# Priority Assignment for Real-Time Flows in WirelessHART Networks

Abusayeed Saifullah, You Xu, Chenyang Lu, and Yixin Chen
Department of Computer Science and Engineering
Washington University in St. Louis
{saifullaha, yx2, lu, chen}@cse.wustl.edu

***Abstract*—WirelessHART is a new wireless sensor-actuator network standard specifically developed for process industries. A key challenge faced by WirelessHART networks is to meet the stringent real-time communication requirements imposed by process monitoring and control applications. Fixed-priority scheduling, a popular scheduling policy for real-time networks, has recently been shown to be an effective real-time transmission scheduling policy in WirelessHART networks. Priority assignment has a major impact on the schedulability of real-time flows in these networks. This paper investigates the open problem of priority assignment for periodic real-time flows in a WirelessHART network. We first propose an optimal priority assignment algorithm based on local search for any given worst case delay analysis. We then propose an efficient heuristic search algorithm for priority assignment. We also identify special cases where the heuristic search is optimal. Simulations based on random networks and the real topology of a physical sensor network testbed showed that the heuristic search algorithm achieved near optimal performance in terms of schedulability, while significantly outperforming traditional priority assignment policies for real-time systems.**

## I. Introduction

Wireless Sensor-Actuator Networks (WSANs) represent a new generation of communication technology for industrial process control. A feedback control system in process industries (e.g., oil refineries) is implemented in a WSAN for process monitoring and control applications. Networked control loops impose stringent reliability and real-time requirements for communication of sensors and actuators [16]. To meet these requirements in harsh industrial environments, WirelessHART has been developed as an open WSAN standard with unique features such as centralized network management and scheduling, multi-channel TDMA, redundant routes, and channel hopping [2], [7]. With the adoption of WirelessHART, recent years have seen successful real-world deployment of WSANs for process monitoring and control [7]. As they continue to evolve in process industries, real-time transmission scheduling issues are becoming increasingly important for WirelessHART networks.

In this paper, we consider a WirelessHART network that supports feedback control loops through periodic real-time data flows from sensors to controllers and then to actuators. We focus on priority assignment for real-time flows whose transmissions are scheduled based on a fixed priority policy. Due to its simplicity and efficiency, fixed priority scheduling is a commonly adopted real-time scheduling policy in CPU scheduling and traditional real-time networks (e.g., Control-Area Networks). Recent study has shown that fixed priority scheduling is an effective policy for real-time flows in WirelessHART networks and developed worst case delay analysis to be used for efficient schedulability test [13]. Priority assignment has a significant impact on the schedulability of real-time flows. However, optimal priority assignment for WirelessHART networks is a challenging and open problem that has not been addressed in the literature. An ideal priority assignment should not only enable real-time flows to meet their deadlines, but also work synergistically with real-time schedulability tests to support effective network capacity planning and efficient online admission control and adaptation.

For a given schedulability test, a priority assignment algorithm is *optimal* if it can find a priority ordering under which a set of flows is deemed schedulable by the test whenever there exists any such priority ordering. Since an optimal priority assignment is NP-hard for all but a few special cases, simple heuristics such as Deadline Monotonic and Rate Monotonic policies are commonly adopted in practice in real-time networks [12]. However, as shown in our simulation results presented in this paper, the effectiveness of these heuristics in WirelesHART networks is far from the optimal.

This paper is the first to address the optimal priority assignment problem for real-time flows in WirelessHART networks. Specifically, our key contributions are four-fold: (1) We design a local search algorithm for priority assignment that is optimal for any given schedulability test based on worst case delay analysis; (2) We propose an efficient heuristic search algorithm for priority assignment; (3) We identify special cases where the heuristic search is optimal; (4) We present simulation results based on both random networks and the real topology of a physical sensor network testbed. Our results showed that the heuristic search algorithm achieved near-optimal performance in terms of schedulability, while significantly outperforming traditional real-time priority assignment policies.

In the rest of this paper, Section II describes the WirelessHART network model. Section III defines the priority assignment problem. Section IV reviews existing schedulability tests and identifies the key insights underlying the priority assignment approach. Sections V and VI present the optimal local search and the heuristic search algorithms, respectively. Section VII presents the simulation results. Section VIII reviews the related works. Section IX concludes the paper.

## II. WirelessHART Network Model

We consider a WirelessHART network consisting of field devices, a gateway, and a centralized network manager. A *field device* is a sensor, an actuator or both, and is usually connected to process or plant equipment. The gateway provides the host system with access to the network devices. The network manager is located at the gateway and has the complete information of the network. It creates schedules, and distributes among the devices. The unique features that make WirelessHART particularly suitable for industrial process control are as follows.

*Limiting Network Size*. Experiences in process industries have shown the daunting challenges in deploying large-scale WSANs. The limit on the network size for a WSAN makes the centralized management practical and desirable, and enhances the reliability and real-time performance. Large-scale networks can be organized by using multiple gateways or as hierarchical networks that connect small WSANs through traditional resource-rich networks such as Ethernet and 802.11 networks.

*Time Division Multiple Access (TDMA)*. In contrast with CSMA/CA protocols, TDMA protocols provide predictable communication latencies, thereby making themselves an attractive approach for real-time communication. In WirelessHART networks, time is synchronized and slotted. The length of a time slot allows exactly one transmission and its associated acknowledgement between a device pair.

*Route and Spectrum Diversity*. Spatial diversity of routes allows messages to be routed through multiple paths to mitigate physical obstacles, broken links, and interference. Spectrum diversity gives the network access to all 16 channels defined in IEEE 802.15.4 physical layer and allows per time slot *channel hopping* to avoid jamming and mitigate interference from coexisting wireless systems. Besides, any channel that suffers from persistent external interference is *blacklisted* and not used. The combination of spectrum and route diversity allows a packet to be transmitted multiple times, over different channels over different paths, thereby handling the challenges of network dynamics in harsh and variable environments at the cost of redundant transmissions and scheduling complexity.

*Handling Internal Interference*. Due to difficulty in detecting interference between nodes and the variability of interference patterns, WirelessHART allows only one transmission in each channel in a time slot across the entire network, thereby avoiding spatial reuse of channels [7]. Thus, there are at most $m$ concurrent transmissions across the network at any slot, with $m$ being the number of channels. This design decision effectively avoids transmission failure due to interference between concurrent transmissions, and improves the reliability at the potential cost of reduced throughput. The potential loss in throughput is also mitigated due to small size of network.

With the above features, WirelessHART forms a mesh network modeled as a graph $G = (V, E)$, where the node-set $V$ consists of the gateway and field devices, and the edge-set $E$ is the set of communication links between the nodes. A node can send, receive, and route packets but cannot both send and receive in the same time slot. In addition, two transmissions that have the same intended receiver interfere each other. A transmission involves exactly one pair of devices connected by an edge. Therefore, two transmissions that happen along edges $uv$ and $cd$, respectively, are *conflicting* if $(u = c) \vee (u = d) \vee (v = c) \vee (v = d)$. Since conflicting transmissions cannot be scheduled in the same slot, transmission conflicts significantly contribute to communications delays.

## III. Problem Definition

*Real-time flows*. We consider a WirelessHART network with a set of end-to-end flows. Associated with every flow are a sensor node called the *source*, an actuator called the *destination* of the flow, and one or more routes connecting its source to destination through the gateway (where controllers are located). Each flow periodically generates a packet at its source which has to be delivered to its destination within a deadline. A flow may need to deliver its packet through multiple routes. If the delivery through a route fails or some link on a route is broken, the packet can still be delivered through another route. In a schedule, time slots must be reserved for transmissions through each route associated with a flow for redundancy. For schedulability test and priority assignment purposes, through each of its associated routes a flow is treated as an individual flow with the same deadline and period. Therefore, from now onward the term 'flow' will refer to flow through a single route. That is, an original flow with $\phi$ routes is considered $\phi$ flows each with a single route. Thus, we consider there are $n$ flows denoted by the set $F$. Each flow $b \in F$ is characterized by a period $T_b$, a deadline $D_b \leq T_b$, a source, a destination, and a route from its source to destination through the gateway. For each flow $b \in F$, the number of transmissions required to deliver a packet through its route is denoted by $C_b$. Thus, $C_b$ is the number of time slots required by flow $b \in F$.

