z

REWIMO: A Real-Time and Reliable Low-Power Wireless Mobile Network

BEHNAM DEZFOULI, Santa Clara University
MARJAN RADI, The University of Iowa
OCTAV CHIPARA, The University of Iowa

Industrial applications and cyber-physical systems rely on real-time wireless networks to deliver data in a timely and reliable manner. However, existing solutions provide these guarantees only for stationary nodes. In this paper, we present REWIMO, a solution for real-time and reliable communications in mobile networks. REWIMO has a two-tier architecture composed of (i) infrastructure nodes and (ii) mobile nodes that associate with infrastructure nodes as they move. REWIMO employs an on-join bandwidth reservation approach and benefits from a set of techniques to efficiently reserve bandwidth for each mobile node at the time of its admission and over its potential communication paths. To ensure association of mobile nodes with infrastructure nodes over high-quality links, REWIMO uses the two-phase scheduling technique to coordinate neighbor discovery with data transmission. To mitigate the overhead of handling network dynamics, REWIMO employs an additive scheduling algorithm, which is capable of additive bandwidth reservation without modifying existing schedules. Compared to the algorithms used by static real-time wireless networks, the techniques and the algorithms employed by REWIMO result in a significant increase in real-time capacity, enhanced reliability, and considerably faster handling of network dynamics.

CCS Concepts: •**Networks** → **Network protocols; Network algorithms; Network dynamics; Cyber-physical networks; Mobile networks;** *Network architectures;*

Additional Key Words and Phrases: Scheduling, Timeliness, Reliability, Mobility

ACM Reference Format:
Behnam Dezfouli, Marjan Radi and Octav Chipara, 2016. REWIMO: A Real-Time and Reliable Low-Power Wireless Mobile Network. *ACM Trans. Sensor Netw.* x, y, Article z (June 2017), 42 pages.
DOI: 0000001.0000001

1. INTRODUCTION

The adoption of real-time wireless standards such as WirelessHART [25] and ISA100 [26] is making wireless technology an attractive solution for reducing the cost and for simplifying the deployment of process monitoring, control and smart manufacturing systems [27; 29; 30; 31; 12]. At the heart of these standards are centralized algorithms to schedule the transmissions of data flows in order to meet diverse end-to-end deadlines. The real-time systems community has made significant progress in developing scheduling algorithms and associated schedulability analysis [32; 33; 12; 31; 5; 13; 35; 36; 37]. However, existing approaches do not support real-time communications in mobile networks, which limits their applicability.

In this paper, we consider applications that require *real-time and reliable communications in the presence of mobility*. Numerous applications share this requirement. For

This work was supported by the National Science Foundation, under Grant No. 1144664, and by the Roy J. Carver Charitable Trust, under Grant No. 14-4355.
Author's addresses: B. Dezfouli: Department of Computer Engineering, Santa Clara University, CA 95053, USA; M. Radi and O. Chipara, Department of Computer Science, University of Iowa, IA 52242, USA.

DOI: 0000001.0000001

example, a clinical monitoring system may collect vital signs from mobile patients, determine the likelihood of their condition deteriorating, and issue alarms to the medical staff when necessary [38; 39]. The system must guarantee that the generated alarms are delivered within a bounded amount of time to ensure timely response to a patient's clinical deterioration. Another example is a warehouse that uses mobile robots to manage inventory [40]. For humans and robots to work together efficiently and safely, motion-planning data must be delivered to the robots and feedback regarding changes in the factory layout must be collected from robots in real-time.

A real-time wireless network must guarantee that the packets of *real-time flows* are delivered reliably within end-to-end deadlines. The design of such networks is particularly challenging due to issues such as the NP-hardness of scheduling real-time flows and handling unreliable wireless links [41; 33; 42; 43]. Mobility introduces the additional challenge of path uncertainty, as mobile nodes must continuously adapt their paths for reliable communications. Specifically, the network must guarantee that *once a node is admitted to the system, its packets are delivered by their deadlines regardless of the mobility patterns of mobile nodes*.

The problem of real-time communications in mobile networks may be approached using either *on-demand reservation* or *on-join reservation*. In an on-demand reservation approach, a mobile node updates its bandwidth reservation as paths 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 traffic requirements irrespective of its movement. We have evaluated the suitability of an on-demand approach in low-power networks by simulating the movement of users in a building based on the Motetrack dataset[1]. Figure 1 shows the number of updates necessary to achieve a reliability of 95% when collecting data from users moving at different speeds. For example, a network must update its global schedule 3 times per second to ensure real-time delivery from 50 nodes when they are moving at 1.5 m/s. A schedule update typically involves gathering connectivity information from the mobile nodes, uploading this information to the gateway, generating a new schedule, and disseminating it to all nodes. Our evaluations show that this approach introduces significant overhead in low-power networks with constrained bandwidth. More importantly, an on-demand approach cannot guarantee that the real-time communication requirements of a node will always be 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 bandwidth demand exceeds the network capacity. Due to these disadvantages, we will focus on the *on-join reservation* approach in this work.

This paper presents *REWIMO* – a wireless solution for real-time and reliable communications in mobile networks employing the on-join reservation strategy. REWIMO has a hierarchical network architecture composed of fixed infrastructure nodes and mobile nodes. The delivery of packets to the gateway is divided into two parts: from the mobile node to an infrastructure node, and from the infrastructure node to the gateway. An advantage of this network architecture is that mobility only impacts the first hop delivery of data (from mobile nodes to the gateway) that is dynamically updated as a mobile node associates with different infrastructure nodes in response to mobility. However, mobility has no effect on the routing of data from an infrastructure node to the gateway as it is performed over fixed infrastructure nodes. REWIMO addresses three critical challenges to provide real-time and reliable communications:

— The naive use of existing scheduling algorithms (e.g., [32; 31; 35; 12; 13; 5; 33]) in the presence of mobility results in small network capacity. REWIMO incorporates

[1]Full details regarding the simulation setup are included in Section 6.

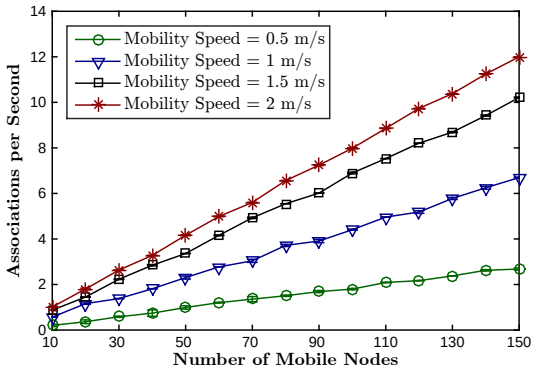


Fig. 1: The total number of associations depends on the number of mobile nodes and their movement speed. An on-demand reservation approach requires frequent path updates, which consume a significant fraction of the network's bandwidth.

novel transmission scheduling techniques to efficiently schedule transmissions in the presence of path uncertainty introduced by mobility. These techniques are based on the observation that even though data from a mobile node may be routed over multiple paths, only *one* path is active at a time. As a consequence, the transmissions of multiple paths can be scheduled in the same slot while dynamically activating one of the transmissions at run-time depending on the association of a mobile node.

— We show that without *coordinating the operation of the control and data planes*, a significant number of packets may be lost when mobile nodes associate with infrastructure nodes over low-quality links. At its core, this problem is the result of mobile nodes having out-of-date information regarding the quality of links to infrastructure nodes. To avoid this problem, REWIMO incorporates a two-phase scheduling technique that jointly schedules both data and beacon transmissions to ensure reliable packet delivery.

— REWIMO incorporates two scheduling algorithms — Mobility-Aware Real-time Scheduling (MARS) and Additive Mobility-Aware Real-time Scheduling (A-MARS) — that present different *trade-offs between network capacity and responsiveness to workload changes*. MARS supports a higher network capacity by recomputing and redistributing complete schedules when a node joins or leaves the network. In contrast, A-MARS schedules nodes additively, i.e., without rescheduling the flows of previously admitted nodes. As a result, it is sufficient to disseminate only the transmission schedules of a newly admitted mobile node in response to workload changes. A-MARS incorporates intelligent scheduling techniques which consider the demand of future real-time flows and provides only slightly lower network capacity compared to MARS.

To perform realistic and repeatable performance measurements, we have developed a sophisticated simulator which uses the packet reception traces provided as part of MoteTrack [2; 1]. Within the architecture proposed for REWIMO, we compare the performance of our algorithms against baseline algorithms designed for stationary real-time networks as well as their mobility-enhanced versions. Our evaluations show that the MARS algorithms increase the throughput and the number of admitted mobile nodes by more than 14x compared to the baselines. Furthermore, while the admission delay of the baselines is longer than 150 seconds and increases with the number

Table I: Summary of key notations

Description	Symbol
Gateway	GW
Set of infrastructure nodes	**I**
Set of mobile nodes	**M**
Set of flows in the network	$\mathbf{F} = \{i, j, ...\}$
Period of flow i	P_i
Deadline of flow i	D_i
Instance k of flow i	$J_{i,k}$
Release time of $J_{i,k}$	$r_{i,k}$
Absolute deadline of $J_{i,k}$	$d_{i,k}$
Set of flow classes	$\mathcal{F} = \{\gamma, \alpha, \beta, ...\}$
Transmission of flow i from node A to next-hop node B	$(AB)_i$
A path from a mobile node to the Gateway	Π
Least common multiplier of flows' periods (hyper-period)	T
A channel number	c_i
A slot number	s_i
Scheduling matrix	$\mathcal{M}$
Global beaconing period	P_{bc}
k^{th} instance of beacon broadcast by node A	$A_{bc,k}$
k^{th} instance of global beaconing period	$J_{bc,k}$
Potential Utilization vector of flow class α	PU_α

of mobile nodes admitted, the admission delay of A-MARS is always lower than 20 seconds, and more importantly, it remains constant as the number of mobile nodes increases. The additive scheduling feature of A-MARS results in a modest reduction in bandwidth utilization (approximatively 15% in the worst-case), compared to MARS, which requires the full reconstruction of schedules. In terms of reliability, we show that scheduling without using the two-phase scheduling technique results in a considerable reduction in reliability. For example, in a given sample scenario our evaluations show 30% reduction in packet delivery reliability when beacon and data transmissions are not coordinated.

The remainder of this paper is organized as follows. Section 2 provides a system overview. Section 3 presents techniques for efficient bandwidth reservation over multiple potential data forwarding paths. The implications of mobility and association on real-time and reliable communications with mobile nodes are presented in Section 4. The data scheduling algorithms used by REWIMO are given in Section 5. Performance evaluation results are presented in Section 6. Related work is given in Section 7. We conclude the paper in Section 8.

2. SYSTEM OVERVIEW

The network employs a two-tier architecture, composed of *infrastructure nodes* (**I**) and *mobile nodes* (M) (see Figure 2). Infrastructure nodes form a multi-hop wireless network through which mobile nodes can communicate with the *Gateway* (GW). The forwarding of packets from mobile nodes to the GW is divided into two parts: from the mobile node to the associated infrastructure node, and from the associated infrastructure node to the GW. We note that mobility only impacts the first-hop connection and has no impact on the delivery of packets once they reached an infrastructure node. The algorithms presented in this paper focus on the collection of data from mobile nodes to the GW as it is a core function of sensor networks. In this section, we first present the concepts of flows, transmissions and schedules. Next, we discuss the components and the high-level operation of REWIMO. Table I summarizes the key notations used in the paper.

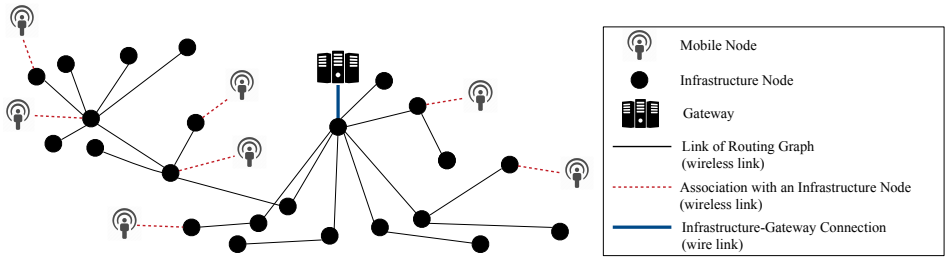


Fig. 2: REWIMO employs a two-tier network architecture composed of infrastructure and mobile nodes. This architecture decomposes mobility from routing, as mobility only affects mobile-infrastructure associations. The wireless infrastructure also simplifies the deployment of REWIMO.

2.1. Flows, Transmissions and Schedules

We adopt real-time flows as the communication primitive. A flow i is characterized by its period P_i and deadline D_i such that $P_i \leq D_i$. Both the period and deadline are measured in slots. 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 $d_{i,k} = r_{i,k} + D_i - 1$. According to the deadline monotonic policy, we assign each flow a static priority such that flows with shorter deadlines have higher priority. The notation $(AB)_i$ denotes a transmission (AB) pertaining to flow i. Note that we will omit i from the notation when the considered flow is clear from the context. Additionally, we will use the notations $(A*)$ and $(*A)$ to denote all of the outgoing and incoming transmissions of node A, respectively.

