Mobility-Aware Real-Time Scheduling for Low-Power Wireless Networks

Behnam Dezfouli, Marjan Radi, Octav Chipara
Department of Computer Science, University of Iowa, IA, USA
{behnam-dezfouli, marjan-radi, octav-chipara}@uiowa.edu

***Abstract*—In this paper we consider the problem of supporting real-time communication in mobile networks. To address this challenge, we propose novel transmission scheduling techniques that handle the routing uncertainty introduced by mobility. The core of the scheduling techniques involves controlling the order in which transmissions are scheduled and intelligently scheduling multiple transmissions in a single entry of the scheduling matrix without conflict. Flow-Ordered Mobility-Aware Real-time Scheduling (FO-MARS) integrates these techniques to provide a 14x increase in real-time capacity compared to the baseline algorithms designed for static real-time networks. Additionally, we propose Additive Mobility-Aware Real-time Scheduling (A-MARS), which can handle network dynamics such as the addition or removal of flows without having to reschedule previously admitted flows. As a result, A-MARS achieves significantly lower admission latency than that of the baselines.**

I. Introduction

The adoption of real-time wireless standards such as WirelessHART [1] and ISA100 [2] is making wireless technology an attractive solution for reducing the cost and for simplifying the deployment of process monitoring and control systems [3]–[6]. At the core of these standards are centralized scheduling algorithms that schedule the transmissions of data flows to meet diverse end-to-end deadlines. The real-time systems community has made significant progress in developing scheduling algorithms and associated schedulability analysis [5]–[13]. However, existing works do not support real-time communication in mobile networks. This shortcoming limits the applicability of prior work to applications that include mobile entities such as humans (e.g., patients in a hospital) or robots (e.g., an assembly line) [15], [16].

The goal of a real-time communication protocol is to ensure that the packets of *real-time flows* are delivered within user-defined end-to-end deadlines. Mobility introduces significant network dynamics that are difficult to handle using existing scheduling algorithms. Specifically, real-time algorithms must guarantee that *once a node is admitted to the network, its packets are delivered by their deadlines regardless of the mobility patterns of mobile nodes*. The key challenge to achieving this goal is to ensure that the *scheduling algorithm handles the routing uncertainty introduced by mobility*.

The problem of real-time communication in mobile networks may be approached through using either an *on-demand reservation* approach or an *on-join reservation* approach. In an on-demand reservation approach, a mobile node frequently requests for bandwidth reservation update as routes change due to mobility. In contrast, in an on-join reservation approach, sufficient bandwidth is allocated when the node joins the network to meet its communication requirements irrespective of its movement. While an on-demand approach may (theoretically) provide a higher capacity, two factors make it unsuitable for real-time mobile networks: (1) Low-power networks have short communication ranges. As a result, mobility triggers frequent path updates that would require significant signaling traffic to keep the network's global schedule updated. (2) More importantly, this approach cannot guarantee that the real-time communication requirements of a node will be always met once it joins the network. Specifically, a node's real-time requirements may be violated when the movement of nodes leads to a network configuration where the aggregated network demand exceeds the network capacity. Due to these disadvantages, we will focus on the *on-join reservation* approach.

Existing scheduling algorithms (e.g., [5]–[13]) cannot effectively schedule the transmission of mobile nodes under the on-join reservation approach. These algorithms assume that data flows follow predetermined paths. This assumption is violated in mobile networks. An approach to addressing this limitation is to use one of the existing scheduling algorithms to allocate bandwidth on all communication paths from mobile sources to a base station. However, as shown in the evaluation section, this naive approach leads to a significantly reduced real-time capacity. The central contributions of this paper is the development of scheduling techniques that take advantage of the intrinsic properties of mobility to improve real-time capacity and handle workload dynamics.

In this paper, we use a hierarchical network architecture and we propose novel scheduling techniques to address the problem of delivering real-time traffic from mobile nodes to a base station. The proposed network architecture is composed of mobile nodes and fixed infrastructure nodes. The delivery of packets to the base station is divided into two parts: from the mobile node to an infrastructure node, and from the infrastructure node to the base station. An advantage of this network architecture is that mobility only impacts the first hop delivery of data from mobile nodes to the base station that is dynamically updated in response to mobility. However, mobility has no impact on the routing of data from an infrastructure node to the base station, which is considered to be fixed. In our prior work [14], we proposed the *schedule combination* techniques based on the observation that even though data from a mobile node may be routed over multiple paths, only *one* path is active during a specific interval. In this paper, we extend these techniques by showing that the order (forward vs. reverse) in which transmissions are considered for scheduling can significantly impact the performance of the constructed schedule. Simulations show that reverse scheduling provides significantly higher real-time capacity than forward

scheduling.

We propose *Flow-Ordered Mobility-Aware Real-time Scheduling* (FO-MARS) algorithm which incorporates the mobility-aware scheduling techniques we developed. While FO-MARS results in a significantly improved performance compared to non mobility-aware baseline algorithms, a limitation of FO-MARS is that it does not effectively handle network dynamics. For example, the number of control packet transmissions necessary to perform any workload change scales with the number of mobile nodes. To address this limitation, we propose *Additive Mobility-Aware Real-time Scheduling* (A-MARS). A-MARS is designed to handle network dynamics such as the addition and removal of flows by employing additive transmission scheduling techniques. These techniques consider the real-time demand of future flows to intelligently schedule packet transmissions such that future flows can be scheduled without changing the current schedule. Therefore, A-MARS handles workload changes without having to reconstruct complete schedules. More importantly, A-MARS makes the cost of workload changes independent of the number of admitted mobile nodes because only the schedule of the modified flow should be disseminated.

To perform realistic and repeatable performance measurements we have developed a sophisticated simulator using the packet reception traces provided as part of MoteTrack [17]. We compare the performance of our algorithms against baselines designed for stationary real-time networks as well as their mobility-enhanced versions. Our evaluations show that the MARS algorithms increase the number of admitted mobile nodes by more than 14 times compared to the baselines. In addition, while the admission delay achieved with the baselines is longer than 150 seconds and increases with the number of mobile nodes, the admission delay of A-MARS is always lower than 20 seconds and is independent of the number of mobile nodes. Furthermore, the additive scheduling feature of A-MARS results in a modest (less than 15%) reduction in bandwidth utilization, compared to FO-MARS which requires the full reconstruction of schedules.

II. System Overview

A. Network Model

Our network model is based on WirelessHART [1], [3], which uses a centralized Gateway (GW) to implement a FTDMA scheduling algorithm. Infrastructure nodes ($\mathcal{I}$) form a multi-hop infrastructure through which mobile nodes ($\mathcal{M}$) can communicate with the GW. The fixed infrastructure is deployed to provide coverage within an area (e.g., a building). An *upstream graph* is constructed to route data from every infrastructure node to the GW ensuring that the packet reception rate (PRR) of each link exceeds 95%. A *downstream graph* is also constructed to support data dissemination from GW to other nodes. We assume that both upstream and downstream graphs are spanning trees. Each infrastructure node periodically sends a *report flow* to the GW. This flow is used to maintain the upstream and downstream graphs, as well as delivering mobile nodes' requests for joining the network to the GW. Both the infrastructure and mobile nodes are equipped with half-duplex radios that may transmit on one of available channels ($ch \in \mathcal{C}$).