*End-to-end delay*. For a flow, if a packet generated at slot $r$ is delivered to its destination at slot $d$ then its *end-to-end delay* for this packet is defined as $d - r + 1$. The worst case end-to-end delay of flow $b \in F$ is denoted by $L_b$.

*Fixed priority scheduling*. In a *fixed priority scheduling policy*, each flow has a fixed priority. At any time slot, among all ready transmissions that do not conflict with the transmissions already scheduled in the same slot, the transmission of the highest priority flow is scheduled on an available channel.

*Schedulability*. Transmissions are scheduled using $m$ channels. The set of periodic flows $F$ is called *schedulable* under a scheduling algorithm $\mathbb{A}$, if $\mathbb{A}$ is able to schedule all transmissions in $m$ channels such that no deadline is missed, i.e., $L_b \leq D_b, \forall b \in F$.

*Schedulability test*. For $\mathbb{A}$, a schedulability test $S$ is *sufficient* if any set of flows deemed schedulable by $S$ is indeed schedulable by $\mathbb{A}$. $S$ is *necessary* if any set of flows deemed unschedulable by $S$ is indeed unschedulable by $\mathbb{A}$. $S$ is *exact* if it is both sufficient and necessary. For a set of flows, an *end-to-end delay analysis* provides a sufficient schedulability test by showing that, for every flow, an upper bound of its worst case end-to-end delay is no greater than its deadline.

*Priority assignment*. In priority assignment, our objective is to assign a distinct priority to every flow. Given set $F$ of $n$ flows, we have priority levels 1 to $n$ denoted by set $P = \{1, 2, \cdots, n\}$. Any *priority assignment* or *ordering* is, thus, a one-to-one function $f : P \rightarrow F$, where $f(i) = b$ if and only if the priority of flow $b \in F$ is $i \in P$. A priority level $i$ is *higher* than another priority level $j$ if and only if $i < j$. Given a priority assignment, $hp(b)$ denotes the set of flows whose priorities are higher than that of flow $b$.

*Optimal priority assignment algorithm*. For a schedulability test $S$, a priority assignment is called *acceptable* if under that assignment all flows are guaranteed to meet their deadlines according to $S$. For flow $b \in F$, let $R_b$ denote its worst case end-to-end delay according to $S$. A priority assignment is *acceptable* denoted by $f^{opt} : P \rightarrow F$ if it satisfies $S$, i.e., $R_b \leq D_b, \ \forall b \in F$. For a schedulability test $S$, a priority assignment algorithm is called *optimal* if it can find an acceptable priority assignment whenever there exists any acceptable assignment. That is, if there exists any priority assignment under which $S$ will determine the flows as schedulable, then an *optimal algorithm* is able to find that priority assignment.

## IV. End-to-End Delay Analysis

In this section, we analyze some properties of the existing schedulability tests for real-time flows in WirelessHART networks. These properties provide key insights for an optimal priority assignment algorithm, and intuition for efficient heuristics. We focus on worst case delay analysis (known as *response time analysis* in CPU scheduling), a common approach for schedulability tests in CPU scheduling [6], [11] and WirekessHART networks [13]. These tests are based on efficient but pessimistic analysis of end-to-end delays, that provide a sufficient but not necessary condition for schedulability.

To find an optimal priority assignment policy for a schedulability test, the idea is to start from the lowest priority level, and to upper bound and lower bound the end-to-end delays of the flows according to the test, thereby avoiding unnecessary options for priority assignment at higher levels. To estimate these bounds, we identify a class of schedulability tests, named *Class-1*, in which the worst case end-to-end delay of a flow depends on the worst case end-to-end delays of higher priority flows. The other class of tests in which the worst case end-to-end delay of a flow is independent of the worst case end-to-end delays of higher priority flows is named as *Class-2*. Our proposed algorithms work with both classes of schedulability tests. While Class-1 tests are usually more precise than Class-2, Class-2 tests provide the advantages of simplifying the search for priority assignment. In the following, we analyze this using one representative schedulability test of each class.

### A. Class-1 Schedulability Test

An example of a Class-1 schedulability test for real-time flows in WirelessHART networks was proposed in [13]. Given the fixed priorities of the flows, a lower priority flow can be delayed by the higher priority ones due to (a) *channel contention* (when all channels are assigned to transmissions of higher priority flows in a slot), and (b) *transmission conflicts* (when a transmission of the flow and a transmission of a higher priority flow conflict). $\Delta(b, a)$ denotes an upper bound of the delay that flow $b$ can experience from a flow $a \in hp(b)$ due to transmission conflicts. $\Omega_b(x)$ denotes the total delay (in an interval of $x$ slots) caused by all higher priority flows on $b$ due to channel contention. According to this test, the worst case end-to-end delay $R_b$ of flow $b$ is the minimum value of $y \geq x^*$ that solves Equation 1, where $x^*$ is the minimum value of $x \geq C_b$ that solves Equation 2 using a fixed-point algorithm. If $x$ or $y$ exceeds $D_b$, then $b$ is decided to be "unschedulable".

$$y = x^* + \sum_{a \in hp(b)} \left\lceil \frac{y}{T_a} \right\rceil . \ \Delta(b, a) \tag{1}$$

$$x = \left\lfloor \frac{\Omega_b(x)}{m} \right\rfloor + C_b \tag{2}$$

$\Delta(b, a)$ is calculated by finding the possible conflicting transmissions of $a$ and $b$ by comparing their routes. For $b$, $\sum_{a \in hp(b)} \lceil \frac{y}{T_a} \rceil \Delta(b, a)$ is an upper bound of its total delay (in an interval of $y$ slots) due to transmission conflicts with $hp(b)$. $\Omega_b(x)$ is calculated based on a mapping of the transmission scheduling in a WirelessHART network to the multiprocessor scheduling. Specifically, $\Omega_b(x)$ is the delay of $b$ when the flows are executed on multiprocessor, and is analyzed using the response time analysis considering each $R_a$ as the worst case response time of $a \in hp(b)$. The authors used the state-of-the-art response time analysis for multiprocessor scheduling [11] as a representative method to calculate $\Omega_b(x)$. Since schedulability test is not the focus of our work, we skip its details and refer to [13]. We point out our observations on this test.

In this test, when set $hp(b)$ is known for $b$, the term $\sum_{a \in hp(b)} \lceil \frac{y}{T_a} \rceil \Delta(b, a)$ can be calculated. But $\Omega_b(x)$ depends on $R_a$ of every $a \in hp(b)$ and, hence, is different for different priority ordering among $hp(b)$. Therefore, $\Omega_b(x)$ cannot be calculated if we know only set $hp(b)$. Thus, the bounds of $R_b$ depend on the bounds of $\Omega_b(x)$. $\Omega_b(x)$ non-decreases with the increase of $R_a$, and non-increases with the decrease of $R_a$ of $a \in hp(b)$. Hence, a lower bound and an upper bound of $\Omega_b(x)$ can be derived when it is calculated with a lower bound and upper bound, respectively, of $R_a$ for every $a \in hp(b)$. Note that $\sum_{a \in hp(b)} \lceil \frac{y}{T_a} \rceil \Delta(b, a)$ is a dominating term in $R_b$ due to high degree of conflicts in the network specially near the gateway (since all flows pass through the gateway). Therefore, our calculated bounds for $R_b$, for any $b \in F$, are tight even if we set $\Omega_b(x)$ to 0 to find a lower bound, or to maximum channel contention delay to find an upper bound of $R_b$.