The GW implements a centralized scheduling algorithm that determines the slot and channel of each transmission. The schedule is represented as a matrix $\mathcal{M}$ whose rows indicate the selected channel and columns indicate the selected time slot. A feasible schedule must satisfy the following constraints: (i) A node transmits or receives only once in a time slot, as low-power radios are typically half-duplex. (ii) To avoid intra-network interference, we enforce that a single transmission can be scheduled on a given channel during a time slot. This assumption is consistent with WirelessHART [25] and is motivated by the need to achieve predictable behavior and high reliability in real-time networks. (iii) The hop-by-hop forwarding of packets introduces precedence constraints such that a sender must receive a packet before forwarding it to the next-hop node. (iv) Each instance of a flow must be delivered to its destination before its absolute deadline.

2.2. Architecture

2.2.1. Gateway (GW). The GW manages the network in a centralized manner[2] by implementing Frequency-Time Division Multiple Access (FTDMA) to support real-time and reliable communications. GW sends *control flow* (ct) messages to the nodes periodically. Control flows are used for various purposes such as schedule dissemination, time synchronization, and sending actuation commands to the nodes. In particular, GW disseminates a control flow whenever it is required to update the global schedule. The GW maintains two graphs, *upstream graph* and *downstream graph*, for packet routing to and from the GW, respectively. This paper assumes both upstream and downstream graphs are spanning trees. The GW is connected to the root of the spanning trees.

2.2.2. Infrastructure Nodes. Infrastructure nodes form a *wireless multi-hop infrastructure* through which mobile nodes can communicate with the GW. The set of infrastructure nodes is denoted by $\mathbf{I} = \{A, B, C, ...\}$. The fixed infrastructure is deployed to

[2]This design is similar to the WirelessHART standard [25; 29].

provide coverage within an area (e.g., a building).[3] Employing wireless infrastructure brings advantages such as ease of installation, simplified maintenance, and cost reductions, particularly in harsh environments with hard to access areas such as factories and refineries [3; 4; 5].

Each infrastructure node sends a *report flow* (rp) to the GW periodically. The report flow is used for three purposes: (i) *Join Report*: whenever a mobile node wants to *join the network*, a report flow is sent to the GW, requesting for bandwidth reservation. (ii) *Leave Report*: when a mobile node leaves the network, a notification is sent to the GW to release the bandwidth assigned to that mobile node. (iii) *Health Report*: nodes regularly report their health status to the GW. The report flows are routed to the GW using the upstream spanning tree.

2.2.3. Mobile Nodes. Each mobile node generates one or more *data flows* destined for the GW. In this paper we are interested in real-time scheduling of the data flows generated by mobile nodes. Since other flows, such as the control flow and reporting flows, are not affected by mobility, they can be scheduled using real-time scheduling algorithms designed for static wireless networks (please see Section 7.1).

2.3. Admitting a Mobile Node to the Network

2.3.1. Beaconing. Beacon broadcast allows the mobile nodes to: (i) discover nearby infrastructure nodes, (ii) perform time synchronization, and (iii) receive new schedules. To avoid beacon collision at mobile nodes, each infrastructure node broadcasts a beacon in a dedicated time slot assigned within the beacon period. The beaconing flow is denoted by bc. All the nodes use the same P_{bc}, and $P_{bc} = D_{bc}$. *The beaconing rate should be fast enough to track changes in neighborhood and links' quality*. In fact, beacon period depends on parameters such as nodes' movement speed and infrastructure's coverage. The beaconing channel can be either fixed and set to the best interference-free channel[4], or nodes could switch to a new beaconing channel in each beacon period. We use the former approach in this paper. In this case, the network discovery duration for a mobile node intending to join the network is bounded as $D_{disc} \leq P_{bc}$. In the latter case, $D_{disc} \leq number\ of\ channels \times P_{bc}$.

2.3.2. Join Request. When a mobile node intends to join the network, it sends a *join request* packet, including information about the data flows the node will generate. Periodically, in a specific time slot and channel, all the infrastructure nodes listen for join request packets. The scheduling of this slot, referred to as the *request reception slot*, is modeled by flow rq with period P_{rq}. Note that there is only one request reception time slot within each instance of rq. Since the number of mobile nodes requesting to join the network is unknown, we use slotted CSMA to transmit the requests within the reserved slots.

2.3.3. Schedule Dissemination. When an infrastructure node receives a join request, it forwards the request to the GW using the report flow (rp). When the GW receives the join request, it reserves bandwidth for the data flows of the mobile node, if possible. The number of admissible mobile nodes depends on the employed scheduling algorithm. After computing the schedule, the GW disseminates the new schedule through control flow (ct). 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

[3]This infrastructure is not similar to cellular and 802.11 LANs where infrastructure nodes communicate through wire links (e.g., X2 for LTE, and Ethernet for 802.11). Cellular and 802.11 networks employ single-hop communication with mobile nodes, and wireless bandwidth reservation between infrastructure nodes is not an issue in these networks.

[4]For example, channel 26 of 802.15.4 does not interfere with 802.11.

modified. When an infrastructure node receives scheduling information in response to a join request, it broadcasts that information through the beaconing flow (bc) to be received by the mobile node. Once the node joins the network, it can start transmitting its data while its real-time and reliable performance is guaranteed.

2.4. Mobility and Association

As a mobile node moves, it associates with different infrastructure nodes. It is essential for the mobile node to have up-to-date information about its neighborhood to achieve high reliability. This requires both selecting an appropriate beaconing rate and carefully coordinating beaconing and data transmissions. The details of the association process are discussed in Section 4.

3. REAL-TIME SCHEDULING TECHNIQUES FOR MOBILE NETWORKS

In this section, we start by formalizing the problem of real-time scheduling for mobile networks. We will show that the naive application of existing scheduling techniques results in low real-time capacity, as slots must be allocated over all potential paths from a mobile node to the gateway. To address this limitation, we characterize the impact of mobility on routing and leverage these observations to develop new scheduling techniques for improving real-time capacity. The techniques presented in this section focus on the scheduling of a single instance of a real-time flow. In Section 5, we will incorporate these techniques in real-time scheduling algorithms that handle multiple real-time flows.

3.1. Flow Merging

The delivery of packets from a mobile node M to the GW is divided into two parts: (i) single-hop communication between the mobile node and an associated infrastructure node and, (ii) potentially multi-hop communication, from the associated infrastructure to the GW[5]. Since we adopt an on-join approach, we must ensure that sufficient bandwidth is allocated for a mobile node to transmit its packets regardless of its mobility pattern.

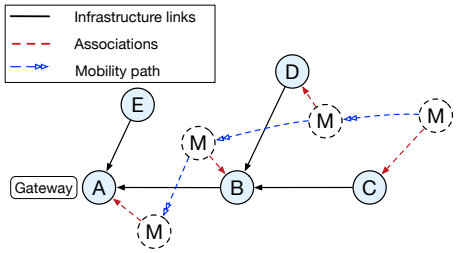
We can account for all of the potential mobility paths of a node by constructing an *Augmented Communication Graph* (ACG). The ACG is constructed by starting with the upstream spanning tree and adding an edge from a mobile node to each infrastructure node. To schedule a single packet from a mobile node, conflict-free transmissions must be assigned to forward packets from the mobile node to the GW over all paths in ACG. As a starting point, we define two transmissions (AB) and (CD) to be scheduled conflict-free if: (i) A, B, C, and D are distinct nodes and (ii) (AB) and (CD) are assigned to transmit on different channels. Later, we will relax these constraints to take advantage of the properties of mobility. The notations $parent(A)$ and $children(A)$ denote the parent and children of A in ACG. Similarly, the depth of a link (AB), denoted by $depth(AB)$, is the depth of B in the ACG.

An example of a topology and associated ACG are included in Figure 3. The ACG has five potential paths that M may use depending on its mobility pattern:

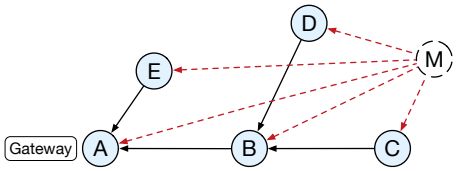
$$\Pi_1\text{: } (MC)(CB)(BA) \quad \Pi_2\text{: } (MD)(DB)(BA)$$
$$\Pi_3\text{: } (MB)(BA) \quad \Pi_4\text{: } (ME)(EA) \quad \Pi_5\text{: } (MA)$$

Here, Π denotes a path from the mobile node to the GW. Consistent with an on-join approach, allocating bandwidth for M's transmission over all paths is sufficient to meet its real-time communication demands *regardless of its mobility pattern*.

[5]As the root of the routing graphs is connected to the GW, we simply refer to the root node as GW. Given this definition, a mobile node may deliver its packet to the GW directly.



(a) M associates with different infrastructure nodes as it moves.



(b) ACG for topology in Figure 3a.

Fig. 3: The ACG captures all the paths through which the data generated by M may be forwarded to the GW.

s	0	1	2	3	4	5	6	7
Π_1	(MC)	(CB)				(BA)		
Π_2		(MD)	(DB)				(BA)	
Π_3				(MB)				(BA)
Π_4			(ME)	(EA)				
Π_5					(MA)			

(a) Naive scheduler

s	0	1	2	3	4	5	6	7
Π_1	(MC)	(CB)	(BA)					
Π_2	(MD)	(DB)	(BA)					
Π_3	(MB)	(BA)						
Π_4	(ME)	(EA)						
Π_5	(MA)							

(b) Flow merging

s	0	1	2	3	4	5	6	7
Π_1	(MC)	(CB)	(BA)					
Π_2	(MD)	(DB)						
Π_3	(MB)							
Π_4	(ME)	(EA)						
Π_5	(MA)							

(c) Flow coordination

s	0	1	2	3	4	5	6	7
Π_1						(MC)	(CB)	(BA)
Π_2						(MD)	(DB)	
Π_3							(MB)	
Π_4							(ME)	(EA)
Π_5								(MA)

(d) Reverse scheduling

Fig. 4: Scheduling of transmissions over all paths in Figure 3 using different techniques. Black and gray (red) transmissions are scheduled in channel c_1 and c_2, respectively.

A naive scheduler would schedule the transmissions of each path in the ACG independently as follows. The naive scheduler maintains a *ready set* that includes all the transmissions that may be scheduled. Initially, the ready set includes all the links from a mobile node to infrastructure nodes. In each time slot, the scheduler considers the transmissions in the ready set according to their depth in the ACG. A transmission is assigned to the first unused channel in the current slot. When no channel is available, we proceed to schedule transmissions in the subsequent slots. Upon scheduling a transmission (MC), the transmission (MC) is removed from the ready set and, the next transmission (CB), where $B = parent(C)$, is added to the ready set.

Let us consider the topology in Figure 3b where the mobile node M has a single flow i with $P_i = D_i = 8$. The naive scheduler produces the schedule shown in Figure 4a and as `row 1` of Figure 5. The two figures provide two different perspectives on how transmissions are assigned. As Figure 5 shows, the naive scheduler requires a total of 11 entries[6] in the scheduling matrix spread over two channels to schedule the transmissions of flow i.

The naive scheduler works similar to the state-of-the-art scheduling algorithms for static real-time networks (e.g., [32; 31; 35; 12; 13; 5; 33]). However, as shown by the experiments presented in Section 6, the naive scheduler provides low real-time capacity. The real-time capacity can be significantly increased by observing that even though

[6]An entry of the scheduling matrix is the intersection of a time slot and a channel.

s			0	1	2	3	4	5	6	7
row 1	Naive Scheduling	c_1	(MC)	(MD)	(DB)	(MB)	(MA)	(BA)	(BA)	(BA)
		c_2		(CB)	(ME)	(EA)				
row 2	Flow Merging	c_1	(MC) (MD) (MB) (ME) (MA)	(CB) (DB) (BA) (EA)	(BA)					
		c_2								
row 3	Flow Coordination	c_1	(MC) (MD) (MB) (ME) (MA)	(CB) (DB) (EA)	(BA)					
		c_2								
row 4	Reverse Scheduling	c_1						(MC) (MD)	(ME) (MB) (DB) (CB)	(EA) (BA) (MA)
		c_2								

Fig. 5: The scheduling matrix for Figure 3 using different scheduling techniques.

there are multiple paths over which M's packets may be forwarded, there is *only one path active at a time*. To take advantage of this property, we relax the constraints of scheduling transmissions in the ACG to allow for multiple transmissions of the *same* flow to be scheduled in the *same entry of the scheduling matrix*. A consequence of this relaxation is that the transmissions scheduled in the same entry may violate the definition of conflict-free transmissions. However, the association algorithm (to be presented in Section 4) guarantees that at run-time a mobile node associates and uses a single path to route the data for each instance of a real-time flow.

THEOREM 3.1. *Flow Merging: For a flow i, transmissions $(AB)_i$ and $(CD)_i$ on two paths $(AB)_i \in \Pi_1$ and $(CD)_i \in \Pi_2$ $(\Pi_1 \neq \Pi_2)$ may be scheduled in the same entry of the scheduling matrix.*

PROOF. A mobile node may use any of the paths $\Pi_1, \Pi_2, \ldots, \Pi_{|I|}$ to deliver the packets of an instance $J_{i,k}$ of flow i. The mobile node will be associated with a single infrastructure node and activates path Π_a. Consider two transmissions $(AB)_i$ and $(CD)_i$ belonging to paths Π_1 and Π_2, respectively. There are three cases to consider:
Case 1: Neither path is activated ($\Pi_1 \neq \Pi_2 \neq \Pi_a$). Since neither $(AB)_i$ nor $(CD)_i$ are activated, there cannot be any conflict.
Case 2: One path is activated ($\Pi_1 = \Pi_a$ or $\Pi_2 = \Pi_a$). Since either $(AB)_i$ or $(CD)_i$ is activated, a conflict cannot happen since a single transmission is activated.
Case 3: Both paths are activated ($\Pi_1 = \Pi_a$ and $\Pi_2 = \Pi_a$). This case cannot happen because only one path is activated to handle an instance of flow i. □

It is important to note that Theorem 3.1 allows merging the transmissions belonging to the same flow; the transmissions of different flows cannot be merged since mobile nodes make association decisions independently in a distributed manner.