B. Flows, Transmissions and Schedules

We adopt real-time flows as a communication primitive. A flow i is characterized by its period P_i and deadline D_i ($D_i \leq P_i$). An instance $J_{i,k}$ of flow i is released each period k at time $r_{i,k} = k \times P_i$, where $k \in \mathbb{N}$. The absolute deadline of $J_{i,k}$ is $d_{i,k} = r_{i,k} + D_i - 1$. Flows are assigned static priorities according to a deadline monotonic policy. We use the notation $\Pi^i_{M,X}$ to denote the path used to route data from a mobile node M to the GW using an infrastructure node X for flow i[1]. Similarly, the notation $(AB)_i$ represents a transmission (AB) pertaining to flow i. The scheduler constructs a global scheduling matrix $\mathcal{S}$. Each transmission is assigned a time slot and a channel. We refer to a time slot and a channel pair as an *entry* because it can identify an entry of the scheduling matrix.

Transmissions must be scheduled such that the following constraints are satisfied: (1) A node transmits or receives only once in a time slot, as radios are half-duplex. (2) In each time slot, WirelessHART requires that at most one transmission should happen in each channel. (3) The hop-by-hop forwarding of packets introduces precedence constraints such that a sender must receive a packet before forwarding it to the destination.

C. Admitting a Mobile Node to the Network

1) Beaconing: Beacon broadcast allows the mobile nodes to discover nearby infrastructure nodes.

2) Join Request: When a mobile node intends to join the network, it sends a *join request*, including information about the data flows the node will generate. Periodically, in a time slot, all the infrastructure nodes listen for receiving join request packets. Mobile nodes intending to join the network employ slotted CSMA for channel access in this slot.

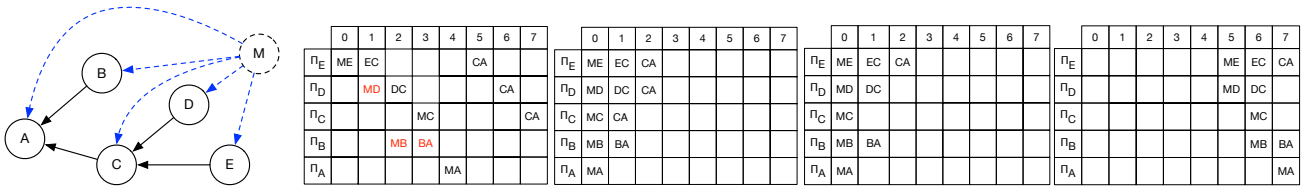
3) Schedule Reception: When an infrastructure node receives a join request, it includes that request within the reporting data sent to the GW. When the GW receives a join request, it should reserve bandwidth for the data flows of the mobile node. After schedule computation, the GW disseminates the new schedule through control packets. However, in addition to the schedule computed for a new mobile node, the GW may also need to disseminate the schedules of other nodes if their schedules have been modified. When an infrastructure node receives scheduling information in response to a join request, it should broadcast that data in its next beacon.

III. Real-time Scheduling in Mobile Networks

In this section, we start by formalizing the problem of real-time scheduling in the presence of mobility. Next, we propose scheduling techniques that take advantage of the structure of the problem to improve real-time capacity. These techniques will be used as part of the algorithms proposed in Section IV.

Let us consider the problem of delivering packets from a mobile node M to the GW for a single instance of a flow i. We assume $P_i = D_i = 8$ slots. The delivery of packets is divided into two parts: (1) single-hop communication from the mobile node M to an associated infrastructure node and, (2) potentially multi-hop communication, from the associated infrastructure to the GW. Since we adopt an on-join approach,

[1]We will omit i and M from the notation when the flow and/or the mobile we are considering are clear from the context.



(a) Topology (b) Naive Schedule (c) Flow Merge (d) Flow Coordination (e) Reverse Schedule

Fig. 1. Example topology and associated schedules. The solid circles indicate infrastructure nodes, and A is the GW. The solid edges form the upstream routing tree. The dashed circle represents the mobile node. The dashed edges represent the potential associations of the mobile node. Black and gray (red) transmissions are scheduled on channels 0 and 1, respectively.

we must ensure that we allocate sufficient capacity for mobile node M to transmit its packets regardless of its mobility pattern (and that of other nodes). Accordingly, in the general case, M may deliver its packets to *any* of the infrastructure nodes depending on its location. We formalize this problem by constructing an *Augmented Communication Graph* (ACG) that captures the workload introduced by the mobile nodes. The ACG is constructed by starting from the upstream spanning tree and then adding an edge from a mobile node to each infrastructure node. Figure 1a shows the ACG of an example topology with infrastructure nodes A, B, C, D, and E, and a single mobile node M. The notation $children(A)$ denotes the children of A in the ACG. Similarly, the depth of a link (AB), denoted as $depth(AB)$, is the depth of B in the ACG.

A naive scheduler that is not mobility aware would schedule M's transmissions over the five possible paths (Π_A, Π_B, Π_C, Π_D, and Π_E) *independently*. The scheduler maintains a set of *ready* transmissions which includes all the transmissions that can be considered for scheduling. Initially, only the transmissions from the mobile node are ready. The scheduler iteratively considers each transmission in the ready set and attempts to schedule it in the current slot. Additionally, amongst the ready transmissions, higher priority is given to a transmission with higher depth. Transmissions are assigned to channels such that: (1) a single transmission is performed per channel, and (2) a node transmits/receives at most once in a slot. If a transmission is scheduled, then it is removed from the ready set, and the next transmission of that flow is added to the ready set to be considered for scheduling starting from the next time slot. Note that the definition of the ready transmission enforces the hop-by-hop forwarding constraints. The scheduling matrix computed by the naive scheduler is shown in Figure 1b. The matrix uses 8 slots and 2 channels to schedule M's flow.

The naive scheduler is the state-of-the-art scheduling for static real-time networks. However, the real-time capacity of the network may be increased by observing that even though there are multiple routes that M's packets may follow, *only one* path is active at a time. Therefore, packet transmission from different paths may be scheduled in the same slot and channel, without violating the scheduling constraints.

Theorem 1 (Flow Merging)**.** *For a flow i, transmissions $(AB)_i$ and $(CD)_i$ on two paths $(AB) \in \Pi^i_{M,X}$ and $(CD) \in \Pi^i_{M,Y}$ $(X \neq Y)$ may be scheduled in the same entry.*

Proof: Refer to our technical report [18]. ■

The naive scheduler modified with Theorem 1 has significantly higher capacity. The constructed scheduling matrix requires only three slots and uses a single channel, as Figure 1c shows.