### B. Class-2 Schedulability Test

Now we present a schedulability test of Class-2. An optimal priority assignment policy for this test is comparatively easier since the worst case end-to-end delay of a flow can be derived whenever its higher priority flows are known. Such an observation has been made previously in [9], [10] for priority assignment in multiprocessor scheduling by exploiting the analogy with that in uniprocessor scheduling [5].

When $R_b$ is calculated using the fixed-point algorithm in Equation 2 for flow $b$, $x^*$ represents the worst case response time of $b$ when the flows are executed on multiprocessor. Now, to determine $R_b$ using Equation 2, $x^*$ for flow $b \in F$ is calculated based on the polynomial-time response time analysis for multiprocessor proposed in [6] as follows.

$$x^* = C_b + \left\lfloor \frac{1}{m} \sum_{a \in hp(b)} \min(W_b(a), D_b - C_b + 1) \right\rfloor \quad (3)$$

where $W_b(a) = \lambda_b(a).C_a + \min(C_a, D_b + D_a - C_a - \lambda_b(a).T_a)$; and $\lambda_b(a) = \lfloor \frac{D_b + D_a - C_a}{T_a} \rfloor$.
Here $x^*$ for flow $b$ is a function of $C_b$, $D_b$, and of $C_a$, $T_a$, and $D_a$ of every $a \in hp(b)$, which remain unchnaged over every priority ordering among $hp(b)$. Hence, $R_b$ can be found using Equations 1 and 3 when the set $hp(b)$ is known. This test, hence, represents Class-2 schedulability tests.

## V. Priority Assignment Using Local Search

In this section, we exploit the observations made in Section IV on the classification of schedulability tests and develop an optimal priority assignment algorithm based on local search (LS). Given a schedulability test $S$ and set $F$ of $n$ flows, if there exists any acceptable priority assignment, then the optimal algorithm is able to find that assignment. If no acceptable assignment exists, then it returns a priority assignment that is likely to be good for schedulability of the flows. The proposed algorithm is compatible to any Class-1 and Class-2 schedulability test (Section IV). We also analyze some special cases where the algorithm runs in pseudo polynomial time.

The idea underlying local search is to lower bound and upper bound the worst case end-to-end delay according to $S$ of every flow in an acceptable priority assignment. Starting from the lowest priority level, the algorithm explores different options, in the form of a search tree, for assigning priorities at higher levels. For a possible acceptable assignment in a subtree, if, for every flow, an upper bound of its worst case end-to-end delay according to $S$ happens to be no greater than its deadline, then the subtree is a *sufficient branch* and all other branches are discarded. Upper bounding the delays, thus, provides a *sufficient condition* that guarantees that an acceptable assignment can be found in a branch. For a possible acceptable assignment in a subtree, if, for every flow, a lower bound of its worst case end-to-end delay according to $S$ is no greater than its deadline, then the subtree is designated as a *necessary branch*. Thus, lower bounding the delays provides a *necessary condition*. Any branch that dissatisfies this condition is guaranteed not to lead to an acceptable assignment, and is discarded as an *unnecessary branch*.

Having the above idea, the search starts from any initial priority assignment $f : P \rightarrow F$. If an acceptable priority assignment exists, then it can be found by reordering $f$. Every node in the tree performs a reordering of the priorities, thereby representing a new priority assignment. Specifically, the search starts reordering from the lowest priority level $n$. When it reaches priority level $l \geq 1$, a node has $l-1$ options to assign priority $l-1$. For every option, it generates a child node. The branches in the subtree rooted at each child node represent different reordering from level 1 to $l-1$. Considering $f$ as the priority assignment at a node, we introduce the following notations to establish the bounds in its subtree:

- $R^{\text{opt}}_{f(i)}$: denotes the worst case end-to-end delay of $f(i)$ according to $S$ in an acceptable priority assignment $f^{\text{opt}}$.
- $R^{\text{ub}}_{f(i)}$: denotes an upper bound of $R^{\text{opt}}_{f(i)}$.
- $R^{\text{lb}}_{f(i)}$: denotes a lower bound of $R^{\text{opt}}_{f(i)}$.
- $f_{l,k}$: denotes the priority assignment from level $l$ to $k$ in $f$, where $1 \leq l \leq k \leq n$ (Figure 1). In $f$, a partial priority assignment $f_{l,k}$ is called *acceptable* if $f_{l,k} = f^{\text{opt}}_{l,k}$, i.e., the assignment from priority level $l$ to $k$ in $f$ is the same as that in an acceptable priority assignment.

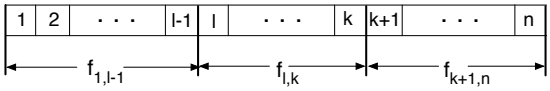



Fig. 1. Priority assignment $f$ at a node

- $R^{\infty}_{f(i)}$: denotes an upper bound of the end-to-end delay of $f(i)$ under infinite number of channels, i.e., when $f(i)$ experiences no channel contention and is delayed only due to transmission conflicts with the higher priority flows. Therefore, for flow $f(i)$, $R^{\infty}_{f(i)}$ is calculated using Equation 1 with $x^*$ being replaced by $C_{f(i)}$.

### A. Upper Bound of Worst Case End-to-End Delay

The upper bounds are calculated based on the observations made in Section IV. Specifically, an upper bound of the worst case end-to-end delay of a flow is determined by considering the upper bounds of its higher priority flows.

In a priority assignment $f$, let the assignment $f_{k+1,n}$ from level $k+1$ ($\leq n$) to $n$ has been decided to be in an acceptable assignment. To decide whether the assignment $f_{l,k}$ from level $l$ ($\leq k$) to $k$ is also in that acceptable assignment, the upper bound $R^{\text{ub}}_{f(i)}$ for every $f(i)$, $l \leq i \leq k$, is calculated as follows.

- The set $\{f(j) | 1 \leq j < l\}$ is considered as the set of higher priority flows of $f(l)$. When $l = 2$, $R^{\text{ub}}_{f(1)}$ for flow $f(1)$ is set to $C_{f(1)}$. When $l > 2$, $R^{\text{ub}}_{f(j)}$ of every flow $f(j)$, $1 \leq j < l$, is set to its deadline $D_{f(j)}$.
- If $l \leq m$, flow $f(i)$, $l \leq i \leq m$, does not experience any channel contention, and hence $R^{\text{ub}}_{f(i)} = R^{\infty}_{f(i)}$. For every other flow $f(i)$, $m < i \leq k$, $R^{\text{ub}}_{f(i)}$ is calculated according to schedulability test $S$ using $R^{\text{ub}}_{f(j)}$ as the worst case end-to-end delay of $f(j)$ for every $j < i$.
- If $l > m$, then for every flow $f(i)$, $l \leq i \leq k$, $R^{\text{ub}}_{f(i)}$ is calculated according to $S$ using $R^{\text{ub}}_{f(j)}$ as the worst case end-to-end delay of $f(j)$ for every $j < i$.

The upper bound calculation is shown as Procedure $\text{S}^{ub}(f,\ l,\ k)$. In this procedure, S($f(i)$) returns $f(i)$'s worst case end-to-end delay according to $S$ using $R^{\text{ub}}_{f(j)}$ as the worst case end-to-end delay of $f(j)$ for every $j < i$. $\text{S}^{ub}(f,\ l,\ k)$ returns *false* if the upper bound is greater than deadline for any flow $f(i)$, $l \leq i \leq k$. Otherwise, it returns $true$. Thus, if $\text{S}^{ub}(f,\ l,\ k)$ returns $true$, then it is a guarantee that there

exists a priority assignment among the flows $\{f(j)|1 \leq j < l\}$ such that using that assignment (from priority level 1 to $l-1$), and the current assignment $f_{l,k}$ ( from level $l$ to $k$), the resulting assignment from level 1 to $k$ is the same as that in an acceptable assignment. $\mathrm{S}^{ub}(f,\ l,\ k)$ thus provides a *sufficient condition* to determine if reordering $f_{1,l-1}$ can guarantee an acceptable assignment. Any partial assignment $f_{l,k}$ is said to *satisfy* $\mathrm{S}^{ub}(f,\ l,\ k)$, if $\mathrm{S}^{ub}(f,\ l,\ k)$ returns $true$.