The schedule constructed using the naive scheduler modified with Theorem 3.1 is shown in Figure 4b and as `row 2` of Figure 5. Flow merging significantly improves real-time capacity: the number of scheduling matrix entries required to schedule the flow instance is reduced from 11 to 3. The reduction in the number of used entries is the result of scheduling multiple transmissions in the same entry of the scheduling matrix. As it is apparent from the schedule shown in `row 2` of Figure 5, an entry may contain potentially conflicting transmissions. For example, conflicting transmissions (MC) and (MD) are scheduled in entry $\mathcal{M}[c_1, 0]$. However, we ensure that no conflict happens by

s	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Instance	$J_{i,k}$								$J_{i,k+1}$							
Schedule	(MC) (MD) (MB) (ME) (MA)	(CB) (DB) (BA) (EA)	(BA) (BA)						(MC) (MD) (MB) (ME) (MA)	(CB) (DB) (BA) (EA)	(BA) (BA)					
Activated Path	Π_1								Π_2							
Schedule Activation	(MC)	(CB)	(BA)						(MD)	(DB)	(BA)					

Fig. 6: REWIMO may use different paths to forward the data of instances $J_{i,k}$ and $J_{i,k+1}$. Depending on the activated path, different transmissions will be performed.

activating one path for handling each instance of a flow. Figure 6 shows paths Π_1 and Π_2 being activated to route the packets of instances $J_{i,k}$ and $J_{i,k+1}$. Accordingly, (MC) on Π_1 is activated to handle $J_{i,k}$. Similarly, (MD) on Π_2 is activated to handle $J_{i,k+1}$. We defer the details of the association and activation algorithms to Section 4.

3.2. Efficient Use of Flow Merging

Flow merging provides the basic mechanism for improving the real-time capacity of the network. In the following, we consider the question of how to maximize the opportunities for flow merging during the scheduling process.

A close inspection of the scheduling matrix shows that transmission (BA) was scheduled three times even though it actually transmits at most one packet during any instance of flow i (see Figure 4b). 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 must wait to receive the transmissions from its children before transmitting to its parent. We refer to this rule as *flow coordination*:

RULE 1. *Flow Coordination: For a flow i, a transmission $(BC)_i$ must be scheduled after transmissions $(AB)_i$, where $A \in children(B)$, have been scheduled.*

Rule 1 coordinates transmission scheduling over multiple paths by constraining when transmissions are added to the ready set. Figure 4c shows how the transmissions of various paths are scheduled when Rule 1 is applied. The scheduling matrix corresponding to this schedule is given in `row 3` of Figure 5. The flow coordination rule does not reduce the number of scheduling matrix entries used compared to flow merging. However, as shown in Figure 4c, the number of transmissions scheduled is reduced from 11 to 9.

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 such that: the transmissions are scheduled from the mobile node toward the gateway and the scheduling of an instance $J_{i,k}$ starts in slot $r_{i,k}$ when the instance is released. Forward scheduling yields suboptimal results because it limits the reuses of infrastructure nodes by other flows. To highlight the inefficiency of forward scheduling, we say that a node is *blocked* in a time slot if it is scheduled to receive or transmit for a flow. An infrastructure node that is blocked in a slot cannot be reused to schedule other flows on any channel of that slot. Figure 7 shows as `row 1` the blocking slots for forward scheduling. For example, 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 transmission or reception of flow

s			0	1	2	3	4	5	6	7	#Blocked Slots
row 1	Forward Scheduling	A	×	×	×						3
		B	×	×	×						3
		C	×	×							2
		D	×	×							2
		E	×	×							2
row 2	Reverse Scheduling	A								×	1
		B							×	×	2
		C						×	×		2
		D						×	×		2
		E							×	×	2

Fig. 7: The blocking slots of infrastructure nodes when forward scheduling (`row 1`) and reverse scheduling (`row 2`) are applied. Node A is blocked in three slots and one slot when forward and reverse scheduling are used, respectively.

i. It is easy to see that adding $(AB)_i$ to such an entry does not increase the number of blocked nodes. Additionally, scheduling transmissions in reverse order starting from the deadline towards the release time may reduce blocking. This ordering implies the construction of a *reverse ready* set that initially includes the transmissions that deliver a flow to the GW. In this approach, 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 order transmissions in the reverse ready set according to their depth such that transmissions with smaller depth are considered for scheduling before those with higher depth.

For the scenario given in Figure 3b, reverse scheduling starts scheduling $J_{i,0}$ in slot $d_{i,0} = 7$. The reverse ready set initially includes all the transmissions to the GW: $\Theta_{rready} = \{ (MA), (EA), (BA) \}$. Since the transmissions pertain to the same flow, the flow merging technique allows us to schedule all of them in the same slot 7. The ready set is updated by adding the transmissions that become available. For example, the scheduling of (BA) results in the addition of (DB) and (CB) to the reverse ready set. The algorithm produces the schedule shown in Figure 4d and as `row 4` in Figure 5. The reverse schedule reduces the number of blocked slots from 12 to 9 (compare `row 1` and `row 2` of Figure 7).

The reverse scheduling reduces the number of blocking slots compared to the forward scheduling technique. In fact, reverse scheduling schedules an infrastructure node to receive in only one slot of the schedule of a flow instance, which is optimal.

THEOREM 3.2. *Assume that we are interested in scheduling a single flow. Reverse scheduling schedules only one reception time slot for an infrastructure node in a single slot, which is optimal.*

PROOF. Considering Figure 8, assume that we need to schedule a transmission $(AB)_i$ for flow i generated by M, and M can deliver this flow to A over n paths with different number of hops. The shortest path is (MA) with only one hop, and the longest path has n hops. Using the forward scheduling technique, transmissions of type $(*A)_i$ are added to the ready set and scheduled during n consecutive slots to deliver this flow to A. Therefore, A is blocked in $n+1$ slots. Using the reverse scheduling technique, transmissions $(*A)_i$ are added to the ready set at the same time, and they can all be scheduled in one slot. Therefore, the blocking number of A is reduced to 2, which is the minimum number of required slots to receive a packet and forward it to the next hop. □

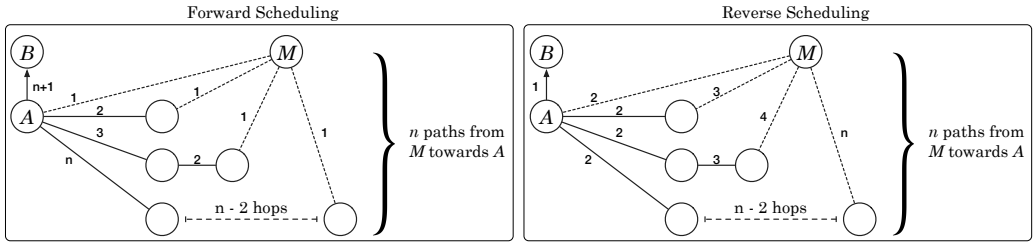


Fig. 8: Assume n paths, with lengths 1 to n, exist between M and A. The value on each link is the slot number in which that link is scheduled. The blocking number of A in this scenario equals $n+1$ and 2 using forward and reverse scheduling techniques, respectively.

In addition to its effects on improving network capacity, reverse scheduling also improves the energy-efficiency of REWIMO. Referring to Figure 8, node A needs to listen to the medium during n slots when forward scheduling is used. However, this value is reduced to 1 when reverse scheduling is employed. We will evaluate the energy efficiency of REWIMO in Section 6.

4. ASSOCIATION AND PATH ACTIVATION

The scheduling techniques developed in the previous section increase real-time capacity under the assumption that even though there are many paths that may be used to forward data from a mobile node, only one of them is used to forward an instance of a flow. The goal of the path activation algorithm is to ensure that *only one path that provides high reliability is activated* to route the data for a flow instance.

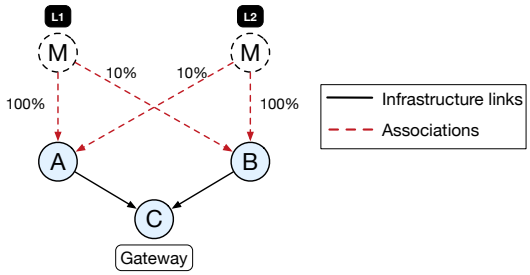
4.1. Impact of Scheduling on Reliability

Consistent with REWIMO's two-tier architecture, we divide the activation process into two parts: (i) the dynamic association of a mobile node to a single infrastructure node and (ii) the activation of a path from an infrastructure node to the GW. Mobile nodes use the association algorithm given in Algorithm 1 to decide about association with infrastructure nodes. The association algorithm forwards a mobile node's data to the infrastructure node which provides highest link quality. The infrastructure node that received the data will forward it by activating its path to the GW. The activation of a link along the path is triggered by the reception of the packet from the previous infrastructure node. The activation process takes place at run-time after scheduling. Furthermore, the activation process guarantees that only one path is active at a time by associating with *a single* infrastructure node to transmit the packets of a flow instance. Therefore, the key challenge is to activate a high-quality link between the mobile node and infrastructure, which requires a mobile node to have up-to-date estimates of its link quality to neighbors.

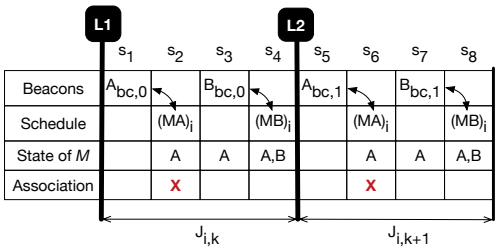
The classic approach to solving the association problem is for a mobile node to maintain a neighborhood table of estimated link quality to its neighbors. The information in the table is updated based on the reception of periodic beacons from the infrastructure nodes. We ensure that the stale information is removed from the table by removing the entries belonging to neighbors from which it did not receive a beacon during the last period. All infrastructure nodes transmit beacons with the same period P_{bc}; however, they are shifted in time to avoid concurrent transmissions. We will use the notation $A_{bc,k}$ to denote the transmission of the k^{th} beacon from A. We assume information obtained from a beacon is valid for at least τ slots. The variable τ captures the time scale at which neighborhood changes occur. Therefore, the quality of link (MA) is known

ALGORITHM 1: Association Algorithm (running on mobile nodes)

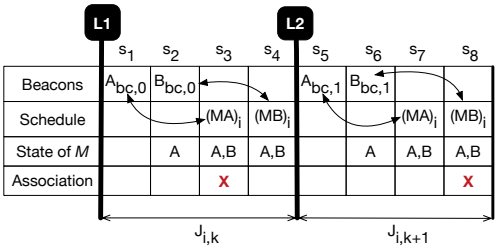
```
1  table : neighborhood table
2  on receive(A_{bc,k}):
3      table[A] = quality of link (MA) based on received beacon A_{bc,k};
4      cancel timeout(A) if pending;
5      schedule timeout(A, P_{bc});
6  on timeout(A):
7      delete table[A];
8  on slot s:
9      Let B be the infrastructure node that has the best link quality in table;
10     activate (MB)_i if it is scheduled in slot s (i.e., ∃c ∈ C s.t. (MB)_i ∈ M[c, s]));
```



(a) M may associate with infrastructure nodes A or B with link reliability of 10% and 100%, respectively.



(b) Beacon and data transmissions are not coordinated.



(c) Beacon and data transmissions are coordinated.

Fig. 9: A mobile node M moves from location L1 to L2. At location L1, M may associate with either A with 100% reliability or with B with 10% reliability. Conversely, at location L2, M may associate with either A with 10% reliability or with B with 100% reliability. Figures 9b and 9c capture the association decisions of M when beacons are transmitted asynchronously or synchronously. The tables include four rows describing: the time when beacons were transmitted (row "Beacons"), the time when transmissions were scheduled for a flow i (row "Schedule"), the state of the neighborhood table (row "State"), and the decision for association (row "Association"). Note that slots $s_1, s_2, \ldots, s_8$ may not be contiguous.

accurately if:

$$\exists k' \in \mathbb{N} \text{ such that } S[(MA)] - S[A_{bc,k'}] \leq \tau \tag{1}$$

where $S[(MA)]$ is the slot when transmission (MA) is scheduled and $S[A_{bc,k'}]$ is the slot when beacon $A_{bc,k'}$ is scheduled. In the following, we will show that without synchronizing the transmissions of beacons and data transmissions, it is possible for a mobile node to make suboptimal association decisions even when the information in the neighborhood table is up-to-date (i.e., the constraint in Equation 1 is satisfied).