A close inspection of the scheduling matrix shows, however, that there are some sources of inefficiency. For example, link CA is scheduled in three entries even though it actually transmits at most one packet during a period of the flow generated by M (see Figure 1c). The reason for this inefficiency is that the scheduling of the five paths is not coordinated. We can impose the constraint that an infrastructure node cannot transmit until its children's have been scheduled. The following rule further reduces the number of entries required to schedule the flow instance, as shown in Figure 1d.

Rule 1 (Flow Coordination)**.** *For a flow i, a transmission $(AB)_i$ must be scheduled once after transmissions $(CA)_i$, where $C \in children(A)$, have been scheduled.*

The real-time capacity of the network also depends on the order in which transmissions are considered for scheduling. Thus far, we have considered transmissions for scheduling in a *forward* manner, meaning that: (1) transmissions are added to the set of ready transmissions starting from the mobile node and, (2) scheduling an instance $J_{i,k}$ is started in slot $r_{i,k}$. Unfortunately, the forward scheduling approach yields suboptimal results because it limits the reuse of infrastructure nodes for scheduling other flows. A node is *blocked* in a slot when it is scheduled to receive or transmit for a flow. An infrastructure node blocked in a slot cannot be reused to schedule other flows on any channel of that slot. For example, in Figure 1d, node A is blocked in slots $0, 1$, and 2.

We propose two approaches to reduce the blocking of infrastructure nodes. First, when considering a transmission $(AB)_i$, we prefer scheduling the transmission in an entry in which either A or B is already scheduled for transmitting or receiving flow i. It is easy to see that adding $(AB)_i$ to such an entry does not increase the number of blocked nodes. An additional approach to reducing blocking is to schedule transmissions in reverse order starting from the deadline towards the release time. This ordering implies the construction of a *reverse ready* set that initially includes the transmissions that deliver a flow to the GW. In addition, scheduling an instance $J_{i,k}$ is started from time slot $d_{i,k}$. When a transmission $(AB)_i$ is scheduled in a slot s, the incoming links to A (from its children) become reverse ready and are considered for scheduling in the next time slot (i.e., $s - 1$). We prioritize transmissions in the reverse ready set according to their depth, and links with smaller depth are given higher priority.

Rule 2 (Reverse Scheduling)**.** *In reverse order, a transmission $(AB)_i$ can be considered for scheduling only after*

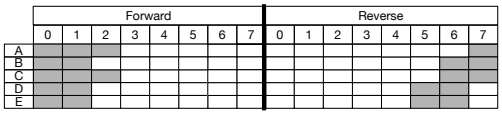


Fig. 2. The opportunities to schedule transmissions of other flows on infrastructure nodes. The gray entries are blocked.

all transmissions $(CA)_i$*, where* $C = children(A)$*, have been scheduled. Transmissions are prioritized according to the receiver's depth.*

The schedule obtained using Rule 2 is shown in Figure 1e. Note that the schedule uses the same number of entries as the forward scheduling strategy. The difference between the two cases becomes apparent when we consider the opportunities for reusing the infrastructure nodes for other flows. As evident in Figure 2, the reverse scheduling approach provides several opportunities to schedule other flows concurrently with the current one with the net effect of increasing real-time capacity. For example, while node A is blocked in three slots when forward scheduling is used, this node is blocked in only one slot when reverse scheduling is employed.

Theorem 2 (Reverse Scheduling Optimality)**.** *Assume that we are interested in scheduling a single flow in the network. When scheduling packets in reverse scheduling order, an infrastructure node is scheduled to receive in a single time slot, which is optimal.*

Proof: Refer to our technical report [18]. ■

IV. Scheduling Algorithms

In this section we introduce two scheduling algorithms that benefit from the techniques introduced in the previous section. The first algorithm employs the previously developed insights to effectively schedule real-time flows from mobile nodes. A limitation of the algorithm is that workload changes require recomputing the entire scheduling matrix. The second algorithm may handle such dynamics without having to reconstruct the entire matrix. However, this comes at the cost of slightly lower real-time capacity.

A. FO-MARS

Flow-Ordered Mobility-Aware Real-time Scheduling (FO-MARS) is a mobility-aware FTDMA algorithm that uses *flow merging* (Theorem 1), *flow coordination* (Rule 1), and *reverse scheduling* (Rule 2) to improve real-time capacity. Algorithm 1 shows the pseudo-code. At a high-level, FO-MARS constructs a scheduling matrix $\mathcal{S}$ that is repeated periodically. The number of slots of $\mathcal{S}$ is equal to the hyper-period of the flows ($H = \text{lcm}\{P_i | \forall i \in \mathcal{F}\}$, where $\mathcal{F}$ is the set of flows), since the arrival pattern of periodic flows repeats every hyper-period.

FO-MARS determines the entries in the scheduling matrix in a flow-oriented manner by considering each flow in increasing deadline order and scheduling each instance $J_{i,k}$ of the flow that is released during the hyper-period. FO-MARS maintains a reverse ready list (denoted by $rready$) that includes the transmissions that are ready for scheduling. Consistent with our reverse scheduling approach, the scheduling of $J_{i,k}$ is performed by scheduling transmissions from its deadline ($d_{i,k}$) towards its release time ($r_{i,k}$). Initially, the reverse ready list includes all the transmissions that the GW is their destination,

Algorithm 1: The FO-MARS algorithm

Input: $\mathcal{F}$: set of the flows that should be scheduled.
Output: returns scheduling matrix $\mathcal{S}$ when flows are scheduled successfully; returns "unsuccessful" otherwise.

```
1  Procedure FO-MARS
2    Let H be the hyper-period (H = lcm{P_i | ∀i ∈ F})
3    for i in F sorted by deadline do
4      Let M be the mobile node generating flow i
5      for J_{i,k} in released(i, H) do
6        rready = { (AB)_i | (B is GW) and (A ∈ I or A = M ) }
7        issched = False
8        for slot in d_{i,k} : r_{i,k} : −1 do
9          scheduled = ∅
10         for (AB)_i in rready sorted by depth(AB) do
11           channel = may-schedule((AB)_i, slot, S)
12           if channel ≠ None then
13             S[slot, channel] = S[slot, channel] ∪ {(AB)_i}
14             scheduled = scheduled ∪ {(AB)_i}
15         for (AB)_i in scheduled do
16           rready = rready \ {(AB)_i}
17           rready = rready ∪ {(CA)_i | C ∈ children(A)
18                                 and (C ∈ I or C = M) }
19         if rready = ∅ then
20           issched = True
21           break
22       if issched = False then return "unsuccessful"
23   return S

24 Procedure may-schedule((AB)_i, slot, S)
25   Let T include all transmissions S[slot, ch] where ∀ch ∈ C
     /* Half-duplex constraint                                  */
26   for (CD)_j in T do
27     if i ≠ j then
28       if C = A or C = B or D = A or D = B then
29         return None
     /* A channel having (∗B)_i or (A∗)_i                       */
30   for ch in C do
31     if ∃C s.t. (CB)_i ∈ S[slot, ch] or (AC)_i ∈ S[slot, ch] then
32       return ch
     /* A channel with a transmissions for flow i               */
33   for ch in C do
34     if ch includes a transmission (CD)_i in slot then
35       return ch
     /* Return an empty channel                                 */
36   for ch in C do
37     if ch is not used in slot then
38       return ch
39   return None
```