**bool** $\mathrm{S}^{ub}(f,\ l,\ k)$
**begin**
  **if** $l = 2$ **then** $R^{\mathrm{ub}}_{f(1)} \leftarrow C_{f(1)}$;
  **else**
    **for** $j = 1$ *to* $l-1$ **do** $R^{\mathrm{ub}}_{f(j)} \leftarrow D_{f(j)}$;
  **end**
  **if** $l \leq m$ **then**
    $l' = \min(m, k)$;
    **for** $i = l$ *to* $l'$ **do**
      $R^{\mathrm{ub}}_{f(i)} \leftarrow R^{\infty}_{f(i)}$;  `/* Exact value */`
      **if** $R^{ub}_{f(i)} > D_{f(i)}$ **then return** *false*;
    **end**
    $l \leftarrow l' + 1$;
  **end**
  **for** $i = l$ *to* $k$ **do**
    $R^{\mathrm{ub}}_{f(i)} \leftarrow \mathrm{S}(f(i))$;  `/* Using test S */`
    **if** $R^{ub}_{f(i)} > D_{f(i)}$ **then return** *false*;
  **end**
  **return** $true$;
**end**

**Procedure** $\mathrm{S}^{ub}(f, l, k)$: Upper Bound Calculation

From our discussions in Section IV, in the above upper bound calculation, $R^{\mathrm{ub}}_{f(i)}$ for every $f(i),\ l \leq i \leq k$, includes its exact delay due to transmission conflicts (according to $S$). This delay is a dominating term in the worst case end-to-end delay of a flow due to high degree of conflicts in the network specially near the gateway (since all the flows pass through the gateway). As a result, our upper bound estimation becomes precise, thereby making the sufficient condition strong.

*Theorem 1:* Let $f : P \rightarrow F$ be any priority assignment. If there exists an acceptable priority assignment $f^{\mathrm{opt}}_{1,l-1},\ 2 \leq l \leq n+1$, among flows $\{f(i)|1 \leq i < l\}$, then, for $f_{l,k},\ l \leq k \leq n$, $R^{\mathrm{ub}}_{f(i)}$ calculated in Procedure $\mathrm{S}^{ub}(f, l, k)$ is an upper bound of $R^{\mathrm{opt}}_{f(i)}$ for every $f(i),\ l \leq i \leq k$.

*Proof:* First, let $S$ be a Class-2 schedulability test (Section IV). For each flow $f(i), l \leq i \leq k$, we know its higher priority flows, and $R^{\mathrm{ub}}_{f(i)}$ does not depend on any $R^{\mathrm{ub}}_{f(j)}$ where $j < i$. Hence, $R^{\mathrm{ub}}_{f(i)} = R^{\mathrm{opt}}_{f(i)}$.

Now, let $S$ be a Class-1 schedulability test (Section IV). For any $i,\ l \leq i \leq k$, such that $i \leq m$, flow $f(i)$ does not experience any channel contention. Hence, $R^{\mathrm{ub}}_{f(i)} = R^{\infty}_{f(i)} = R^{\mathrm{opt}}_{f(i)}$ holds. For any $i,\ l \leq i \leq k$, such that $i > m$, Procedure $\mathrm{S}^{ub}(f, l, k)$ computes $R^{\mathrm{ub}}_{f(i)}$ for flow $f(i)$ according to $S$ by considering the upper bounds $R^{\mathrm{ub}}_{f(j)}$ for every higher priority flow $f(j)$ where $j < i$. Hence, based on our observations in Section IV, $R^{\mathrm{ub}}_{f(i)}$ is an upper bound of $R^{\mathrm{opt}}_{f(i)}$. ■

*Theorem 2:* Let there exists an acceptable priority assignment $f^{\mathrm{opt}} : P \rightarrow F$. Let $f : P \rightarrow F$ be any priority assignment such that $f_{k+1,n}$ satisfies $\mathrm{S}^{ub}(f, k+1, n)$. Then $f_{k+1,n} = f^{\mathrm{opt}}_{k+1,n}$, i.e., priority assignment from level $k+1$ to level $n$ in $f$ is the same as that in an acceptable assignment. In other words, there is an ordering of priorities from level 1 to $k$ in $f$ that will give an acceptable priority assignment.

*Proof:* Assume to the contrary that there exists no priority ordering among the flows $\{f(j)|1 \leq j \leq k\}$ for which $f_{k+1,n}$ is a part of an acceptable priority assignment. Therefore, there must be at least one flow $f(j),\ 1 \leq j \leq k$, that cannot be assigned any priority from level 1 to $k$. This implies that we must be able to assign some priority $j'$, where $k < j' \leq n$, to this particular flow $f(j)$ since there exists an acceptable priority assignment. But its worst case end-to-end delay at a lower priority level $j'$ must be no less than that at the higher priority level $j$. That is, if $f(j)$ is schedulable at the lower priority level $j'$, it must be schedulable at the higher priority level $j$ which contradicts our assumption. ■

*Lemma 3:* Let there exists an acceptable priority assignment $f^{\mathrm{opt}} : P \rightarrow F$. Let $f : P \rightarrow F$ be any priority assignment such that $f_{k+1,n} = f^{\mathrm{opt}}_{k+1,n},\ 0 \leq k < n$. Now if $f_{l,k},\ 1 \leq l \leq k$ satisfies $\mathrm{S}^{ub}(f, l, k)$, then $f_{l,n} = f^{\mathrm{opt}}_{l,n}$.

*Proof:* By Theorem 1, $R^{\mathrm{ub}}_{f(i)} \leq D_{f(i)}, l \leq i \leq k$. Having assignment $f_{k+1,n}$, $R^{\mathrm{ub}}_{f(i)}$ will not change since each $f(i')$ with $i' > k$ is a lower priority flow of $f(i),\ l \leq i \leq k$. By Theorem 2, $f_{l,k}$ is in an acceptable assignment. Hence, $f_{l,n} = f^{\mathrm{opt}}_{l,n}$. ■

### *B. Lower Bound of Worst Case End-to-End Delay*

Similar to upper bound calculation, a lower bound of the worst case end-to-end delay of a flow is determined by considering the lower bounds of its higher priority flows.

In a priority assignment $f$, let the assignment $f_{k+1,n}$ from level $k+1 (\leq n)$ to $n$ has been decided to be in an acceptable assignment. To decide whether the assignment $f_{l,k}$ from level $l (\leq k)$ to $k$ is also in that acceptable assignment, the lower bound $R^{\mathrm{lb}}_{f(i)}$ for every $f(i),\ l \leq i \leq k$, is calculated as follows.

- The set $\{f(j)|1 \leq j < l\}$ is the set of higher priority flows of $f(l)$. $R^{\mathrm{lb}}_{f(j)}$ of every flow $f(j),\ 1 \leq j < l$, is set to its number of transmissions $C_{f(j)}$.
- If $l \leq m$, then flow $f(i),\ l \leq i \leq m$, does not experience any channel contention and, hence, $R^{\mathrm{lb}}_{f(i)} = R^{\infty}_{f(i)}$. For every other flow $f(i),\ m < i \leq k$, $R^{\mathrm{lb}}_{f(i)}$ is calculated according to schedulability test $S$ using the $R^{\mathrm{lb}}_{f(j)}$ as the worst case end-to-end delay of $f(j)$ for every $j < i$.
- If $l > m$, then for every flow $f(i),\ l \leq i \leq k$, $R^{\mathrm{lb}}_{f(i)}$ is calculated according to $S$ using the $R^{\mathrm{lb}}_{f(j)}$ as the worst case end-to-end delay of $f(j)$ for every $j < i$.