Consider a simple scenario where a mobile node M moves from location L1 to location L2 (see Figure 9a). At location L1, M may associate with either A with 100%

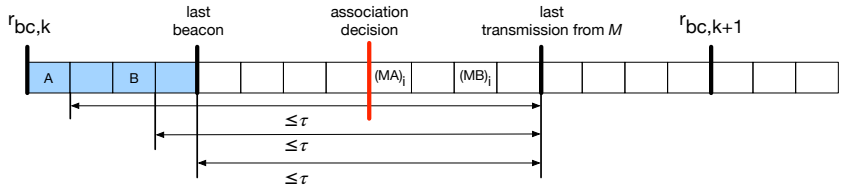


Fig. 10: Organization of beacon and data transmissions within a beaconing instance.

reliability or with B with 10% reliability. At location L2, the situation is the opposite: M may associate with A with 10% reliability or with B with 100% reliability. We will examine M's associations for two consecutive instances $J_{i,k}$ and $J_{i,k+1}$. Figure 9b shows the associations of M when there is no coordination between beacon and data transmissions. The black bars indicate the stability constraints due to Equation 1. For example, the time distance between slots s_1 and s_2 must be smaller or equal to τ (i.e., $s_2 - s_1 \leq \tau$) to ensure that the quality of link $(MA)_i$ is up-to-date at s_2. Initially, M is at location L1 when it receives beacon $A_{bc,0}$. Upon its reception, M updates its state to indicate that it has a high link quality to A. In slot s_2, M must decide whether to associate with A and activate transmission $(MA)_i$. Since M has only information regarding $(MA)_i$ it must associate with node A, which provides high link quality. After the completion of $J_{i,k}$, M moves to L2. The same sequence of events leads again to the activation of $(MA)_i$ for $J_{i,k+1}$. However, at location L2, link (MA) provides only a reliability of 10%. Note that this decision is in spite of having up-to-date link quality estimate for $(MA)_i$ at s_6 and for $(MB)_i$ at s_8.

The reason for this behavior is a race condition between when the state of node M is used to make association decisions and when it is updated via beacons. We observe that there is a critical time $\Psi_{i,k}$ when a node makes an association decision for the first data transmission pertaining to an instance $J_{i,k}$. At $\Psi_{i,k}$, the mobile node must know the quality of links to *all* infrastructure nodes to select the best infrastructure node to associate with:

$$\exists k' \in \mathbb{N} \text{ such that } \Psi_{i,k} - S[A_{bc,k'}] \leq \tau \;\; \forall A \in \mathbf{I} \tag{2}$$

The constraints of Equation 2 capture the minimal coordination required between the transmission of beacon and data packets to ensure that a mobile node associates with high quality infrastructure nodes.

A simple approach to satisfying the constraints in Equation 2 is to ensure that beacons are always scheduled before the slots in which mobile-to-infrastructure transmissions have been scheduled.[7] Figure 9c shows the case when the scheduling of beacon and data transmissions for the considered example is coordinated. The beacons from A and B occur before their respective transmissions. This ensures that at the critical decision points $\Psi_{i,k}$ and $\Psi_{i,k+1}$, node M has accurate information about all its links to infrastructure nodes. Using this information, M associates with A for $J_{i,k}$ and with B for $J_{i,k+1}$.

REWIMO's beacon scheduling algorithm is included as Algorithm 2. It ensures that high-quality paths are activated by satisfying the constraints of Equation 2, as follows. REWIMO defines global beaconing instances, each with period $P_{bc} = \tau$ (see Figure 10).

[7]It is worth noting that this problem cannot be resolved through enabling concurrent reception of mobile nodes' data packets by multiple infrastructure nodes. For example, assume that two infrastructure nodes are enabled to concurrently receive a packet sent by a mobile node. This is a possible scenario: mobile node M activates its link to E and F while it has higher link quality to G and H, and M activates its link to G and H while it has higher link quality to E and F. Therefore, enabling the reception of a packet by multiple infrastructure nodes may only alleviate the problem instead of providing a complete solution.

ALGORITHM 2: Beacon Scheduling Algorithm

Input: P_{bc}: beaconing period; c: channel used for beacon scheduling;
Output: beacon broadcast schedules by infrastructure nodes (satisfying constraints of Equation 2)

```
1 begin
2     for each beaconing instance J_{bc,k} within hyper-period T do
3         φ = 0;
4         for each node A ∈ I do
5             M[c, r_{bc,k} + φ] = A_{bc,k};
6             φ = φ + 1;
7     return;
```

In the beginning of each beaconing instance, all the infrastructure nodes are scheduled to transmit beacons on consecutive slots. The remainder of the instance is dedicated to the transmission of data packets. All the beaconing transmissions are scheduled in channel c because mobile nodes need to know on what channel beacons are sent. We note that there is the potential for the transmissions of a data flow instance to span multiple beaconing instances. However, this would violate the constraints of Equation 2. In this case, REWIMO employs a two-phase scheduling technique (to be described in Section 5) to ensure association reliability by constraining that all mobile-to-infrastructure transmissions occur in the same beaconing instance. No constraints need to be imposed regarding the transmission of packets over the infrastructure nodes to the GW.

THEOREM 4.1. *REWIMO beacon organization meets the constraints of Equation 2.*

PROOF. Consider the k^{th} beaconing instance that starts at $r_{bc,k}$. The beacon transmissions are organized in the beginning of the beaconing instance. Let L be the slot within the beaconing instance when the last beacon is transmitted. The freshness constraint requires that at every critical activation time we have up-to-date information about all infrastructure nodes. Activations may happen in every slot x such that $L < x < r_{bc,k+1}$. Since $x - r_{bc,k} \leq \tau$ and $\tau = P_{bc}$, then at potential activation point the information is up-to-date (consistent with Equation 2). □

4.2. Link Estimation

A practical consideration is how link quality is estimated from beacons. In REWIMO, mobile nodes estimate their links' quality to infrastructure nodes using the *received signal strength indicator* (RSSI) of beacon packets. RSSI is readily available on low-power radios such as CC2420 [55], CC2500 [56] and CC2650 [57]. The robustness of RSSI as an indicator of link quality has been extensively studied empirically. While the usefulness of RSSI to discriminate between links in the transitional region is debatable[8], RSSI can be effectively used to identify high-quality links in the connected region [6; 7; 8]. For example, Baccour et al. [6; 9] showed that LQI and RSSI can differentiate between high-quality links and other links. A property of high-quality links is that they are stable and symmetric [10]. Therefore, receiving a beacon from an infrastructure node over a high-quality link allows the mobile node to estimate its link to the infrastructure node with high accuracy. Additionally, RSSI may be combined with other indicators of link quality such as *link quality indicator* (LQI) to further improve the accuracy of estimations. Note that the deployment of mission-critical wireless networks requires an extensive site survey to ensure the presence of high-quality links

[8]Literature provides evidence supporting both sides of the argument.

(e.g., using tools such as Radio Mapping [11]). Similarly, REWIMO requires the presence of high-quality links at every position at which a user may be located. The network deployment we used in this paper for the performance evaluation of REWIMO satisfies this requirement. To improve the reliability of link estimation without increasing the overhead of beaconing, REWIMO can benefit from mapping tools and localization techniques as well [11; 1]. For example, mobile nodes can keep track of their location and rely on a history of link estimations to find the best infrastructure node nearby in a given position.

5. SCHEDULING ALGORITHMS

In this section, we present REWIMO's scheduling algorithms: *Mobility-Aware Real-time Scheduling* (MARS) and *Additive Mobility-Aware Real-time Scheduling* (A-MARS). The algorithms use the scheduling techniques described in Section 3 to efficiently schedule the real-time flows of mobile nodes. Additionally, as discussed in Section 4, the algorithms must coordinate the schedules of beacon broadcasts and data transmissions to provide high reliability. REWIMO's algorithms provide two operating points with different trade-offs between real-time capacity and responsiveness to dynamics. MARS is designed to achieve the highest real-time capacity by rescheduling all flows when a new mobile node joins or leaves the network. However, this may lead to significant admission delays when there are numerous requests to join or leave the network. In contrast, A-MARS employs predictive techniques that estimate the impact of future flow arrivals to be more responsive than MARS while providing slightly lower real-time capacity.

5.1. MARS: Mobility-Aware Real-Time Scheduling

MARS constructs schedules that, in conjunction with the association algorithm, guarantee real-time and reliable data delivery. MARS runs on the GW and is invoked each time a new node requests to join the network. If the workload can be scheduled, MARS will return a scheduling matrix. Otherwise, it will return "unsuccessful" and notifies the requesting node that it may not be scheduled.

MARS's pseudo-code is included as Algorithm 3. Since real-time flows are periodic, the release pattern of the flows repeats every hyper-period T, which is the least common multiplier of flows' periods. Accordingly, to schedule the entire network, it is sufficient to construct a scheduling matrix $\mathcal{M}$ whose size is $number\ of\ channels \times T$, where T is the hyper-period of the flows. The scheduling matrix $\mathcal{M}$ is repeated every hyper-period.

MARS schedules all the flows $\mathbf{F}$ including those that have been already scheduled. MARS divides the scheduling of an instance of a real-time flow in two phases implemented as procedures `FirstPhase()` and `SecondPhase()`. For a given flow instance $J_{i,k}$, in the first phase infrastructure-to-infrastructure transmissions are scheduled. If the first phase completes successfully, MARS schedules the transmissions between mobile node and infrastructure nodes during the second phase. The scheduling process must ensure that all mobile-to-infrastructure transmissions belonging to an instance of a flow are scheduled during one beaconing instance.

Consider the scheduling of an instance $J_{i,k}$ by the `FirstPhase()` procedure. Consistent with a reverse scheduling approach, the transmissions of $J_{i,k}$ are considered for scheduling starting from the deadline $d_{i,k}$ down to $r_{i,k}$. A feasible schedule is constructed if all the transmissions of $J_{i,k}$ are assigned within interval $r_{i,k} \dots d_{i,k}$. The algorithm maintains two sets Θ_{rready} and $\Theta_{scheduled}$ that are updated in each time slot. The set $\Theta_{scheduled}$ includes the transmissions that were scheduled during the current slot s. The set Θ_{rready} includes all the transmissions that are reverse ready. Initially, Θ_{rready} includes all the transmissions from infrastructure nodes to the GW. In each

ALGORITHM 3: Mobility-Aware Real-time Scheduling (MARS)

Input: $\mathbf{F}$: set of the flows to be scheduled
Output: returns scheduling matrix $\mathcal{M}$ after a successful scheduling; otherwise returns 'unsuccessful'

begin
  T = least common multiplier of flows' periods (hyper-period);
  for *each flow* i *in* $\mathbf{F}$ *ordered by their priority* **do**
    M = the mobile node producing flow i;
    /*PHASE 1: Schedule infrastructure-to-infrastructure transmissions of flow i */
    for *each instance* $J_{i,k}$ *of flow* i *in hyper-period* T **do**
      if *FirstPhase*$(J_{i,k})$ = *unsuccessful* **then return** *unsuccessful* ;
    /*PHASE 2: Schedule mobile-to-infrastructure transmissions of flow i */
    for *each instance* $J_{i,k}$ *of flow* i *in hyper-period* T **do**
      if *SecondPhase*$(J_{i,k})$ = *unsuccessful* **then return** *unsuccessful* ;
  return $\mathcal{M}$;

Procedure FirstPhase()
  $\Theta_{rready} = \{(AB)_i | (B = GW)$ and $(A \in \mathbf{I})\}$;
  for $s = d_{i,k}$ *down to* $r_{i,k}$ **do**
    $\Theta_{scheduled} = \emptyset$;
    for *every transmission* $(AB)_i$ *in* Θ_{rready} *sorted by* $depth(AB)$ **do**
      c = CSA$((AB)_i, s)$ /*check schedulability of $(AB)_i$ in s */
      if $c \neq -1$ **then**
        $\mathcal{M}[c, s] = \mathcal{M}[c, s] \cup \{(AB)_i\}$;
        $\Theta_{scheduled} = \Theta_{scheduled} \cup \{(AB)_i\}$;
    $\Theta_{rready} = \Theta_{rready} \setminus \Theta_{scheduled}$;
    for *every* $(AB)_i$ *in* $\Theta_{scheduled}$ **do**
      $\Theta_{rready} = \Theta_{rready} \cup \{(CA)_i | C \in children(A)$ and $C \in \mathbf{I}\}$;
  if $\Theta_{rready} \neq \emptyset$ **then return** *unsuccessful* ;

Procedure SecondPhase()
  $\mathcal{M}' = \mathcal{M}$;
  $\Theta_{rready} = \Theta' = \{(MA)_i | A = GW\}$;
  $bperiod = \left\lfloor \frac{d_{i,k}}{P_{bc}} \right\rfloor$;
  for $s = d_{i,k}$ *down to* $r_{i,k}$ **do**
    $bcurrent = \left\lfloor \frac{s}{P_{bc}} \right\rfloor$;
    for *every transmission* $(MB)_i$ *in* Θ_{rready} *sorted by* $depth(MB)$ **do**
      c = CSA$((MB)_i, s)$;
      if $c \neq -1$ **then**
        if $bcurrent \neq bperiod$ **then**
          $bperiod = bcurrent$;
          $\Theta_{rready} = \Theta'$;
          $\mathcal{M} = \mathcal{M}'$;
        $\mathcal{M}[c, s] = \mathcal{M}[c, s] \cup \{(MB)_i\}$
        $\Theta_{rready} = \Theta_{rready} \setminus \{(MB)_i\}$
    for *every transmission* $(A*)_i$, *where* $A \in \mathbf{I}$, *scheduled in slot* s **do**
      $\Theta_{rready} = \Theta_{rready} \cup \{(MA)_i\}$;
      $\Theta' = \Theta' \cup \{(MA)_i\}$;
  if $\Theta_{rready} \neq \emptyset$ **then return** *unsuccessful* ;

slot s, Channel Search Algorithm (CSA) (Algorithm 4) is called to determine whether $(AB)_i$ may be scheduled in slot s. After attempting to schedule all the transmissions in Θ_{rready}, the set is updated as follows. First, all the successfully scheduled transmis-

ALGORITHM 4: Channel Search Algorithm (CSA)

Input: $(AB)_i$: transmission; s: time slot;

Output: returns channel c if $(AB)_i$ can be scheduled in slot s; otherwise returns -1

```
1  Procedure CSA((AB)_i, s)
2      /*mobile nodes are not allowed to be involved in data transmission during the slots
        reserved for beaconing*/
3      if (A ∈ M) and (slot s is used for beaconing) then
4          return −1 ;
5      /*check if A or B are involved in the transmission/reception of another flow in this
        time slot (half-duplex constraint) */
6      for every transmission (CD)_j scheduled in slot s do
7          if j ≠ i then
8              if A = C or A = D or B = C or B = D then
9                  return -1;
10     /*merge (AB)_i with another transmission of flow i */
11     else if exists an entry M[c, s] in which a transmission (CD)_i is scheduled then
12         return c;
13     /*return an empty channel */
14     else if exists a free entry M[c, s] then
15         return c;
16     else
17         return −1;
```

sions (i.e, those in $\Theta_{scheduled}$) are removed from Θ_{rready} (line 21). Next, consistent with flow coordination (i.e., Rule 1), for each transmission $(AB)_i$ scheduled in the current slot, the infrastructure transmissions from A's children to B are added to Θ_{rready} (line 22). For example, in Figure 3, if transmission $(BA)_i$ is scheduled in time slot 7, then transmissions $(DB)_i$ and $(CB)_i$ must be added to Θ_{rready} and they may be scheduled in a slot $s \leq 6$.