including the transmission from the mobile node to the GW (see line 6). FO-MARS considers each link $(AB)_i$ in the reverse ready set in the order of their depth. The algorithm determines whether there is a suitable channel in which $(AB)_i$ may be scheduled in the current time slot. If such a channel may be identified using the **may-schedule** procedure, then link $(AB)_i$ is added to the scheduled set (denoted by $scheduled$). When all transmissions in set $rready$ have been considered, the reverse ready set is updated by removing all the links that have been scheduled during the current slot and adding all their children to the set (see lines 16 – 18).

The **may-schedule** procedure determines the channel in which the transmissions can be scheduled. The procedure is designed to: (1) ensure that the scheduling constraints are satisfied, and (2) increase real-time capacity. We improve real-time capacity by taking advantage of the insights developed in Section III. Accordingly, we prefer to schedule $(AB)_i$ with other transmissions from flow i that include either A or B (see line 32). This does not increase the blocking of any node. When this is not possible, we prefer to schedule $(AB)_i$ with other transmissions from flow i since only one of the transmissions scheduled in the entry will be performed at run-time (see line 35). When no transmissions from i are included in the current slot, then we return the first empty channel (if one exists) (see line 38). The **may-schedule** procedure returns 'None' when scheduling in the considered slot is not possible.

When the GW disseminated a newly computed schedule, the new mobile node cannot immediately start communicating with the GW upon schedule reception. Specifically, infrastructure nodes cannot arbitrarily switch to a new schedule if the scheduling algorithm modifies the existing schedules. For example, assume a transmission $(AB)_i$ happens after time slot s in the current scheduling matrix $\mathcal{S}$. However, in the new scheduling matrix $\mathcal{S}'$, this transmission happens before time slot s. If node A receives the new scheduling matrix at time s, transmission $(AB)_i$ will not happen in the current period of flow i if A immediately switches to the new schedule. To avoid packet loss, the safe switching time is at the beginning of the next hyper-period. However, if new bandwidth reservation could be performed without modifying the existing schedules, then the mobile node can start packet generation as soon as the new schedule is received. The next section proposes an algorithm that addresses such workload dynamics, and its node admission delay is independent of the number of mobile nodes and hyper-period duration.

An additional optimization is possible when the periods of the flows are harmonic (i.e., multiples of each other). Owing to the regularity of the arrival pattern, the resulting scheduling matrix is also regular and it may be "compressed" to require fewer transmissions to be disseminated. Consider transmission $(AB)_i$ of flow i will be scheduled for $J_{i,0}$ in slot s_0. Then, $(AB)_i$ will be scheduled for instance $J_{i,k}$ in slot $s_k = s_0 + k \times P_i$, where $k \in \mathbb{N}^*$. This property implies that transmissions are scheduled with the same relative phase in the scheduling matrix and, knowing the schedule of first instance of each flow is sufficient to reconstruct the complete scheduling matrix.

B. A-MARS

Traditional scheduling approaches (including FO-MARS) must recompute and disseminate complete schedules to handle networking dynamics including addition, removal, or changes in flows. In this section, we develop *Additive Mobility-Aware Real-time Scheduling* (A-MARS), an additive scheduling algorithm that schedules incoming flows without modifying the schedule of previously admitted flows. The key challenge in developing this algorithm is to ensure that scheduling a new flow causes minimum impact on the schedulability of future flows. In other words, the effect of scheduling a flow i on flows with higher priority (i.e., shorter deadline) should be minimal.

A-MARS is developed under the assumption that the network will service a set of flow classes ($\bar{\mathcal{F}}$) that are known a priori. A flow class includes flows that have the same period and deadline. Mobile nodes may add or remove flows at arbitrary times, however, the likelihood of a flow belonging to a class γ is a_γ, and $\sum_{\forall \gamma \in \bar{\mathcal{F}}} a_\gamma = 1$.

At a high-level, A-MARS works in two phases: (1) initialization phase, and (2) scheduling phase. The initialization phase prepares an *ordered slot list* L_γ for each flow class γ. This list indicates in what order the scheduling algorithm should consider the slots that can be used for scheduling a flow belonging to class γ. For an instance $J_{\gamma,k}$, the set of slots that can be considered are those between its release time ($r_{\gamma,k}$) and its deadline ($d_{\gamma,k}$). Therefore, for scheduling all the instances of γ during the hyper-period, we must consider the union of the intervals $\overline{r_{\gamma,k},\ d_{\gamma,k}}$, where $k \in \overline{0, H/P_\gamma}$. The responsibility of the initialization phase is to order these slots as a list L_γ, so that the schedulability of higher priority flows is not compromised. After the initialization phase, the scheduling phase uses a modified version of FO-MARS to schedule packet transmissions based on the prepared ordered slots lists.

The core idea behind the initialization phase of A-MARS is to estimate the impact that scheduling a flow can have on its higher priority flows. For example, assume γ is the flow class for which L_γ is being prepared. To prepare this list, the slot preparation algorithm (Algorithm 2) iterates over the slots in list $slots$, and in each iteration, one slot is added to L_γ. To measure how choosing a slot s for scheduling γ (in addition to the slots already in L_γ) would affect the schedulability of a higher-priority class α, we define the *potential utilization* as:

$$PU_\alpha[s, L_\gamma] = \begin{cases} \frac{a_\alpha \times W_\alpha}{D_\alpha - |L_\gamma \cap \overline{r_{\alpha,k},\ d_{\alpha,k}}|} & \text{if } \exists k \in \overline{0, H/P_\alpha} \text{ s.t.} \\ & s \in \overline{r_{\alpha,k},\ d_{\alpha,k}} \setminus L_\gamma \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where W_α is the number of transmissions required to schedule a flow belonging to class α, D_α is the deadline of flow class α, and L is the set of slots that have been already prioritized to be used by flow class γ. The value of $PU_\alpha[s]$ represents the (unnormalized) probability that slots s will be required by $J_{\alpha,k}$ when some slots are used by a lower priority flow class. The numerator estimates the required number of slots for $J_{\alpha,k}$, while the denominator estimates the number of free slots that may be used to perform these transmissions.