The procedure for calculating the lower bounds is shown as Procedure $\mathrm{S}^{lb}(f,\ l,\ k)$. In this procedure, $\mathrm{S}(f(i))$ returns $f(i)$'s worst case end-to-end delay according to $S$ using $R^{\mathrm{lb}}_{f(j)}$ as the worst case end-to-end delay of $f(j)$ for every $j < i$.

$S^{lb}(f,\ l,\ k)$ returns *false* if the lower bound is greater than deadline for any flow $f(i),\ l \leq i \leq k$. Otherwise, it returns $true$. Thus, if $S^{lb}(f,\ l,\ k)$ returns *false*, it is a guarantee that no ordering of flows $\{f(j)|1 \leq j < l\}$ can be in an acceptable assignment. $S^{lb}(f,\ l,\ k)$ thus provides a *necessary condition* to determine if reordering $f_{1,l-1}$ can guarantee an acceptable assignment. Any partial assignment $f_{l,k}$ is said to *satisfy* $S^{lb}(f,\ l,\ k)$, if $S^{lb}(f,\ l,\ k)$ returns $true$.

```
bool S^lb(f, l, k)
begin
    for i = 1 to l − 1 do R^lb_{f(i)} ← C_{f(i)};
    if l ≤ m then
        l' = min(m, k);
        for i = l to l' do
            R^lb_{f(i)} ← R^∞_{f(i)};      /* Exact value */
            if R^lb_{f(i)} > D_{f(i)} then return false;
        end
        l ← l' + 1;
    end
    for i = l to k do
        R^lb_{f(i)} ← S(f(i));      /* Using test S */
        if R^lb_{f(i)} > D_{f(i)} then return false;
    end
    return true;
end
```

**Procedure** $S^{lb}(f, l, k)$: Lower Bound Calculation

From our discussions in Section IV, $R^{\text{lb}}_{f(i)}$ for every $f(i),\ l \leq i \leq k$, includes its exact delay due to transmission conflicts (according to $S$). This delay is a dominating term in the worst case end-to-end delay of a flow due to high degree of conflicts in the network specially near the gateway (since all the flows pass through the gateway). As a result, like the upper bounds, our lower bound estimation also becomes precise, thereby making the necessary condition strong.

*Theorem 4:* Let $f : P \rightarrow F$ be any priority assignment. Let $f^{\text{opt}}$ be an acceptable priority assignment such that there exists an ordering among flows $\{f(j)|1 \leq j < l\}$ which is also in $f^{\text{opt}}$. Then, for $f_{l,k},\ l \leq k \leq n$, $R^{\text{lb}}_{f(i)}$ calculated in $S^{lb}(f, l, k)$ is a lower bound of $R^{\text{opt}}_{f(i)}$ for every $f(i),\ l \leq i \leq k$.

*Proof:* Similar to Theorem 1, $R^{\text{lb}}_{f(i)} = R^{\text{opt}}_{f(i)}$ for every $f(i),\ l \leq i \leq k$, when $S$ is a Class-2 schedulability test. When $S$ is Class-1 schedulability test, the proof is similar to Theorem 1. ■

Corollary 1 follows from Theorem 4.

*Corollary 1:* For any given priority ordering $f : P \rightarrow F$, if $f_{l,k},\ l \leq k \leq n$, does not satisfy $S^{lb}(f, l, k)$, then no acceptable priority assignment can be found from $f$ by reordering the flows from level 1 to $l-1$.

### C. Local Search Framework

Now we structure the search for an acceptable priority assignment into a local search (LS) framework. Starting from an initial assignment, the algorithm performs a reordering of the priorities at every node of its search tree, thereby creating a new assignment. Specifically, the search starts from the lowest priority level and investigates if any flow that has higher priority in current assignment can be assigned this lower priority and generates a child node representing this new assignment. The branches are discarded or explored based on the lower bounds and upper bounds calculated for a branch.

The search tree has as its root node a Deadline Monotonic (DM) priority ordering. It has a maximum of $n+1$ levels with the root being at level $n+1$. If the DM priority assignment is acceptable, then the algorithm terminates. Otherwise, the search branches down by creating new nodes. Every node represents a complete priority assignment a part of which is guaranteed to be in an acceptable assignment. Therefore, besides the priority assignment $f$, every node has two attributes $l$ and $k$, where $1 \leq l \leq n,\ l \leq k \leq n+1$ (Figure 1). In priority assignment $f$ at a node, its part $f_{k+1,n}$ is *guaranteed* to be in an acceptable assignment; $f_{l,k}$ is *not guaranteed but may be* in an acceptable assignment; and $f_{1,l-1}$ is *yet undecided*. Thus, $k$ is the level on the path from root to this node such that every node on this path from level $n$ to $k+1$ has satisfied the sufficient condition. The steps of the search are:

1) The root starts with $l = n+1$ and $k = n$ since no part of its assignment is yet final.
2) For the undecided part $f_{1,l-1}$ of priority assignment $f$ at a node at tree-level $l$, the node creates a child node at tree level $l-1$ for every $i,\ 1 \leq i \leq l-1$, by exchanging the priorities between $f(i)$ and $f(l-1)$.
3) If a child node created in Step 2 satisfies the sufficient condition, then its $k$ becomes 1 less than its $l$ meaning that $f_{k+1,n}$ is now guaranteed to be in an acceptable assignment (according to Theorem 2 and Lemma 3). Hence, all other branches are discarded. This child is expanded further by going to Step 2.
4) If a child node created in Step 2 cannot satisfy the necessary condition, it is closed (according to Corollary 1). Otherwise, it is expanded further by going to Step 2.
5) The search continues creating new nodes until it reaches a node at tree level 1 where $k$ becomes 0 which indicates that an acceptable assignment has been found or until there exists no unexpanded node for which neither the necessary nor the sufficient condition is satisfied. In the latter case, no acceptable assignment is found and the priority assignment of the current node is returned.

The pseudo code is shown as LS Priority Assignment Algorithm. If DM priority assignment $f$ is acceptable, then Procedure $S(f, 1, n)$ returns $true$. Otherwise, the root with $f$, $l = n+1$, and $k = n$ expands by calling procedure LS($root$). The attributes $l$, $k$, and the priority assignment $f$ at any node $nd$ in the search tree is denoted by $nd.l$, $nd.k$, and $nd.f$, respectively. In LS(Node $nd$), if $nd.k = 0$ for current node $nd$, then the search terminates by returning $nd.f$ as an acceptable assignment $f^*$. Otherwise, for every flow $nd.f(i)$ starting from $i = l-1$ to 1, it generates a child node $ch$ with $ch.k = nd.k$ at level $ch.l = nd.l - 1$, and SwapPriority($ch.f(i), ch.f(ch.l)$)

exchanges the priorities between $ch.f(i)$ and $ch.f(ch.l)$. If $S^{ub}(ch.f, ch.l, ch.k)$ returns $true$, then $ch.k$ is updated to $ch.l - 1$, and all other branches are discarded, and only this child is expanded as a sufficient branch by calling LS($ch$). If $S^{lb}(ch.f, ch.l, ch.k)$ returns *false*, then $ch$ is closed. Otherwise, it is expanded as a necessary branch. If the tree cannot expand any more and $k > 0$ at every node, then no acceptable assignment exists, and $f$ of current node is returned as $f^*$.

```
bool LS(Node nd)
begin
    if nd.k = 0 then
        f* ← nd.f; return true;  /* Acceptable */
    end
    for i = nd.l − 1 down to 1 do
        Create a Child Node ch;
        ch.f ← nd.f; ch.l ← nd.l − 1; ch.k ← nd.k;
        SwapPriority(ch.f(i), ch.f(ch.l));
        if S^ub(ch.f, ch.l, ch.k) = true then
            ch.k ← ch.l − 1;      /* Sufficient */
            if LS(ch) = true then     /* branch */
                return true;           /* found */
            break;    /* Cut other branches */
        end
        if S^lb(ch.f, ch.l, ch.k) = false then
            continue;    /* Close this child */
        else if LS(ch) = true then
            return true;  /* Necessary branch */
        end
    end
    f* ← nd.f;              /* No acceptable */
    return false;        /* assignment exists */
end
```