CSA determines if a transmission $(AB)_i$ can be scheduled in a slot s. This algorithm first checks whether scheduling this transmission conflicts with beacon transmissions. No mobile-to-infrastructure transmission can be scheduled in a beaconing slot since all mobile nodes must listen to beacon packets during this time. Next, CSA checks whether A or B are transmitting or receiving for another flow $j \neq i$ (line 6-9). If this is the case, then the transmission cannot be scheduled in the current slot since it would result in a conflict with flow j's transmissions. Otherwise, CSA uses the flow merging technique (Theorem 3.1) to merge $(AB)_i$ with other transmissions of i that have been scheduled (line 11). When merging is not possible, $(AB)_i$ may be assigned to an empty channel (line 14). CSA returns -1 when there is no free channel available in the current slot s.

Procedure `SecondPhase()` (in Algorithm 3) schedules mobile-to-infrastructure transmissions of an instance $J_{i,k}$. The procedure attempts to schedule mobile-to-infrastructure transmissions in slots $d_{i,k}$ down to $r_{i,k}$ and within one beaconing instance. The algorithm maintains a set of mobile-to-infrastructure transmissions that are reverse ready as Θ_{rready}. A transmission $(MA)_i$ is added to Θ_{rready} when we encounter an infrastructure-to-infrastructure transmission $(A*)_i$ that was assigned to slot s in the first phase (see line 40). The transmission $(MA)_i$ is removed from Θ_{rready} after it is successfully scheduled (see line 39).

The algorithm enforces that all mobile-to-infrastructure transmissions occur within a single beaconing instance using variables $bcurrent$ and $bperiod$. $bcurrent$ is the bea-

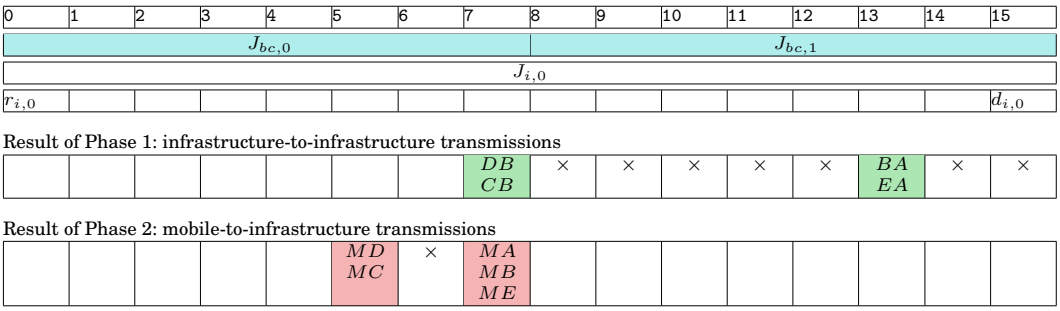


Fig. 11: An example of two-phase scheduling. For simplicity, we assume the slots marked by "×" cannot be used for scheduling flow i. $P_{bc} = 8$, and $P_i = 16$.

coning period of the current slot s (i.e., $bcurrent = \lfloor s/P_{bc} \rfloor$). $bperiod$ is the beaconing period to which the mobile-to-infrastructure transmissions are assigned. The constraint is that all mobile-to-infrastructure transmissions are assigned in the same beaconing period if at the end of scheduling all transmissions in Θ_{rready} the variables $bcurrent$ and $bperiod$ are equal. When $bcurrent \neq bperiod$, MARS must undo the partial schedule that was constructed thus far (see lines 35 – 37). This includes reverting the changes to the scheduling matrix and updating Θ_{rready} to include all the transmissions that became ready in slots $d_{i,k}, \ldots, s$.

5.1.1. Example. Figure 11 shows a sample scheduling generated by MARS for the scenario given in Figure 3. We assume that other higher priority flows have already been scheduled before i. The slots used by those flows are marked by the "×" symbol to indicate these slots cannot be used for scheduling flow i.

In Phase 1, the algorithm schedules infrastructure-to-infrastructure transmissions. Initially, the set of reverse-ready transmissions is $\Theta_{rready} = \{(BA)_i, (EA)_i\}$. We assume that node A is blocked in slots 15 and 14, and therefore, these transmissions cannot be scheduled in these slots. In slot 13, MARS schedules and combines $(BA)_i$ with $(EA)_i$. After this, the set of reverse-ready transmissions is $\Theta_{rready} = \{(DB)_i, (CB)_i\}$. In this example, we assume node B is blocked in slots 12 to 8, and therefore, the transmissions in the reverse ready set can be combined and scheduled in slot 7.

In Phase 2, the algorithm schedules mobile-to-infrastructure transmissions within one beaconing instance. However, not all of these transmissions can be scheduled during $J_{bc,1}$ because there are two infrastructure-to-infrastructure transmissions scheduled during $J_{bc,0}$. The transmissions $(MD)_i$ and $(MC)_i$ are reverse ready and may be scheduled in slots $s < 7$. Therefore, even if MARS could schedule $(MA)_i$, $(MB)_i$ and $(ME)_i$ during $J_{bc,1}$, the algorithm should retry placing them in $J_{bc,0}$. Transmissions $(MD)_i$ and $(MC)_i$ are scheduled is 5, which is the first unused slot smaller than 7.

5.1.2. Improvement. The performance of MARS can be improved when the period of flows are multiples of each other. In this case, we can generate a regular scheduling pattern that results in lower overhead of schedule dissemination. Assume that a transmission $(AB)_i$ is scheduled in time slot s_0 for instance $J_{i,0}$ of i. Having this, if $P_i < T$, then the scheduling algorithm can repeat the schedule in time slots $s_k = s_0 + k \times P_i$, where $k \in [1, T/P_i - 1]$. Furthermore, instead of disseminating the schedule computed for each instance of every flow, it is required to only disseminate the schedule established for the first instance of each flow. This is in particular an important improvement because most of the existing work on real-time wireless networking consider such relationship between flow periods [12; 46; 33; 58; 59; 60; 4].

As mentioned earlier, MARS schedules flows in the order of their priority. However, when the GW receives a bandwidth reservation request, MARS does not necessarily need to reschedule all the flows. As MARS schedules the flows in the order of their deadlines, scheduling a new flow only requires the scheduling of that flow as well as all the flows that have lower priority (i.e., longer deadline) than that of the new flow.

5.1.3. Shortcomings of MARS. MARS has two shortcomings that limit its ability to adapt to nodes joining and leaving the network. First, MARS may need to reconstruct full schedules in response to network dynamics. Aside from the computational time, this requires frequent dissemination of potentially large schedules to all the nodes. Second, 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 that in the current scheduling matrix $\mathcal{M}$, a transmission $(AB)_i$ happens after time slot s_k. If node A receives the new scheduling matrix $\mathcal{M}'$ at time s_k and immediately switches to the new schedule, transmission $(AB)_i$ will not happen in this hyper-period if it is scheduled before time slot s_k in the new matrix. To avoid packet loss, the safe switching time is at the beginning of the next hyper-period.

5.2. A-MARS: Additive Mobility-Aware Real-Time Scheduling

A-MARS is designed to schedule *a new flow without modifying the schedule of previously admitted flows*. This has two benefits: First, the GW needs to disseminate only the schedules of a newly admitted mobile node, and therefore, it introduces a lower overhead than MARS. Second, a newly admitted mobile node does not need to wait for a new hyper-period to join the network. The critical challenge of additive scheduling is to *schedule a flow without hindering the schedulability of future high-priority flows*.

A-MARS is developed under the assumption that the network services a set of flow classes ($\bar{\mathcal{F}} = \{\gamma, \alpha, \beta, ...\}$). A class γ is characterized by its period P_γ and deadline D_γ. Multiple flows may belong to a flow class. The priority of a flow class γ is computed as $1/D_\gamma$. For example, we assume that $D_\beta < D_\alpha < D_\gamma$, therefore, the priority of these flows is as follows: $\text{priority of } \gamma < \text{priority of } \alpha < \text{priority of } \beta$. Each flow i generated by a mobile node belongs to exactly one flow class (e.g., $i \in \gamma$). 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 $\sum_{\forall \gamma \in \bar{\mathcal{F}}} a_\gamma = 1$. In other words, the average ratio of flows belonging to a flow class γ to all the flows during the lifetime of the network is a_γ.

A-MARS is composed of a Slot Ordering Algorithm and a Flow Scheduling Algorithm. The Slot Ordering Algorithm constructs an *ordered slot list* U_γ for each flow class γ. The list reflects the order in which the scheduling algorithm will consider the slots that can be used for scheduling a flow belonging to class γ. Therefore, in contrast with MARS, which sequentially evaluates the schedulability of transmissions in slots $d_{i,k}$ to $r_{i,k}$, A-MARS evaluates transmissions schedulability based on the slot ordering given by list U_γ. The slot list is created to minimize the impact that scheduling γ in a slot has on the schedulability of higher-priority flows. At a high level, the order of slots is determined by measuring the effect of using each slot on the schedulability of higher priority flows. The Flow Scheduling Algorithm is a modified version of MARS to schedule flow instances using the ordered slot lists.

5.2.1. Slot Ordering Algorithm. For each flow class γ, the Slot Ordering Algorithm prepares a list that represents the order in which slots must be considered for scheduling each flow $i \in \gamma$. To measure how choosing a slot s for scheduling γ (in addition to the slots already in U_γ) would affect the schedulability of a higher-priority class α, we

define *potential utilization* as:

$$\begin{cases} PU_\alpha[s] = \frac{a_\alpha \times w_\alpha}{D_\alpha - (U_\gamma \cap \{r_{\alpha,k},\ldots,d_{\alpha,k}\})} & \text{if } \exists k \in [0, T/P_\alpha - 1], \\ & \text{s.t. } s \in [r_{\alpha,k}, d_{\alpha,k}]. \\ PU_\alpha[s] = 0 & \text{otherwise.} \end{cases} \quad (3)$$

where D_α is the deadline of flow class α, U_γ includes the slots already ordered for flow class γ, a_α is the likelihood of a flow belonging to a class α, and w_α is the number of transmissions required to schedule a flow belonging to class α. When $U_\gamma = \emptyset$, $PU_\alpha[s]$ simply reflects the probability of scheduling a transmission of flow class α in slot s. When U_γ is not empty, the PU of α increases as more slots are added to U_γ. In other words, adding more slots to U_γ results in a lower denominator as $U_\gamma \cap \{r_{\alpha,k}, ..., d_{\alpha,k}\}$ returns a higher value, and therefore, the potential utilization increases.

To clarify the concept of potential utilization, we consider the case when there are three flow classes $\bar{\mathcal{F}} = \{\gamma, \alpha, \beta\}$ in the network (see Figure 12):

$$P_\gamma = 32, D_\gamma = 28 \qquad P_\alpha = 16, D_\alpha = 10 \qquad P_\beta = 8, D_\beta = 6 \quad (4)$$

We want to prepare the ordered slot list for γ. For simplicity we assume $w_\gamma \times a_\gamma = w_\alpha \times a_\alpha = w_\beta \times a_\beta = 1$. As α and β have higher priority than γ, we need to compute and update PU vectors for α and β during the preparation of U_γ. Initially, none of the slots is marked as used (i.e., $U_\gamma = \emptyset$). Therefore, for example, $PU_\alpha[s] = 1/(10-0) = 0.1$ for $s \in [0, 9]$ and $s \in [16, 25]$. Also, $PU_\beta[s] = 1/(6-0) = 0.16$ for $s \in [0, 5]$, $s \in [8, 13]$, $s \in [16, 21]$ and $s \in [24, 29]$.