We now present the operation of the slot preparation algorithm (Algorithm 2). Assume we are interested in preparing L_γ. First, a PU list is prepared for every flow class in set HP, which includes the flow classes with higher priority (see line 7). These PU vectors are updated after each iteration in which a new slot is added to L_γ. Initially, the list $slots$ includes all the slots that should be prioritized, and each iteration of the while loop moves one slot from $slots$ to L_γ. In each iteration, amongst the set of slots in $slots$, we should find the slot that results in minimum increase of the PU vectors. For a slot s, increase in the PU of a flow class is measured through: (1) creating a temporary ordered slots list L'_α that includes the slots already ordered as well as slot s (see line

Algorithm 2: AMARS's Slot Preparation Algorithm

Input: $\overline{\mathcal{F}}$: set of flow classes;
Output: an ordered slot list L_γ for each flow class $\gamma \in \overline{\mathcal{F}}$;

1 **Procedure AMARS_SlotPrep()**
2 Let HP be the set of flow classes with higher priority than γ
3 $L_\gamma = \emptyset$
4 $slots = \bigcup_{k=0}^{H/P_\gamma} \overline{r_{\gamma,k},\ d_{\gamma,k}}$
5 **while** $slots \neq \emptyset$ **do**
6 **for** flow class α in HP **do**
7 PU_α = **computePU**(α, L_γ, $slots$)
8 **for** s in $slots$ **do**
9 $L'_\gamma = L_\gamma \cup \{s\}$
10 **for** α in HP **do**
11 PU'_α = **computePU**(α, L'_γ, $slots$)
12 $ADPU[s]$ += $\sum_{s' \in slots \setminus s} (\mathrm{PU}'_\alpha[s'] - \mathrm{PU}_\alpha[s'])$
13 $s_{best} = \mathrm{argmin}_{s \in slots} ADPU[s]$
14 $L_\gamma = L_\gamma \cup \{s_{best}\}$
15 $slots = slots \setminus \{s_{best}\}$
16 **Procedure computePU**(α, L_γ, $slots$)
17 **for** s in $slots$ **do** compute $PU_\alpha[s, L_\gamma]$ using Equation 1
18 **return** PU_α

9), (2) computing a new PU vector considering list L'_α for each flow class α (see line 11), (3) subtraction of PU from PU' over all the slots except s, which reflects the increase in the PU value of the slots (see line 12), and (4) addition of the differential values (see line 12). When considered for all the PU vectors, we refer to the sum of the increases in the PU vectors as the *accumulated differential potential utilization* ($ADPU$). In each iteration, when all the slots in $slots$ have been evaluated, the slot that results in minimum ADPU is identified as s_{best} (see line 13) and this slot is moved from $slots$ to L_γ. Additionally, the PU vectors are updated at the beginning of the next iteration considering slot s_{best} marked as prioritized (see line 14 and 7).

We clarify the operation of A-MARS through the sample scenario given in Figure 3. There are three flow classes: γ, β, and γ. We are interested in computing L_γ for class γ. This figure includes the first three iterations of Algorithm 2. When $L_\gamma = \emptyset$, slot 15 results in minimum ADPU. In other words, if we consider slot 15, the increase in PU values is 0. Intuitively this is expected since slot 15 cannot be used to schedule transmissions for either α or β. This slot is moved from $slots$ to L_γ. In the second iteration, both slot 7 and 23 result in minimum ADPU, which is 0.1. The tie is broken randomly and slot 23 is chosen. Following the same operation, slot 7 is found as the best slot after the third iteration. The last row of Figure 3 shows the iteration number in which a slot is prioritized and added to L_γ. Therefore, this list reflects our priority to use the slots in range $[0, 31]$ for scheduling flows of class γ.

FO-MARS may be modified to account for scheduling each flow through using the ordered slots list prepared for that flow's class. The original FO-MARS algorithm considers the slots from $d_{i,k}$ to $r_{i,k}$ when scheduling the transmissions of a flow instance $J_{i,k}$ (see line 8 of Algorithm 1). However, when modified for operation with ordered slots lists, the algorithm should try to schedule a flow belonging to class γ using the slot priorities provided in L_γ. To this end, the algorithm uses the first slot of L_γ and tries scheduling. If scheduling is failed, the algorithm uses the first two entries of L_γ. It should be noted that since reverse scheduling is employed, the algorithm considers these slots in a descending order of slot numbers. This process is repeated until the scheduling succeeds, or the scheduling is failed after using all the slots in L_γ. Considering the priorities achieved in Figure 3, scheduling is first evaluated using slot 15. If scheduling is failed, slot 23 and 15 are considered. Similarly, slot 23, 15, and 7 are considered in the third scheduling try.

C. Handling Workload Dynamics: FO-MARS vs A-MARS

In our network model, workload dynamics are the result of flows being added, removed, or their parameters modified. FO-MARS schedules flows according to static priorities. Therefore, the addition of a new flow results in FO-MARS computing the schedule of the new flow as well as recomputing the schedule of flows with lower priority than the considered flow. Accordingly, the amount of control traffic that must be disseminated is $O(|\mathcal{F}| \cdot \Psi(|E|))$, where $\Psi(|E|)$ is an efficient encoding of transmissions over network edges ($|E|$). In sharp contrast, A-MARS handles workload changes more efficiently, and the addition of a flow requires that we only compute the schedule of that flow. Accordingly, A-MARS handles a workload change in $O(\Psi(E))$, which provides significantly higher efficiency.

V. Performance Evaluation

In order to have a realistic and repeatable performance evaluation, we used the packet reception traces of MoteTrack [17]. Using MicaZ nodes with CC2420 radio, the trace provides packet reception characteristics from 23 infrastructure nodes on various mobility paths. The mobility model is consistent with users moving along the hall-ways of the building (see Figure 4). The initial position of each mobile node is randomly selected on one of the mobility paths. The mobile nodes move at constant speed until reaching the intersection of two paths. At that point, either a new path is randomly selected or the current path is continued. The moving direction is reversed when a node reaches the end of a path. A routing graph is created using an approach similar to that described in [19].

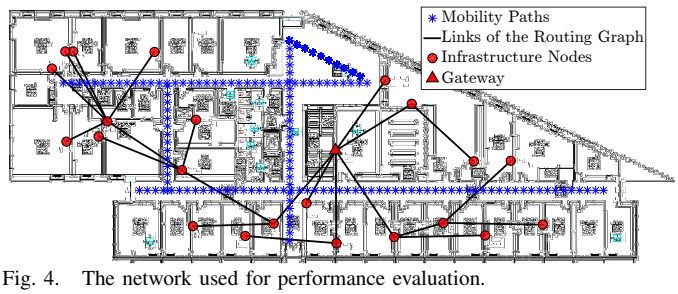


Fig. 4. The network used for performance evaluation.