**Procedure** LS(Node $nd$): Local Search

**input** : Set $F$ of $n$ flows, and schedulability test $S$
**output**: $f^* : P \rightarrow F$, where $P = \{1, 2, \cdots, n\}$
$f \leftarrow$Deadline Monotonic priority assignment;
**if** $S(f, 1, n) = true$ **then** /* DM satisfies $S$ */
    $f^* \leftarrow f$; *return* "acceptable assignment found";
**end**
Create a Node $root$ with attributes $f,\ l,\ k$;
$root.f \leftarrow f;\ root.l \leftarrow n + 1;\ root.k \leftarrow n;$
**if** $LS(root) = true$ **then**
    *return* "acceptable assignment found";
**else**
    *return* "no acceptable assignment exists";

**Algorithm:** LS Priority Assignment

*Theorem 5:* For a given set $F$ of $n$ flows and a schedulability test $S$, there exists an acceptable priority assignment of $F$ if and only if the priority assignment $f^*$ returned by the LS algorithm is acceptable.

*Proof:* Let there exists an acceptable priority assignment of $F$. By Theorem 2 and Lemma 3, if the search stops with $k = 0$ at a node, then the priority assignment of that node must be acceptable. Suppose to the contrary the search has stopped at a node $nd$ with $k > 0$ and priority assignment $f$. Since an acceptable priority assignment exists, by Theorem 2, there must exist a priority ordering $f'_{1,k}$ among flows $\{f(i)|1 \leq i \leq k\}$ in $f$ such that $f'_{1,k}$ and the current assignment in $f$ from level $k+1$ to $n$ will give an acceptable assignment. Hence, at least one necessary branch must reach level 1 from $nd$, and at least one such branch that reaches a node $nd'$ at level 1 must correspond to $f'_{1,k}$. Node $nd'$ must satisfy the sufficient condition and its $k'$ is updated to $1 - 1 = 0$ which contradicts our assumption. To prove in the other direction, let $f^*$ returned by the algorithm is considered acceptable. This implies that the search has stopped with $k = 0$. By Theorem 2 and Lemma 3, $f^*$ must satisfy $S$. ■

### D. Analysis

Now we analyze the LS Algorithm for some special cases where it runs in pseudo polynomial time.

*Case 1*. According to Section IV, when $S$ is a Class-2 schedulability test, both the lower bound and the upper bound calculated for a flow are its exact worst case end-to-end delay according to $S$. That is, both the necessary condition and the sufficient condition become exact. As a result, the search tree consists of just one path. If there exists an acceptable assignment, then the path reaches level 1. Otherwise, it stops at some level where no flow can be assigned that priority. In either case, the search always has $l = k$ and, at every level $l$ ($\leq n+1$), it tests at most $l-1$ flows. Thus the algorithm runs in $O(n^2 t)$ time, where $t$ is the time to calculate the worst case end-to-end delay of a flow using $S$ and is pseudo polynomial.

*Case 2*. Based on our observations in Section IV, when $m \geq n$, there is no channel contention and, hence, $R^{\infty}_{f(i)}$ is the worst case end-to-end delay for every flow $f(i)$. Hence, similar to Case 1, both the sufficient condition and the necessary condition are exact, and the algorithm runs in $O(n^2 t)$ time. When $n > m$, the same thing happens when the value of $k$ becomes no greater than $m$ during the search.

*Case 3*. If the DM priority assignment is acceptable, then the search stops immediately and returns that ordering. Assigning DM priorities takes $O(n \log n)$ time, and to verify if it is acceptable by $S$, we need $O(nt)$ time. Hence, the algorithm runs in $O(n \log n + nt)$ time.

## VI. Priority Assignment Using Heuristic Search

While the proposed LS algorithm is optimal and runs efficiently in most cases (as shown in simulation in Section VII), a faster execution time cannot be guaranteed theoretically for all the time. Therefore, in this section, we propose an efficient near-optimal heuristic search. It is also based on the similar strategy but is forced to discard many branches by expanding only the branches deemed good. Hence, it runs much faster at the cost of loosing the optimal behavior in some cases.

The LS algorithm can take longer time mostly when it is hard to find a sufficient branch. The key idea behind the

heuristic search (HS), therefore, is to loose this condition for faster execution. To determine whether the subtree rooted at a node $ch$ is sufficient, the LS algorithm calls Procedure $S^{ub}(ch.f, ch.l, ch.k)$. The procedure determines the branch as sufficient only if the upper bound of the worst case end-to-end delay of every flow $ch.f(i)$, $l \leq i \leq k$, is no greater than its deadline. Since this is an overestimate, the HS algorithm instead checks only for current level $ch.l$. That is, only if the upper bound of the worst case end-to-end delay of flow $ch.f(ch.l)$ is no greater than its deadline, it discards all other branches and expands only this branch. Note that such a branch is still good (but not guaranteed to be the best as the new condition is not sufficient) since every node from level $n$ to $ch.l$ on this branch has either satisfied this new condition or the necessary condition which we have previously argued to be a strong condition because of precise lower bound estimation.

The HS algorithm considers every level from $ch.l$ to $ch.k$ in Procedure $S^{lb}(ch.f, ch.l, ch.k)$ as the necessary condition. Since the *new condition* $S^{ub}(ch.f, ch.l, ch.l) = true$ does not mean that $f_{l,n}$ is acceptable, the search updates $k$ as long as the new condition is satisfied on a root to leaf path, and stops updating it after the first time the new condition is violated on that path. That is, $ch.k$ is updated only if $ch.k \geq ch.l - 1$. The HS algorithm is, thus, pseudo coded by making two changes in Procedure LS(Node $nd$) of the LS algorithm:

1) Replace the condition $S^{ub}(ch.f, ch.l, ch.k) = true$ with $S^{ub}(ch.f, ch.l, ch.l) = true$.
2) Before the statement $ch.k \leftarrow ch.l - 1$, add the check **if**$(ch.k \geq ch.l - 1)$.

By Theorem 2, the partial priority assignment $f_{k+1,n}$ of $f$ at a node of the search tree in the HS algorithm is still guaranteed to be a part of an acceptable assignment (if there exists one at all). However, when the algorithm terminates at a node $nd$, some node on a level from $nd.k$ to $nd.l$ (when $nd.k \neq nd.l$) on the path from the root to $nd$ may have violated the sufficient condition which the HS algorithm is not aware of. In that case, the algorithm is not optimal. However, our simulation studies have shown that such cases hardly happen in practice.