The Slot Ordering Algorithm is included as Algorithm 5. The algorithm constructs an ordered slot list U_γ for each flow class $\gamma \in \bar{\mathcal{F}}$. The variable $slots$ includes all the slots pertaining to γ that are left to be ordered. Initially, $slots$ includes all the slots belonging to the instances of γ (line 3). Then, the algorithm iterates through each slot s remaining in $slots$ to determine the impact that using that slot for scheduling γ has on a higher-priority class α. To quantify the effect of selecting a slot on the values of PU vectors, we define *accumulated differential potential utilization* (ADPU) as follows:

$$ADPU = \sum_{\forall PU_\alpha \in \mathcal{PU}} \left(\sum_{\forall s' \in slots \setminus s} (PU'_\alpha[s'] - PU_\alpha[s']) \right) \quad (5)$$

where $\mathcal{PU}$ includes the PU vector of flow classes with higher priority than γ, PU'_α is the PU vector of flow class α after marking slot s as used, and $slots$ is the set of slots not yet ordered. The slot s_{best} that has minimum impact (lowest ADPU) is added to the ordered list U_γ and is removed from $slots$.

Consider using the Slot Ordering Algorithm in the scenario given in Figure 12. In the first iteration, slots 14 and 15 result in $ADPU = 0$, as shown in `row 6.1`. The algorithm breaks the tie randomly and chooses slot 15, as `row 6.2` shows. Slot 14 is selected in the second iteration, as `row 6.3` shows. In iteration 3, slots 6, 7, 22 and 23 have the smallest $ADPU = 0.1$. Randomly, the algorithm chooses slot 23, as `row 6.4` shows. In iteration 4, slots 6 and 7 have an $ADPU = 0.1$, and the algorithm chooses slot 7. This process is repeated until all the slots have been ordered. The iteration number in which each slot is added to U_γ is given in Figure 13. It can be observed that,

$$U_\gamma = \{15, 14, 23, 7, 22, 6, 27, 13, 21, 5, 26, 12, 20, 4, 11, 25, 3, 19, 10, 2, 24, 18, 9, 1, 17, 8, 16, 0\} \quad (6)$$

It is worth noting that A-MARS's Slot Ordering Algorithm is run once before the actual network operation. As long as the set of flow classes (which is different from the

ALGORITHM 5: A-MARS's Slot Ordering Algorithm

Input: $\bar{\mathcal{F}}$: set of flow classes;
Output: a list of ordered slots (U_γ) for each flow class $\gamma \in \bar{\mathcal{F}}$;

begin
  for *every flow class* $\gamma \in \bar{\mathcal{F}}$ **do**
    $slots = \cup_{k=0}^{(T/P_\gamma)-1}[r_{\gamma,k}, d_{\gamma,k}]$; /*the set of slots that must be ordered*/
    $U_\gamma = \emptyset$; /*set of ordered slots for flow class γ */
    while $slot \neq \emptyset$ **do**
      $ADPU = 0$;
      /*compute the PU of flow classes with higher priority (i.e., shorter deadline)*/
      for *every* α *with priority higher than* γ **do**
        $PU_\alpha = \texttt{computePU}(\alpha, U_\gamma)$;
      /*measure the effect of choosing each slot $s \in slots$ on the schedulability of higher-priority flow classes*/
      for *every* $s \in slots$ **do**
        for *every* α *with priority higher than* γ **do**
          $PU'_\alpha = \texttt{computePU}(\alpha, U_\gamma \cup \{s\})$;
          $ADPU[s] = ADPU[s] + \sum_{s' \in slots \setminus \{s\}}(PU'_\alpha[s'] - PU_\alpha[s])$
      /*choose the slot with the minimum impact on the schedulability of higher-priority flow classes*/
      $s_{best} = \text{argmin}_{s \in slots} ADPU[s]$;
      $U_\gamma = U_\gamma \cup \{s_{best}\}$;
      $slots = slots \setminus \{s_{best}\}$;

/*computes the PU vector for flow class α, given the slots in U are marked as used*/
Procedure computePU(α, U)
  for $s = 0$ *to* $s = T - 1$ **do**
    if *exists* $k \in [0, T/P_\alpha - 1]$ *so that* $s \in [r_{\alpha,k}, d_{\alpha,k}]$ **then**
      $PU_\alpha[s] = \frac{a_\alpha \times w_\alpha}{D_\alpha - (U \cap \{r_{\alpha,k}, \ldots, d_{\alpha,k}\})}$
    else
      $PU_\alpha[s] = 0$;
  return PU_α;

set of flows generated by mobile nodes) is unchanged, we do not need to re-run the Slot Ordering Algorithm.

A property of the Slot Ordering Algorithm is that it evenly distributes the slots used for scheduling flows in class γ over the periods of flow classes with shorter periods, as the following property shows:

PROPERTY 1. *Assume U_γ is the list of ordered slots prepared for flow class γ. Also assume α is the flow class that its period is shorter than that of γ, and $k \times P_\alpha = P_\gamma$, where $k \in \mathbb{N}$. Choosing the first l slots of set U_γ always satisfies the following property:*

$$\text{minimize} \sum_{k=1}^{P_\gamma/P_\alpha} (n_k - n \times \frac{P_\alpha}{P_\gamma})^2 \tag{7}$$

where n_k is the number of slots conflicting with $J_{\alpha,k}$, and n is the number of slots conflicting with any of the instances of flow class α.

PROOF. We need to show that the maximum difference between the number of slots conflicting with any two instances of α is never higher than 1. Assume that the number

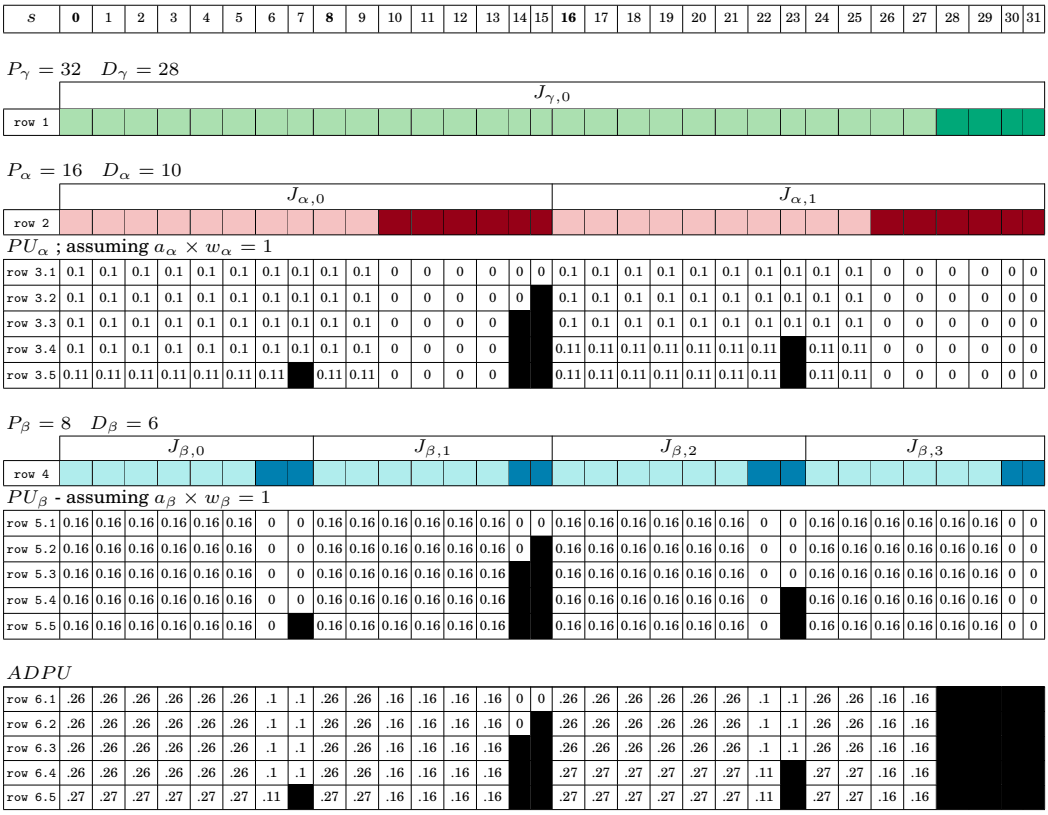


Fig. 12: Five iterations of A-MARS's Slot Ordering Algorithm. We are interested to compute the list of ordered slots for flow class γ. `row 3.1-3.5` show the changes of PU_α during five iterations. `row 5.1-5.5` show the changes of PU_β during five iterations. `row 6.1-6.5` show the changes of $ADPU$ during five iterations.

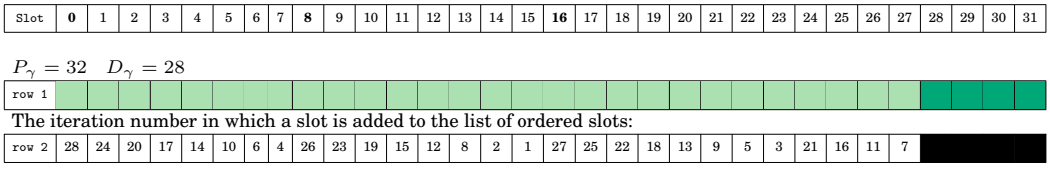


Fig. 13: The ordering of slots for $J_{\gamma,0}$ in Figure 12.

of conflicting slots with $J_{\alpha,k}$ and $J_{\alpha,k+1}$ are denoted by n_k and n_{k+1}. Also assume that selecting a slot from $J_{\alpha,k}$ results in $n_k - n_{k+1} \geq 2$. We prove that this never happens. When $n_k = n_{k+1}+1$, choosing a slot from $J_{\alpha,k}$ results in $1/(D_\alpha - n_k)$ increase in potential utilization, and choosing a new slot from $J_{\alpha,k+1}$ results in $1/(D_\alpha - n_{k+1})$ increase in potential utilization. As $n_k > n_{k+1}$, choosing a slot from instance $k+1$ results in a lower $ADPU$. □

5.2.2. Flow Scheduling Algorithm. After the completion of the Slot Ordering Algorithm, the Flow Scheduling Algorithm reserves bandwidth for new flows additively. Consider the scheduling of the transmissions of an instance $J_{i,k}$ where i belongs to flow class γ. First, the algorithm identifies list U_γ, and extracts $U_{\gamma,k}$, which orders the slots in the range $r_{i,k}, \ldots, d_{i,k}$. The ordering of slots in list $U_{\gamma,k}$ reflects the impact of using each

slot on the schedulability of flows with higher priority. However, before using the slots in $U_{\gamma,k}$, a new list Δ is constructed, which reorders the slots in decreasing temporal order to facilitate the use of reverse scheduling. The list Δ includes the first δ slots in $U_{\gamma,k}$, ordered in reverse temporal order. Initially, $\delta = 1$, which means only the first slot in $U_{\gamma,k}$ is used by the algorithm. After each unsuccessful round of scheduling, one more slot of $U_{\gamma,k}$ is added to Δ. Note that Δ must be ordered in descending order of slot numbers because of using reverse scheduling. This process is repeated until either the flow instance $J_{i,k}$ is successfully scheduled, or the algorithm fails when all the slots in $U_{\gamma,k}$ have been consumed without successful scheduling.

For U_γ prepared in Figure 12, if scheduling a flow belonging to this class fails for five rounds, then the generated Δ lists until successful scheduling are:

1^{st} round (failed): $\Delta = \{15\}$ 2^{nd} round (failed): $\Delta = \{15, 14\}$
3^{rd} round (failed): $\Delta = \{23, 15, 14\}$ 4^{th} round (failed): $\Delta = \{23, 15, 14, 7\}$
5^{th} round (failed): $\Delta = \{23, 22, 15, 14, 7\}$ 6^{th} round (success): $\Delta = \{23, 22, 15, 14, 7, 6\}$

5.2.3. Example. Figure 14 shows the scheduling matrix generated for a particular REWIMO network when A-MARS is used. We have decomposed the matrix into several sub-matrices to clearly show the schedules established for data transmission and management purposes. In this example, there are two flow classes $\bar{\mathcal{F}} = \{\gamma, \alpha\}$, where $P_\gamma = D_\gamma = 256$, and $P_\alpha = D_\alpha = 128$. We also assume that the network has scheduled a flow $i \in \gamma$. We have decomposed the schedules established for i into two sub-matrices (e) and (f), where they show the infrastructure-to-infrastructure and mobile-to-infrastructure transmissions, respectively. We assume there are 16 radio channels available, the network has 23 infrastructure nodes, and $P_{bc} = P_{rq} = P_{ct} = P_{rp} = 128$.

The following observations can be made: First, as Figure 14(a) shows, the Beacon Scheduling Algorithm (Algorithm 2) schedules all the beaconing slots during the first 23 slots of each beaconing instance. Second, a request reception (rq) slot is scheduled in slot 23 and slot 151 (see Figure 14(b)). During these slots, all the infrastructure nodes listen to the channel to receive the join requests sent by mobile nodes seeking network admission. Third, a control flow (ct) is scheduled from the GW towards the infrastructure nodes, as Figure 14(c) shows. This flow is used for the dissemination of transmission schedules. Fourth, a report flow (rp) is scheduled from every infrastructure node towards the GW, as Figure 14(d) shows. This flow conveys mobile nodes' admission requests to the GW. Fifth, the transmissions of i are scheduled so that they cause minimum conflict with the scheduling of flows belonging to α. Therefore, as Figure 14(e) shows, infrastructure-to-infrastructure transmissions have not been accumulated near time slot 255; rather, they have been distributed with respect to the instances of flow class α. Additionally, as some infrastructure-to-infrastructure transmissions have been scheduled during the interval $[0, 127]$, mobile-to-infrastructure transmissions cannot be fully scheduled during the interval $[128, 255]$. Therefore, as Figure 14(f) shows, mobile-to-infrastructure transmissions have been all scheduled during beacon period $[0, 127]$, using the two-phase scheduling technique.