Each infrastructure node broadcasts a beacon every 512 slots. A request reception slot is scheduled every 512 slots to receive join requests. Similarly, the period of control and report flows are 512 slots (see Section II-A). We consider two types

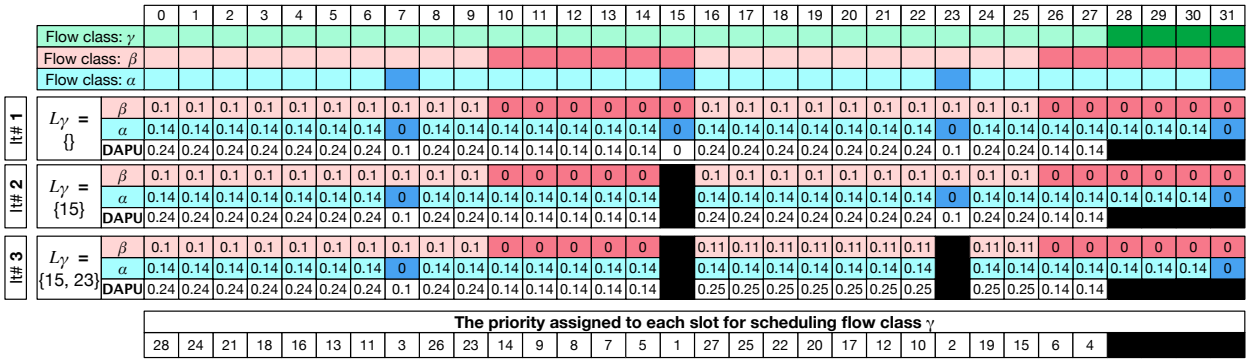


Fig. 3. A network with three flow classes α, β, and γ. For these classes: $P_\gamma = 32$, $D_\gamma = 28$, $P_\beta = 16$, $D_\beta = 10$, $P_\alpha = 8$, and $D_\alpha = 7$. We are interest in preparing L_γ. The calculations assume that flows arrive with equal probability (i.e., $a_\alpha = a_\beta = a_\gamma = 1$), and their workload is $W_\alpha = W_\beta = W_\gamma = 1$. The figure includes the first three iterations (denoted by 'It#') of Algorithm 2. The last row shows the iteration number in which each slot is added to L_γ.

TABLE I. BASELINE ALGORITHMS

Static Real-Time Scheduling (*-SRS)
Algorithms designed for static real-time networks. These algorithms do not benefit from Theorem 1, Rule 1, and Rule 2. These algorithms are as follows: *Earliest Deadline First-SRS* (EDF-SRS): The schedulability of released transmissions is evaluated in the order of their absolute deadlines. *Deadline Monotonic-SRS* (DM-SRS): The schedulability of released transmissions is evaluated in the order of their relative deadlines. *Least Laxity First-SRS* (LLF-SRS): The schedulability of released transmissions is evaluated in the order of their laxities. Laxity of a transmission in a given time slot is computed as $d' - h$, where d' is the remaining number of time slots until deadline and h is the number of hops to the destination.
Enhanced Static Real-Time Scheduling (*-ESRS)
EDF/DM/LLF-ESRS: Adds Rule 1 to EDF/DM/LLF-SRS
Combination-Enabled Real-Time Scheduling (*-CERS)
EDF/DM/LLF-CERS: Adds Theorem 1 and Rule 1 to EDF/DM/LLF-ESRS

TABLE II. GENERAL PERFORMANCE EVALUATION PARAMETERS

Time Slot Duration = 10ms (based on WirelessHART and ISA100)
Packet Format: 802.15.4 | Max Packet: 127B | Max Payload: 108B
Radio Channels: 11-26 | Radio Transmission Power = 0dBm
Battery: 2500mAh 3V | Speed: 1m/s | Guard Time (t_g) = 1ms

of workloads: (1) *Homogeneous*: all the mobile nodes' flows belong to the same class. For example, $P_{data} = 128$ means the period of all the mobile nodes' flows equals 128 slots (1.28 second). (2) *Heterogeneous*: each mobile node randomly chooses its flow from a set of flow classes. To measure energy consumption, we have implemented the radio state machine and energy consumption characteristics of CC2420 at the physical layer [20] . Our MAC implementation considers time synchronization errors through using a $1ms$ guard time (t_g) at the beginning of each time slot. In a reception time slot, a node waits for $2t_g + 160\mu s$, and goes to the sleep mode if no packet is detected. Note that $160\mu s$ is the preamble+SFD transmission duration [21].

The baseline algorithms compared against FO-MARS and A-MARS are given in Table I. We repeated the experiment 20 times for each configuration, and report the median, lower quartile and higher quartile. The parameters used for simulation are given in Table II.

A. Results and Discussions

1) Scalability: To measure real-time capacity, Figure 5 presents the maximum number of mobile nodes admitted by various scheduling algorithms. The figure shows the very poor performance of the algorithms designed for static wireless networks (see the SRS algorithms in Table I), which is due to the lack of flow merging, flow coordination, and reverse scheduling. The results show that using Theorem 1 and Rule 1 in CERS algorithms increase the number of admitted mobile nodes by more than 6x, compared with SRS algorithms. Additionally, the results reveal the effect of reverse scheduling on the efficiency of schedule combination. Specifically, the MARS algorithms increase the number of admitted mobile nodes by 2.5x, compared to the CERS algorithms. Although A-MARS relies on a predictive strategy to schedule flows additively, Figure 5 shows that A-MARS does not sacrifice bandwidth utilization. While the bandwidth reservation efficiency of A-MARS is as good as FO-MARS when the workload is homogeneous, it introduces less than 15% reduction in the number of admitted mobile nodes when the workload is heterogeneous.

2) Admission Delay: Figure 6 shows admission delay, defined as the interval between a mobile node's turn on time and the time at which it can start communicating with the GW. The admission delay achieved with A-MARS is independent of the number of mobile nodes and flow periods (always < 20 seconds), thanks to its additive scheduling feature. The admission delay of FO-MARS is the same as that of A-MARS when flow periods are homogeneous (see Figure 6 (a)-(b)) because, the request for scheduling a data flow only requires the scheduling of that flow. However, for the heterogeneous workload, the admission delay of FO-MARS increases with the number of admitted mobile nodes as an increasing amount of control traffic is necessary to disseminate the recomputed scheduling matrix. The admission delay of all the baseline algorithms are significantly higher than that of the MARS algorithms because they necessarily need to reschedule all the flows when a join request is received. Meanwhile, the admission delay of SRS algorithms is always higher than 150 seconds (see Figure 6 (b) and (d)). Furthermore, the admission delay of the baselines depends on the relationship between flow periods. For example, comparing Figure 6(a) and (b) reveals the shorter admission delay achieved with smaller periods. The reason is that, for example, when $P_{data} = 128$, the GW should distribute the schedule of each mobile node four times $(512/128)$, while this reduces to one when $P_{data} = 512$.