*Analysis*. In Case 1 and Case 2 (Subsection V-D), the optimal LS algorithm always maintains $l = k$. Hence, the new condition used in the HS algorithm becomes a sufficient condition. Case 1 and Case 2, thus, hold for the HS algorithm. That is, if $m \geq n$ or $S$ is a Class-2 schedulability test, then it is optimal and runs in $O(n^2 t)$ time, where $t$ is the time to calculate the worst case end-to-end delay of a flow using $S$. Besides, Case 3 always holds for it. It trivially dominates DM in that whenever the DM priority assignment is acceptable the HS algorithm also determines that assignment as acceptable and runs in $O(n \log n + nt)$ time. In other cases for Class-1 tests, although the execution time of the HS algorithm is theoretically exponential, it can be guaranteed to run faster in practice. A long execution time can happen if the new condition is hardly satisfied or the necessary condition cannot discard enough branches. Note that the new condition is hardly satisfied when the flows have very tight deadlines. However, in this case, the necessary condition will discard many branches. Again, the necessary condition may not discard enough branches if the deadlines are not tight. In this case, the new condition is easily satisfied to discard all other branches, thereby making the search faster.

## VII. Performance Evaluation

We evaluate our priority assignment algorithms through simulations based on both random topologies and the real topology of a physical testbed. We compare the heuristic search (HS) with the optimal local search (LS) algorithm and the following priority assignment policies: (a) *Deadline Monotonic (DM)* assigns priorities to flows according to their relative deadlines; (b) *Proportional Deadline monotonic (PD)* assigns priorities to flows based on *relative subdeadline* defined for a flow as its relative deadline divided by the total number of transmissions along its route.

*Metrics*. We evaluate the algorithms in terms of the following metrics. (a) *Acceptance ratio:* fraction of the test cases deemed schedulable according to the schedulability test used. (b) *Execution time:* average execution time (with the 95% confidence interval) needed to generate a priority assignment.

*Simulation Setup*. A fraction ($\theta$) of nodes is considered as sources and destinations. A node with the highest degree is selected as the gateway. The *reliability* of a link is represented by the *packet reception ratio (PRR)* along it. The most reliable route connecting a source to a destination is selected as the first route. For additional routes (for redundancy) between the same pair, we exclude the links used by existing routes between the pair and select the next most reliable route. Period $T_b$ of every flow $b$ is generated randomly in a range denoted by $T_\sim = 2^{i \sim j}$ slots, $i \leq j$. The relative deadline $D_b$ of every flow $b$ is randomly generated in a range between $C_b$ and $\alpha * T_b$ slots, for $0 < \alpha \leq 1$. The algorithms have been implemented in C and tested on a Macbook Pro laptop. Table I summarizes the notations used in this section.

| | |
|---|---|
| $N$ : | Number of nodes in the network |
| $m$ : | Number of channels |
| $\rho$ : | Edge-density of the network |
| $\theta$ : | Fraction of total nodes that are source or destination |
| $\gamma$ : | Number of routes between every source and destination |
| $T_\sim$ : | Period range |
| $\alpha$ : | Deadline parameter (e.g., $C_b \leq D_b \leq \alpha * T_b$, for flow $b$) |

TABLE I
Notations

### A. Simulations with Testbed Topologies

We evaluate our algorithms on the topology of a physical indoor testbed in Bryan Hall of Washington University [1]. The testbed consists of 48 TelosB motes each equipped with a Chipcon CC2420 radio compliant with IEEE 802.15.4. At transmission power of 0 dBm, 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 5 rounds. Then every link with a higher than 80% PRR is considered *reliable* and drawn in Figure 2 (embedded on the floor plan of Bryan Hall). Using 12 channels,

we compare HS, LS, DM, and PD considering the Class-1 schedulability test presented in Section IV on this topology.

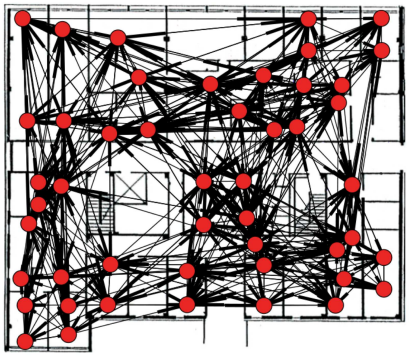
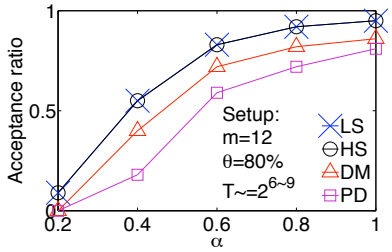
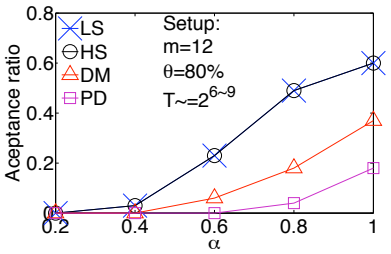


Fig. 2. Testbed topology (at transmission power of 0 dBm) in Bryan Hall

*Varying deadlines*. We generate flows in the network by randomly selecting the sources and destinations considering $\theta = 80\%$ (i.e., 40% of the total nodes are sources while another 40% are destinations). The periods of the flows are randomly generated in range $2^{6\sim9}$ time slots. We generate 100 test cases and plot the acceptance ratios in Figure 3 by varying the deadlines of the flows (by changing $\alpha$). For $\gamma = 1$ (Figure 3(a)), when $\alpha = 0.4$, there are acceptable assignments in 55% test cases but PD is able to find an acceptable assignment only in 18% cases. Thus, the difference between the acceptance ratios of LS and PD is 0.37 when $\alpha = 0.4$. For any $\alpha \geq 0.4$, the difference remains at least 0.14. For DM, this difference is 0.10 to 0.15. When $\gamma = 2$ (Figure 3(b)), the differences are 0.23 to 0.45 for PD, and 0.17 to 0.31 for DM. HS performs almost like an optimal algorithm in this setup since it selects good branches and uses a strong necessary condition to discard unnecessary branches (as explained in Section VI).



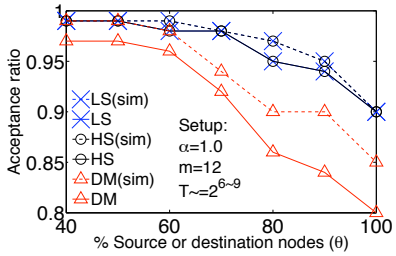

(a) Acceptance ratio when $\gamma = 1$



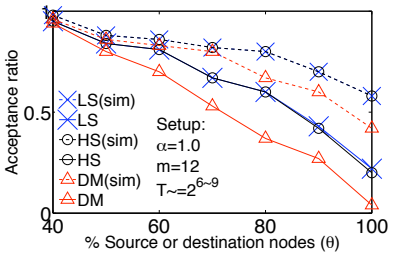

(b) Acceptance ratio when $\gamma = 2$

Fig. 3. Performance under varying deadlines

*Varying sources and destinations*. Now we vary $\theta$ resulting in different numbers of flows in the network. Since PD performs worst, we omit any further comparison with PD. For every $\theta$, we generate 100 test cases and show the performances in Figure 4. Since we use a sufficient schedulability test, there are cases when a priority assignment is not acceptable by the test but the flows may meet their deadlines if they are scheduled using that priority assignment. Therefore, every priority assignment generated by an algorithm is tested in simulation by scheduling all flows within their hyper-period. In the figure, each curve "sim" shows the fractions of test cases that have no deadline misses in simulations. When $\gamma = 1$ (Figure 4(a)), the difference between the acceptance ratios of LS and DM is always 0.02 to 0.11. Their difference in simulation is 0.02 to 0.05 when $\theta \geq 60\%$. When $\gamma = 2$ (Figure 4(b)), their difference is 0.11 to 0.23 in acceptance ratio, and 0.02 to 0.16 in simulation. Here, HS performs like LS when $\gamma = 1$. When $\gamma = 2$, its acceptance ratio is 0.01 to 0.02 less than that of LS, if $\theta \geq 80\%$.