6. PERFORMANCE EVALUATION

In this section we present the implementation details of our simulator as well as the performance study of REWIMO.

6.1. Simulation Tool and Benchmarks

Using the OMNeT++ Discrete-Event Simulation Framework [61] we developed a sophisticated simulator that implements the details of a real-world deployment. In ad-

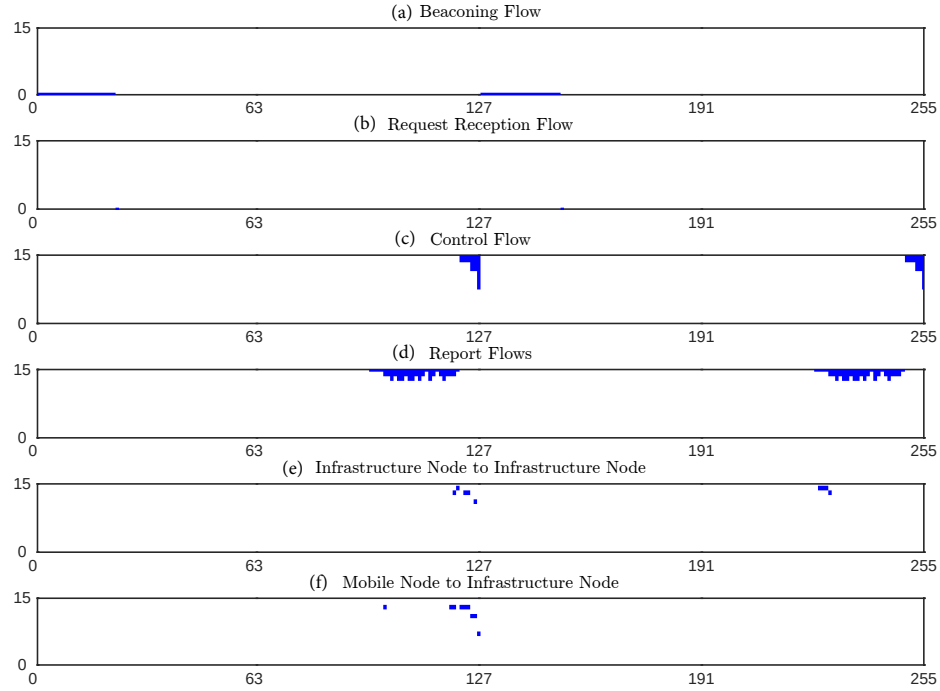


Fig. 14: A scheduling matrix computed by A-MARS. The scheduling matrix has been decomposed into six sub-matrices to clearly show the schedules established for data transmission and management purposes. We assume the network services two flow classes: $\bar{\mathcal{F}} = \{\gamma, \alpha\}$, where $P_\gamma = D_\gamma = 256$ and $P_\alpha = D_\alpha = 128$. We assume only one flow has been scheduled, and this flow is $i \in \gamma$.

dition, the developed simulator uses the packet reception traces provided as part of the MoteTrack project [2; 1] to realistically model wireless communications. Figure 15 shows the architecture of the developed simulator.

The MoteTrack traces, which are used by the Wireless Channel module, were collected using MicaZ nodes equipped with CC2420 [55] radios that transmit at 0dBm over the 16 channels that are available on 802.15.4 radios. The data was collected by having 23 infrastructure nodes transmit packets. A mobile node was placed at 358 locations within a building and recorded link quality statistics including the received signal strength (RSSI) for each channel.

Figure 16 shows the network topology, mobility paths, and the routing graph. The routing graph, which is a spanning tree, is constructed using [62] and [63]. As Figure 15 shows, an infrastructure node denoted as "Infrastructure Node 1" is the root of the routing tree and is connected to the Gateway module through a wire link. The root node delivers data flows (originated by mobile nodes) and report flows (originated by infrastructure nodes) to the Gateway module through this link. The root node periodically receives control flow from the Gateway module (for schedule dissemination and time synchronization purposes) to be distributed in the network. Other infrastructure nodes (i.e., Infrastructure Node 2 through n) communicate with the root node and mobile nodes only through the Wireless Channel module.

The Gateway module distributes a newly-computed schedule through the control flow. The control flow is first sent to the root of the tree, and then that node distributes the received schedules. The root node sends to each child the control packets that in-

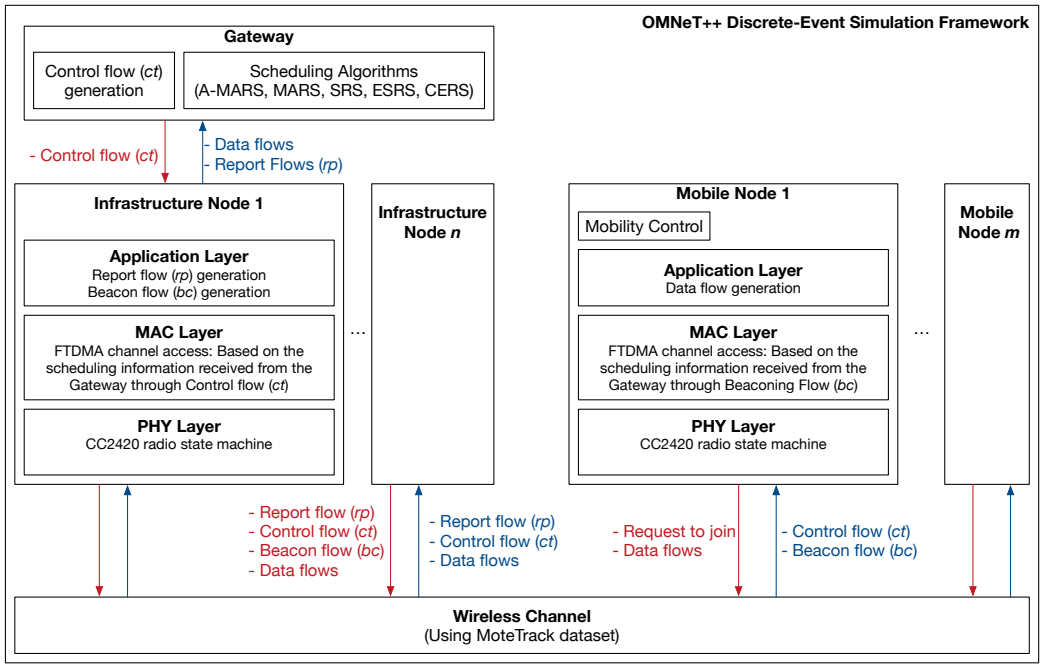


Fig. 15: The architecture of the developed simulator.

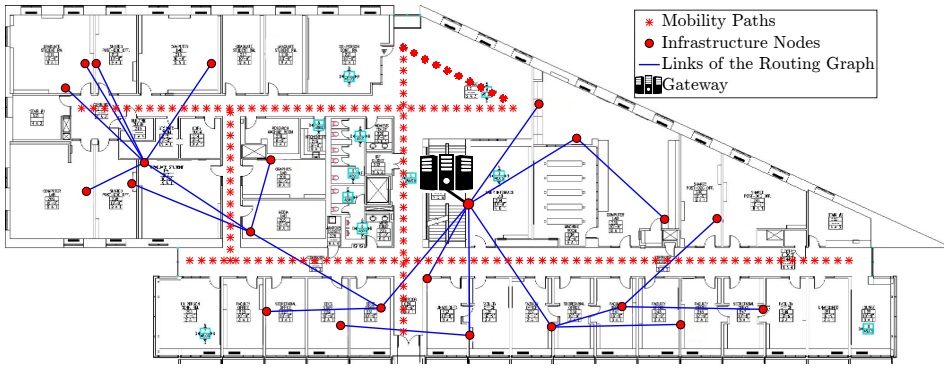


Fig. 16: The network used for performance evaluation.

clude the schedules belonging to that child and its descendants. Other nodes distribute the received schedule in a similar manner. Nodes also encode transmission schedules to reduce the overhead of schedule dissemination. The encoding mechanism is demonstrated in Figure 17. This structure is employed due to the following reasons. In the first level, we used slot number because the number of bits used to identify a slot is more than that of channel number and flow number. For example, when the hyperperiod duration is 800 slots, we need to use 10 bits for slot number, while only 4 bits are used to specify a channel number when 16 channels exist. Since multiple transmissions of a flow may be combined in an entry of the scheduling matrix, transmissions are in the deepest level of the tree, while their parent is the flow they belong to. Note

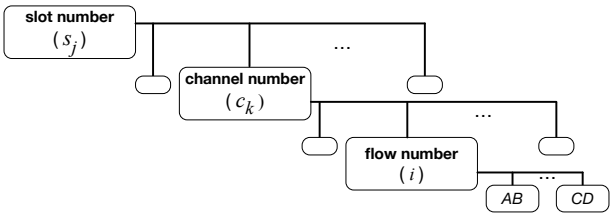


Fig. 17: The approach employed for encoding scheduling information.

that, as at most one flow can be scheduled in a given time slot and channel, the second level of the tree can be either channel number or flow number; we chose channel number.

Using this approach, for example, we schedule `row 4` of Table 5 as follows:

$$[s_5][c_1][i][MC, MD][s_6][c_1][i][ME, MB, DB, CB][s_7][c_1][i][EA, BA, MA] \quad (8)$$

The number of bits for channel identification is 4 (the 802.15.4 standard uses 16 channels). The number of bits for slot number, flow number and sender/receiver address are computed as $\lfloor \log_2 n \rfloor + 1$, where n can be the hyper-period, number of flows or number of nodes. In addition, we also include two data identification bits before a slot number, channel number, flow number and a pair of sender/receiver addresses. Note that we used two bits as there are four types of data used to convey scheduling information: (i) slot number (e.g., s_1), (ii) channel number (e.g., c_1), (iii) flow number (e.g., i), and (iv) transmission (e.g., AB).

Initially, when there is no FTDMA schedule established, infrastructure nodes use a CSMA access mechanism on channel 26 to exchange the control flow generated by the Gateway module. The result of this initialization phase is the establishment of a FTDMA schedule for report flows, control flow and beacon flow. Later, when a mobile node wants to join the network, infrastructure nodes disseminate the new schedule received from the Gateway module through the schedule established for control flow dissemination. When a FTDMA schedule is established, each node uses at its MAC layer a timer that counts every 10ms, which is the duration of each time slot. The timer counts from 0 to $T - 1$ (i.e., the hyper-period length) periodically.

We have emulated the motion of users (mobile nodes) moving along the hallways of the building. The mobility of each mobile node is controlled independently and through its Mobility Control module. The initial position of each mobile node is randomly selected on the mobility paths. After a mobile node has joined the network, it starts moving on a mobility path until reaching the intersection of two paths. At that point, either a new path is selected or movement on the current path is continued. The moving direction is reversed when a node reaches the end of a path. The movement speed is 1 m/s, unless mentioned otherwise.

During the simulations, the Wireless Channel module estimates the link quality of mobile nodes to infrastructure nodes by first identifying the closest location to the position of the mobile node for which we have data in the MoteTrack dataset. We then use as the RSSI of the link a random sample selected from the available trace on the channel on which the transmission is performed. This mechanism has been implemented using a hash function that maps a location into an array entry that points to an array of RSSI samples. The RSSI sample is then used by the Wireless Channel module to obtain an estimated PRR using the equations mentioned in [64]. The PPR value is then used to decide about the delivery of the packet to the destination node. A similar approach is employed at the MAC Layer module of mobile nodes to estimate their link quality to infrastructure nodes and decide about link activation.

Table II: The baseline algorithms compared against MARS and A-MARS.

Static Real-Time Scheduling (*-SRS)
These algorithms represent the approaches that satisfy real-time communications in static multi-hop networks. As discussed in Section 3, these algorithms do not benefit from Rule 1, Theorem 3.1 and Rule 3.2. Please refer to Section 7.1 for a summary of published works in this category.
• *Earliest Deadline First-SRS* (EDF-SRS): The schedulability of ready transmissions is evaluated in the order of their absolute deadlines. • *Deadline Monotonic-SRS* (DM-SRS): The schedulability of ready transmissions is evaluated in the order of their relative deadlines. • *Least-Laxity First-SRS* (LLF-SRS): The schedulability of ready transmissions is evaluated in the order of their laxities. Laxity of a transmission (AB) 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 minimum number of time slots (i.e., hops) required to forward a packet from node A to the GW.
Enhanced Static Real-Time Scheduling (*-ESRS)
ESRS algorithms are the enhancement of SRS algorithms with Rule 1.
• EDF-ESRS: Adds Rule 1 to EDF-SRS • DM-ESRS: Adds Rule 1 to DM-SRS • LLF-ESRS: Adds Rule 1 to LLF-SRS
Combination-Enabled Real-Time Scheduling (*-CERS)
CERS algorithms are the enhancement of SRS algorithms with Rule 1 and Theorem 3.1.
• EDF-CERS: Adds Rule 1 and Theorem 3.1 to EDF-SRS • DM-CERS: Adds Rule 1 and Theorem 3.1 to DM-SRS • LLF-CERS: Adds Rule 1 and Theorem 3.1 to LLF-SRS

As mentioned in Section 2, both control flow and beaconing flow are used for time synchronization. Although our simulation tool does not include the implementation of a time synchronization protocol, we consider a 1ms clock drift due to the inaccuracy of time synchronization [41]. At the MAC layer, we cope with time synchronization errors through using a 1ms guard time at the beginning of each time slot. For clock accuracy $\pm$10ppm and synchronization error 50μs, we need to use inequality $(P_{ct}+P_{bc})\times(time\ slot\ duration)<48$ (seconds) to schedule control flow and beaconing flow to achieve guard time $t_g=$ 1ms [41]. The scheduling algorithms implemented by the Gateway module satisfy this inequality. In a reception time slot a node waits for $2t_g+160\mu s$ and transitions to the sleep mode if no packet is detected. Note that since we employed 802.15.4-compatible packets, $160\mu s$ is the preamble+SFD transmission duration [64].