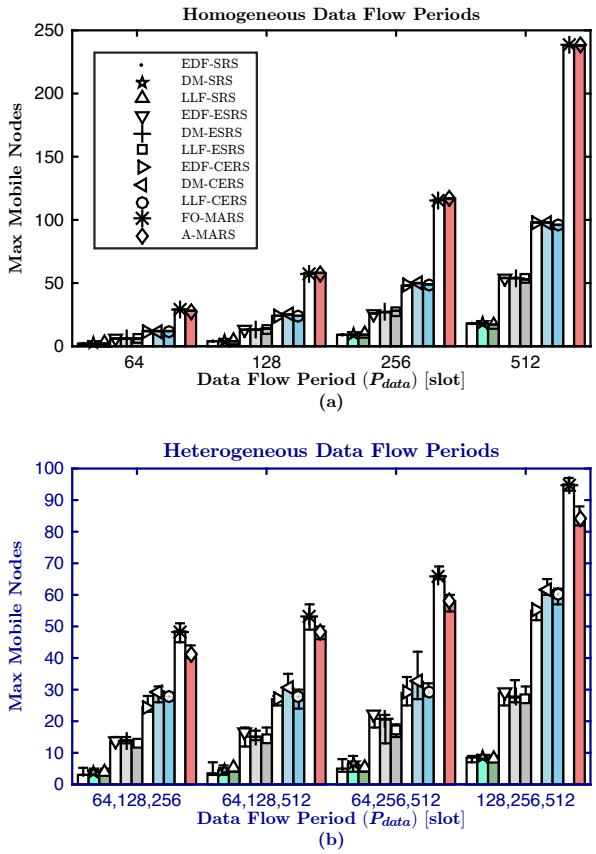


Fig. 5. Maximum number of admitted mobile nodes with homogeneous workload (a), and heterogeneous workload (b).

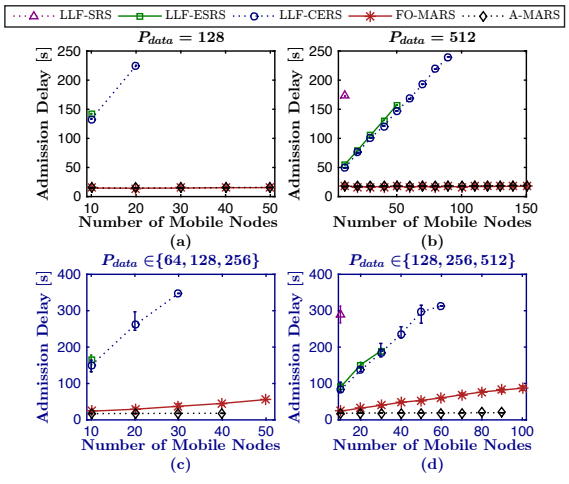


Fig. 6. Admission delay achieved with various scheduling algorithms considering homogeneous workload (a)-(b) and heterogeneous workload (c)-(d).

3) Lifetime: Figure 7 presents the average lifetime of infrastructure nodes versus the number of active mobile nodes. We do not report the lifetime of mobile nodes because it was higher than that of infrastructure nodes. We increase the number of mobile nodes in steps of 10 and measure steady-state energy consumption. Although with a given P_{data} and a number of mobile nodes the time spent in transmit mode is the same for all the algorithms, the lifetime differences are due to the different durations the radio spends in receive mode. From the MAC point of view, flow merging (Theorem 1) and flow coordination (Rule 1) reduce the number of time slots in which a node expects to receive a packet. Additionally, this number is further reduced using the reverse scheduling strategy (Rule 2). For example in Figure 2, the number of slots in which A should wait for receiving a packet belonging to the flow generated by M is 3 using forward scheduling, and this value is reduced to 1 using reverse scheduling. Also note that compared to FO-MARS, the additivity feature of A-MARS contributes to energy saving through reducing the amount of schedule dissemination data.

4) Algorithm Execution Duration: Figure 8 shows algorithm execution duration versus the number of mobile nodes admitted using an i7-4980HQ processor. The execution duration of baseline algorithms (i.e., ESRS and CERS) are considerably higher than that of MARS algorithms because they need to schedule all the flows in all of their periods within the hyper-period. The higher execution duration of A-MARS compared to FO-MARS is due to its iterative evaluation of schedulability using a subset of the slots in a prepared ordered slots lists. Additionally, as the number of scheduling tries depends on the flow arrival pattern, A-MARS shows high variations.

VI. Related Work

Although mobility support in low-power wireless networks have been addressed by various works [22]–[25], unfortunately, these approaches cannot guarantee real-time communication with mobile nodes, and this is mainly due to the lack of employing bandwidth reservation. More importantly, they do not impose any admission control mechanism for joining mobile nodes. Therefore, both reliability and delay depend on medium access intensity, which is mainly determined by mobility pattern and number of mobile nodes.

Several scheduling mechanisms have been recently proposed for enabling low-power real-time communication [5]–[13]. In [5] and [13], transmission schedules are determined based on the number of nodes and network topology, before network deployment. [6] and [12] address scheduling over multiple paths between stationary nodes. The development of a real-time communication system for a refinery with stationary nodes has been presented in [11]. A laxity-based heuristic scheduling algorithm has been proposed by [8] for reducing schedule computation delay in static WirelessHART networks. Unfortunately, none of these algorithms provide efficient bandwidth reservation when mobile nodes are introduced to the network. In this paper we referred to the prior work on real-time scheduling in static networks as *Static Real-time Scheduling* (SRS) algorithms (see Table I).

MBStar [26] aims to reduce the cost of schedule distribution to one-hop sensor nodes. Using offset-free scheduling, the data generation offsets are determined to reduce the number of collisions. Similar to MBStar, RT-WiFi [27] only addresses mobility over a single hop, and eliminates the need for dynamic association.

In our prior work [14] we identified several sources of scheduling inefficiency, and we proposed the schedule combination theorem for improving real-time capacity. Based on

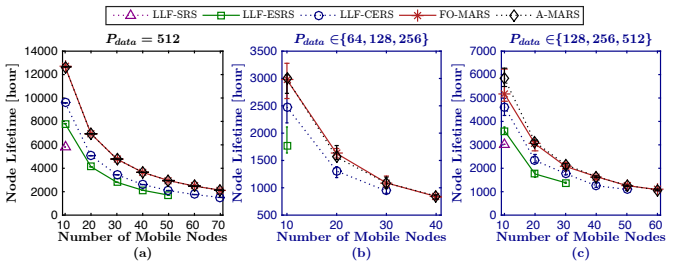


Fig. 7. Average lifetime of an infrastructure node versus the number of active mobile nodes.