(a) Acceptance ratio when $\gamma = 1$



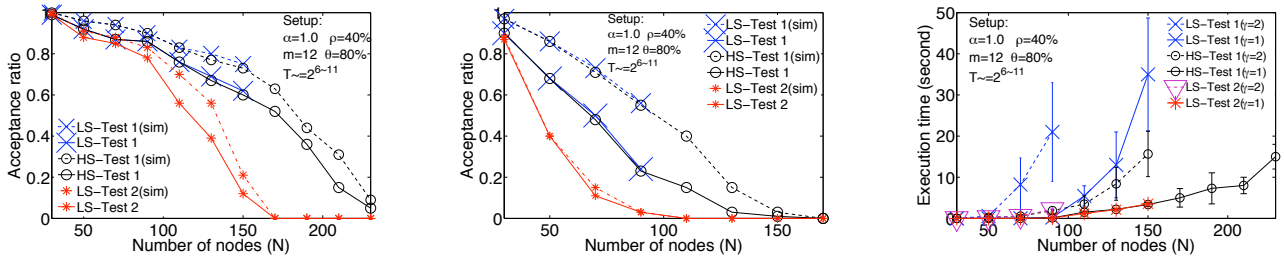

(b) Acceptance ratio when $\gamma = 2$

Fig. 4. Performance under varying number of sources and destinations

## B. Simulations with Random Topologies

We test the scalability of our algorithms on random topologies of different number of nodes ($N$). For every $N$, we generate 100 random networks, each with an edge-density ($\rho$) of 40%, i.e., with $N(N-1)40/200$ edges. PRR of each edge is randomly assigned between 0.80 and 1.0. In this setup, periods are in range $2^{6\sim11}$ slots to accommodate large networks. Here, we consider both the Class-1 schedulability test (Test 1) and the Class-2 schedulability test (Test 2) presented in Section IV. Starting with $N = 30$, we increase $N$ as long as HS can find acceptable assignments and plot the performances in Figure 5.

For $\gamma = 1$ (Figure 5(a)), HS is able to find an acceptable assignment in every case when $N \leq 110$. When $N > 110$, there is a difference from 0.01 to 0.02 between LS and HS in both acceptance ratio and simulation. For $\gamma = 2$ (Figure 5(b)), the difference in both acceptance ratio and simulation is 0.01 to .02 when $N > 50$. The figures also indicate that the acceptance ratio with Test 2 is much lower than that with Test 1. For Test




(a) Acceptance ratio when $\gamma = 1$ (b) Acceptance ratio when $\gamma = 2$ (c) Execution time

Fig. 5. Performance under varying network sizes

2 in this set up, there exists an acceptable assignment when $N \leq 150$ and $\gamma = 1$ (Figure 5(a)), and when $N \leq 90$ and $\gamma = 2$ (Figure 5(b)). This is because Test 2 is a less effective schedulability test compared to Test 1. However, both LS and HS are optimal for Test 2 (and we plot it only for LS).

We abort LS if it cannot complete any test case in 10 minutes. Using Test 1, we have been able to record its performance when $N \leq 150$ for $\gamma = 1$, and $N \leq 90$ for $\gamma = 2$. With $\gamma = 1$, its average execution time remains within 5s when $N \leq 100$, and increases sharply to 36s for $N = 150$ (Figure 5(c)). In contrast, HS takes 14s on average when $N = 230$ and $\gamma = 1$. When $\gamma = 2$, its average execution time is 15s when $N = 150$. If $N > 150$, HS cannot find an acceptable priority assignment for any test case. Using Test 2, the execution time of LS remains less than 5s (as long as there exists an assignment acceptable by Test 2). These results indicate that HS is an effective priority assignment algorithm as it is near-optimal and scales better than LS.

## VIII. Related Works

For transmission scheduling in wireless sensor networks, schedulability analysis has been addressed in previous works [3], [8] considering the fixed priorities as given. For WirelessHART networks, convergecast scheduling for linear and tree [15], [17], [18] topologies, and real-time flow scheduling for arbitrary topologies [14] have been addressed in recent works. The schedulability analysis proposed in [13] assumes that the priorities of flows are given. None of these works addresses the priority assignment for real-time flows. Complementary to these works on scheduling for WirelessHART networks, we propose an optimal and a near-optimal priority assignment algorithm for real-time flows with fixed priorities.

Alur et al. [4] have proposed a mathematical framework to model and analyze schedules using automata for WirelessHART networks. But their formal method approach can be computationally expensive making it suitable only for *offline* design. In contrast, our heuristic search is suitable for *online* admission control and adaptation, which is needed to handle both dynamic workloads and topology changes common in wireless networks. Moreover, our approach is tailored for fixed priority scheduling that is commonly adopted in real-time networks due to its simplicity and efficiency.

## IX. Conclusion

WirelessHART is an important standard for wireless sensor-actuator networks that have increasing adoption in process industries. This paper is the first to address the priority assignment for real-time flows in WirelessHART networks. We have proposed an optimal algorithm based on local search and an efficient heuristic for priority assignment. Simulations on random networks and a sensor network testbed topology showed that the heuristic achieved near-optimal performance in terms of schedulability, while significantly outperforming traditional priority assignment policies for real-time systems.

## Acknowledgement

This research was supported by NSF under grants CNS-0448554 (CAREER), CNS-1017701 (NeTS), and CNS-0708460 (CRI).

## References

[1] http://mobilab.wustl.edu/testbed.
[2] WirelessHART specification, 2007. http://www.hartcomm2.org.
[3] Abdelzaher, T. F., Prabh, S., and Kiran, R. On real-time capacity limits of multihop wireless sensor networks. In *RTSS '04*.
[4] Alur, R., D'Innocenzo, Johansson, Pappas, and Weiss, G. Modeling and analysis of multi-hop control network. In *RTAS '09*.
[5] Audsley., N. C. On priority assignment in fixed priority scheduling. *Information Processing Letters 79*, 1 (2001).
[6] Bertogna, M., Cirinei, M., and Lipari, G. Schedulability analysis of global scheduling algorithms on multiprocessor platforms. *Parallel and Distributed Systems, IEEE Transactions on 20*, 4 (2009), 553 –566.
[7] Chen, D., Nixon, M., and Mok, A. *WirelessHART™ Real-Time Mesh Network for Industrial Automation*. Springer, 2010.
[8] Chipara, O., Lu, C., and Roman, G.-C. Real-time query scheduling for wireless sensor networks. In *RTSS '07*.
[9] Davis, R. I., and Burns, A. Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. In *RTSS '09*.
[10] Davis, R. I., and Burns, A. Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. *Real-Time Syst. 47* (January 2011), 1–40.
[11] Guan, N., Stigge, M., Yi, W., and Yu, G. New response time bounds for fixed priority multiprocessor scheduling. In *RTSS '09*.
[12] Liu, J. W. *Real-time Systems*. Prentice Hall, 2000.
[13] Saifullah, A., Xu, Y., Lu, C., and Chen, Y. End-to-end delay analysis for fixed priority scheduling in WirelessHART networks. In *RTAS '11*.
[14] Saifullah, A., Xu, Y., Lu, C., and Chen, Y. Real-time scheduling for WirelessHART networks. In *RTSS '10*.
[15] Soldati, P., Zhang, H., and Johansson, M. Deadline-constrained transmission scheduling and data evacuation in WirelessHART networks. In *The European Control Conference 2009 (ECC '09)*.
[16] Song, J., Mok, A. K., Chen, D., and Nixon, M. Challenges of wireless control in process industry. In *CRTES '06*.
[17] Zhang, H., Osterlind, F., Soldati, P., Voigt, T., and Johansson, M. Rapid convergecast on commodity hardware: Performance limits and optimal policies. In *SECON '10*.
[18] Zhang, H., Soldati, P., and Johansson, M. Optimal link scheduling and channel assignment for convergecast in linear WirelessHART networks. In *IEEE WiOpt* (Seoul, Korea, Jun 2009).