in industrial wireless mesh networks," in *RTAS '11*.

[5] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "End-to-end delay analysis for fixed priority scheduling in WirelessHART networks," in *RTAS '11*.

[6] J. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou, "Real-time communication and coordination in embedded sensor networks," *Proceedings of the IEEE*, vol. 91, no. 7, pp. 1002–1022, 2003.

[7] H. Li, P. Shenoy, and K. Ramamritham, "Scheduling messages with deadlines in multi-hop real-time sensor networks," 2005, pp. 415–425.

[8] X. Wang, X. Wang, X. Fu, G. Xing, and N. Jha, "Flow-based real-time communication in multi-channel wireless sensor networks," in *EWSN '09*.

[9] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly, "Distributed multi-hop scheduling and medium access with delay and throughput constraints," in *MobiCom '01*.

[10] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He, "RAP: A real-time communication architecture for large-scale wireless sensor networks," in *RTAS '02*.

[11] K. Karenos, V. Kalogeraki, and S. V. Krishnamurthy, "A rate control framework for supporting multiple classes of traffic in sensor networks," in *RTSS '05*.

[12] K. Karenos and V. Kalogeraki, "Real-time traffic management in sensor networks," in *RTSS '06*.

[13] T. He, B. M. Blum, Q. Cao, J. A. Stankovic, S. H. Son, and T. F. Abdelzaher, "Robust and timely communication over highly dynamic sensor networks," *Real-Time Syst.*, vol. 37, no. 3, 2007.

[14] T. W. Carley, M. A. Ba, R. Barua, and D. B. Stewart, "Contention-free periodic message scheduler medium access control in wireless sensor/actuator networks," in *RTSS '03*.

[15] K. Liu, N. Abu-Ghazaleh, and K.-D. Kang, "JiTS: Just-in-time scheduling for real-time sensor data dissemination," in *PERCOM '06*.

[16] Y. Gu, T. He, M. Lin, and J. Xu, "Spatiotemporal delay control for low-duty-cycle sensor networks," in *RTSS '09*.

[17] N. Pereira, B. Andersson, E. Tovar, and A. Rowe, "Static-priority scheduling over wireless networks with multiple broadcast domains," in *RTSS '07*.

[18] O. Chipara, C. Lu, and G.-C. Roman, "Real-time query scheduling for wireless sensor networks," in *RTSS '07*.

[19] T. F. Abdelzaher, S. Prabh, and R. Kiran, "On real-time capacity limits of multihop wireless sensor networks," in *RTSS '04*.

[20] P. Jurcík, R. Severino, A. Koubâa, M. Alves, and E. Tovar, "Real-time communications over cluster-tree sensor networks with mobile sink behaviour," in *RTCSA '08*.

[21] J. B. Schmitt and U. Roedig, "Sensor network calculus A framework for worst case analysis," in *DCOSS '05*.

[22] H. Zhang, P. Soldati, and M. Johansson, "Optimal link scheduling and channel assignment for convergecast in linear WirelessHART networks," in *IEEE WiOpt*, 2009.

[23] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-time scheduling for WirelessHART networks," in *RTSS '10*.

[24] ——, "Priority assignment for real-time flows in WirelessHART networks," in *ECRTS '11*.

[25] P. Soldati, H. Zhang, and M. Johansson, "Deadline-constrained transmission scheduling and data evacuation in WirelessHART networks," in *ECC '09*.

[26] A. Saifullah, C. Wu, P. Tiwari, Y. Xu, Y. Fu, C. Lu, and Y. Chen, "Near optimal rate selection for wireless control systems," in *RTAS '12*.

[27] H. Zhang, F. Osterlind, P. Soldati, T. Voigt, and M. Johansson, "Rapid convergecast on commodity hardware: Performance limits and optimal policies," in *SECON '10*.

[28] N. Guan, M. Stigge, W. Yi, and G. Yu, "New response time bounds for fixed priority multiprocessor scheduling," in *RTSS '09*.

[29] M. Bertogna, M. Cirinei, and G. Lipari, "Schedulability analysis of global scheduling algorithms on multiprocessor platforms," *Parallel and Dist. Sys., IEEE Transactions on*, vol. 20, no. 4, pp. 553 –566, 2009.

[30] Wireless sensor network testbed, http://mobilab.wustl.edu/testbed.

[31] N. C. Audsley., "On priority assignment in fixed priority scheduling," *Information Processing Letters*, vol. 79, no. 1, 2001.