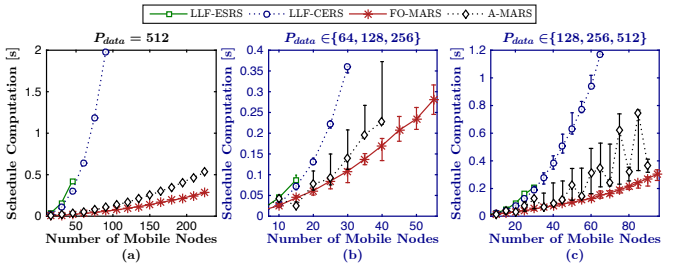


Fig. 8. Algorithm execution duration.

that, we proposed a laxity-based scheduling algorithm that its operation is similar to that of the LLF-CERS algorithm considered in this paper as a baseline. In this paper, we have improved real-time network capacity through proposing the reverse scheduling strategy that is the basis of the FO-MARS algorithm. More importantly, we proposed A-MARS which incorporates a novel approach for handling workload dynamics.

VII. Conclusion

This paper proposed two algorithms to address the development challenges of mobile real-time networks: FO-MARS, and A-MARS. Both algorithms rely on a set of techniques to improve the efficiency of on-join bandwidth reservation for mobile nodes. Specifically, we presented flow merging, flow coordination, and reverse scheduling, as the essential techniques for improving the efficiency of bandwidth reservation for mobile nodes. Furthermore, the salient feature of A-MARS is additive scheduling, which allows A-MARS to handle network dynamics without re-computing the scheduling matrix. As a result, the admission delay of flows is independent of the number of mobile nodes. Both MARS algorithms significantly increase the number of admitted mobile nodes by 14x, compared to several baseline algorithms.

VIII. Acknowledgment

This work was supported by the National Science Foundation (Grant No. 1144664) and by the Roy J. Carver Charitable Trust (Grant No. 14-4355).

References

[1] "HART Communication Protocol and Foundation." [Online]. Available: http://en.hartcomm.org

[2] "ISA100, Wireless Systems for Automation". [Online]. Available: https://isa.org

[3] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt, "WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control," in *RTAS*. IEEE, Apr. 2008, pp. 377–386.

[4] S. Petersen and S. Carlsen, "WirelessHART Versus ISA100.11a: The Format War Hits the Factory Floor," *IEEE Industrial Electronics Magazine*, vol. 5, no. 4, pp. 23–34, Dec. 2011.

[5] T. O'donovan et al., "The GINSENG system for wireless monitoring and control," *ACM Transactions on Sensor Networks*, vol. 10, no. 1, pp. 1–40, 2013.

[6] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon, "Reliable and real-time communication in industrial wireless mesh networks," in *RTAS*, 2011, pp. 3–12.

[7] H. Z. H. Zhang, P. Soldati, and M. Johansson, "Optimal link scheduling and channel assignment for convergecast in linear WirelessHART networks," in *WiOPT*. 2009, pp. 1–8.

[8] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-time scheduling for WirelessHART networks," in *RTSS*, 2010, pp. 150–159.

[9] O. Chipara, C. Lu, J. A. Stankovic, and G.-C. Roman,, "Dynamic conflict-free transmission scheduling for sensor network queries," *IEEE Transactions on Mobile Computing*, vol. 10, no. 5, pp. 734–748, 2011.

[10] O. Chipara, C. Lu, and G.-C. Roman, "Real-Time Query Scheduling for Wireless Sensor Networks," in *in RTSS 2007*, pp. 389 - 399.

[11] W. Pöttner, H. Seidel, and J. Brown, "Constructing schedules for time-critical data delivery in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. V, no. 3, pp. 1–31, 2014.

[12] M. Yan, K. Y. Lam, S. Han, E. Chan, Q. Chen, P. Fan, D. Chen, and M. Nixon, "Hypergraph-based data link layer scheduling for reliable packet delivery in wireless sensing and control networks with end-to-end delay constraints," *Information Sciences*, vol. 278, pp. 34–55, 2014.

[13] P. Suriyachai, J. Brown, and U. Roedig, "Time-Critical Data Delivery in Wireless Sensor Networks," in *DCOSS*, 2010, pp. 216–229.

[14] B. Dezfouli, M. Radi, and O. Chipara, "Real-Time Communication in Low-Power Mobile Wireless Networks," in *CCNC*, IEEE, January 2016.

[15] O. Chipara, C. Lu, T. C. Bailey, and G.-C. Roman, "Reliable clinical monitoring using wireless sensor networks," in *SenSys*, 2010, p. 155.

[16] F.-J. Wu, Y.-F. Kao, and Y.-C. Tseng, "From wireless sensor networks towards cyber physical systems," *Pervasive and Mobile Computing*, vol. 7, no. 4, pp. 397–413, 2011.

[17] "MoteTrack." [Online]. Available: http://www.eecs.harvard.edu/ konrad/projects/motetrack/

[18] B. Dezfouli, M. Radi, and O. Chipara, "Mobility-Aware Real-Time Scheduling for Low-Power Wireless Networks," *Technical Report*, University of Iowa, Department of Computer Science, 2016. [Online]. Available: http://cs.uiowa.edu/ ochipara/papers/MARS.pdf

[19] B. Dezfouli, M. Radi, S. A. Razak, K. Whitehouse, K. A. Bakar, and T. Hwee-Pink, "Improving broadcast reliability for neighbor discovery, link estimation and collection tree construction in wireless sensor networks," *Computer Networks*, vol. 62, pp. 101–121, Apr. 2014.

[20] B. Dezfouli, M. Radi, S. A. Razak, HP. Tan, and K. A. Bakar, "Modeling low-power wireless communications," *Journal of Network and Computer Applications*, vol. 51, pp. 102–126, 2014.

[21] B. Dezfouli, M. Radi, K. Whitehouse, S. A. Razak, and HP. Tan, "CAMA: Efficient Modeling of the Capture Effect for Low-Power Wireless Networks", *ACM Transactions on Sensor Networks*, vol. 11, pp. 1–43, 2014.

[22] L. van Hoesel and P. Havinga, "Collision-free Time Slot Reuse in Multi-hop Wireless Sensor Networks," in *ISSNIP*. IEEE, 2005, pp. 101–107.

[23] A. Jhumka and S. Kulkarni, "On the design of mobility-tolerant TDMA-based media access control (MAC) protocol for mobile sensor networks," in *ICDCIT*. 2007, pp. 42–53.

[24] L. van Hoesel, A. Tuysuz-Erman, and P. Havinga, "Ideas on node mobility support in schedule-based medium access," in *ISSNIP*. 2008, pp. 539–544.

[25] A. Gonga, O. Landsiedel, and M. Johansson, "MobiSense: Power-efficient micro-mobility in wireless sensor networks," in *DCOSS*, 2011, pp. 1–8.

[26] X. Zhu, S. Han, P.-C. Huang, A. K. Mok, and D. Chen, "MBStar: A Real-time Communication Protocol for Wireless Body Area Networks," *ECRTS*. 2011, pp. 57–66.

[27] Y. H. Wei, Q. Leng, S. Han, A. K. Mok, W. Zhang, and M. Tomizuka, "RT-WiFi: Real-time high-speed communication protocol for wireless cyber-physical control applications," in *RTSS*, 2013, pp. 140–149.