To measure energy consumption, we implemented the radio state machine and energy consumption characteristics of CC2420 [64] in the PHY Layer module. It is worth mentioning that energy measurement takes into account the effect of all protocol overheads associated with time synchronization, schedule dissemination, beaconing and request to join the network.

We consider two flow generation patterns:

— *Homogeneous*: There is only one flow class ($\bar{\mathcal{F}}=\{\gamma\}$). Therefore, all the mobile nodes generate data flows with similar period and deadline.
— *Heterogeneous*: Three flow classes exist ($\bar{\mathcal{F}}=\{\gamma,\alpha,\beta\}$). Each mobile node generates a flow belonging to a flow class randomly chosen from the set of available flow classes. For these flow classes we assume $a_\gamma=a_\alpha=a_\beta=1/3$.

The baseline algorithms compared against MARS and A-MARS are presented in Table II. We repeated the experiment 20 times for each configuration, and report the median, lower quartile and higher quartile. The performance evaluation parameters are given in Table III. We set the beacon period $P_{bc}=512$ (slots) when the flow periods are in the form of 2^n, and $P_{bc}=800$ (slots) when the flow periods are in the form

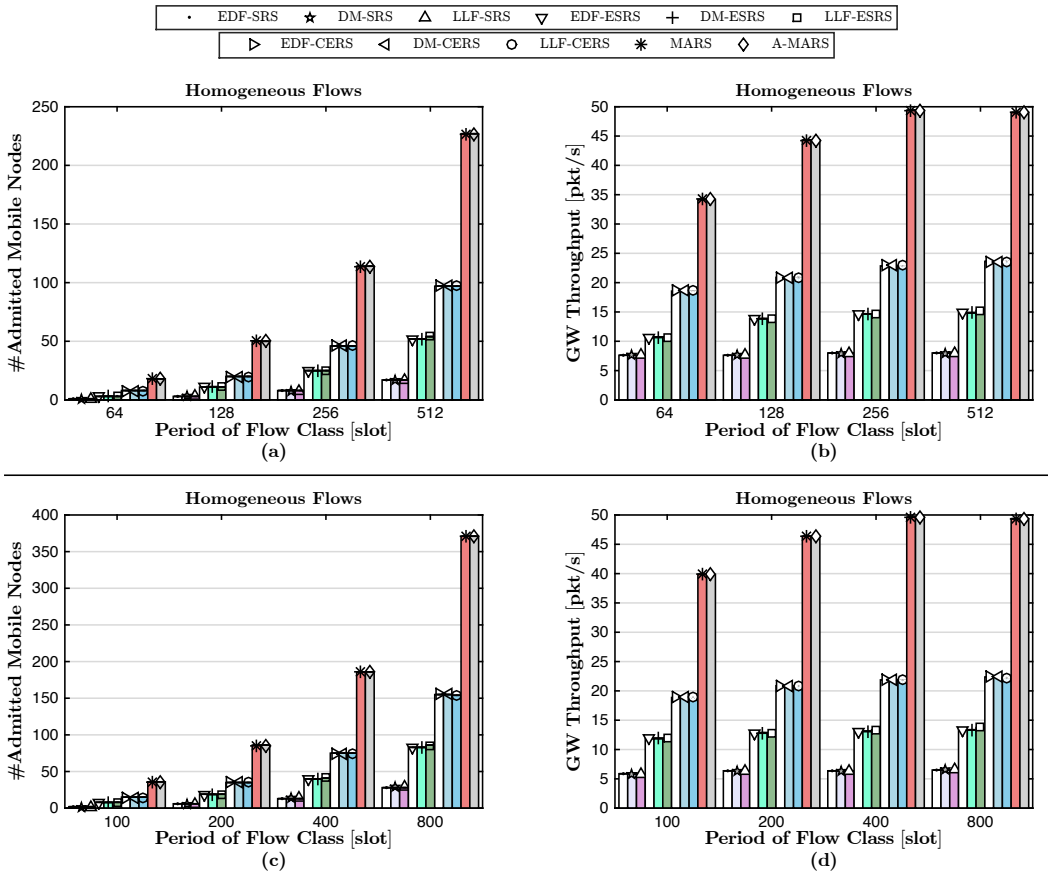


Fig. 18: Maximum number of admitted mobile nodes and GW throughput when the traffic pattern is homogeneous. As the flows are homogeneous, there is only one flow class in the network, and a value on the x-axis shows the period and deadline of the flow class. Compared to SRS algorithms, the increase in the number of admitted mobile nodes achieved by ESRS, CERS and MARS algorithms are 3x, 5.5x, and 14x, respectively.

Table III: General performance evaluation parameters

Frame Format		
Packet Format: 802.15.4	Max Packet Size: 127B	Max Payload Size: 108B
Radio		
Speed: 250kbps	Channels: 11-26	Transmission Power = 0dBm
Rx Power = 19.7mA	Tx Power = 17.4mA	Power Down = 20μA
Other Parameters		
Battery: 2500mAh 3V	Time Slot = 10ms	Guard Time (t_g) = 1ms

of $100 \times n$. For the MoteTrack deployment, a beacon period less than 1000 slots is necessary to ensure reliable communications ($> 95\%$).

6.2. Results and Discussions

6.2.1. Real-time Capacity. To evaluate the efficiency of bandwidth reservation, we measured *the number of mobile nodes admitted* and *GW throughput*. The number of mobile nodes admitted reflects the efficiency of the scheduling algorithms in terms of real-time

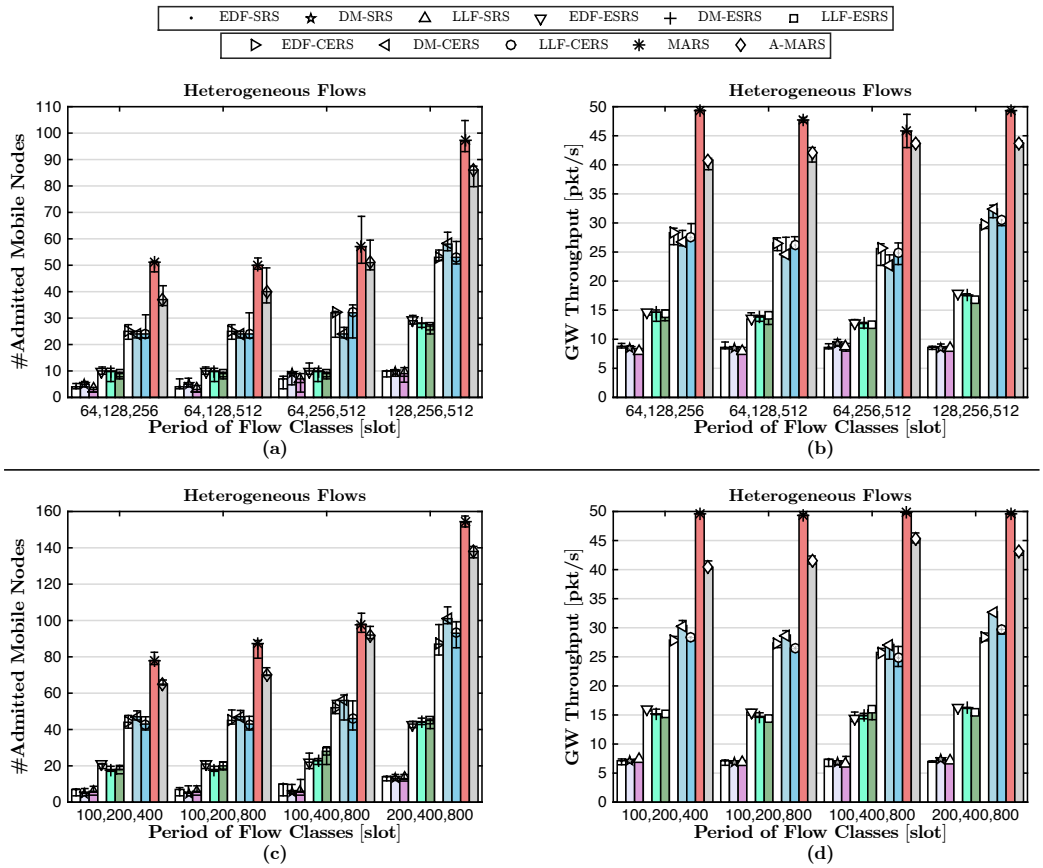


Fig. 19: Maximum number of mobile nodes and GW throughput when the traffic pattern is heterogeneous. The values on the x-axis show the period and deadline of flow classes. For example, "100, 200, 400" indicates $P_\beta = D_\beta = 100$, $P_\alpha = D_\alpha = 200$, and $P_\gamma = D_\gamma = 400$. Compared to SRS algorithms, the increase in the number of admitted mobile nodes achieved by ESRS, CERS and MARS algorithms is about 3.2x, 7x, and 14X, respectively. Additionally, A-MARS shows less than 15% reduction in the number of admitted mobile nodes, compared to MARS.

capacity. The GW throughput is defined as the total number of packets exchanged with the GW per second.

Figures 18 and 19 show the results for homogeneous and heterogeneous traffic patterns, respectively. These figures show that the algorithms designed for static real-time wireless networks (i.e., the SRS algorithms, please see Section 7.1) have low real-time capacity when applied to mobile networks. This is because these algorithms do not benefit from flow coordination, flow merging and reverse scheduling techniques to efficiently reserve bandwidth over multiple potential communication paths. The results show that using flow coordination (Rule 1) and flow merging (Theorem 3.1) in CERS algorithms increases the number of admitted mobile nodes by more than 6x compared with SRS algorithms. The performance improvements due to reverse scheduling may be evaluated by comparing the MARS and CERS algorithms. Specifically, the MARS algorithms increase the number of admitted mobile nodes by 2.5x and enhance bandwidth utilization by up to 120% compared to the CERS algorithms. Although A-MARS relies on a predictive strategy to schedule flows additively, our results show that this

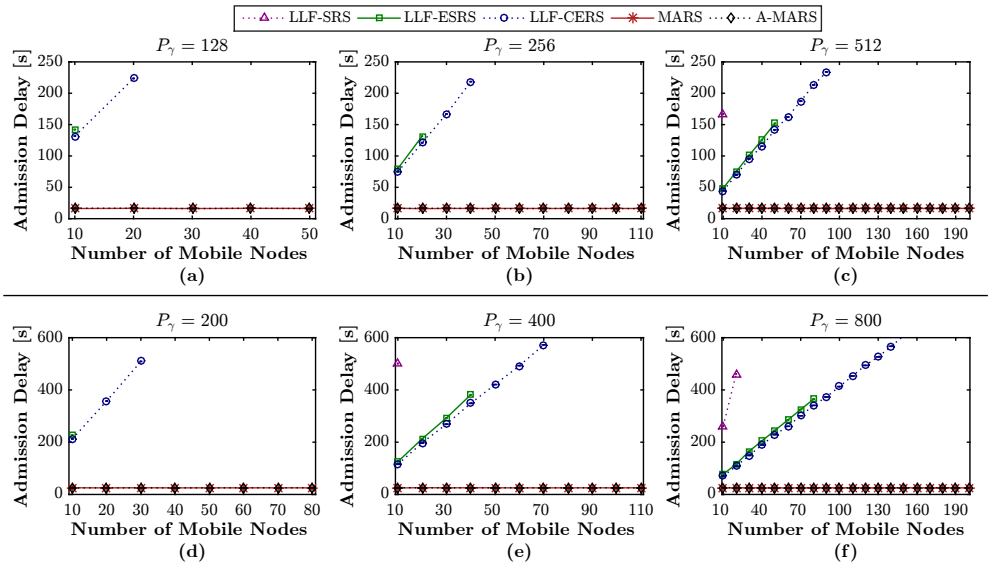


Fig. 20: Admission delay of the scheduling algorithms when the traffic pattern is homogeneous. The value above each figure shows the period (and deadline) of flow class. For example, "100" means all the nodes' data flows belong to flow class γ, where $P_\gamma = D_\gamma = 100$. The admission delays of MARS and A-MARS are less than 20 seconds.

algorithm results in a slight decrease in network capacity: while the bandwidth reservation efficiency of A-MARS is as good as MARS with homogeneous flows, it introduces less than 15% reduction in the number of admitted mobile nodes when heterogeneous flows are used. The performance of A-MARS is the same as that of MARS when the traffic pattern is homogeneous because, when there is only one flow class, the two algorithms consider transmissions for scheduling in the same order and, as a result, will construct the same schedule.

6.2.2. Admission Delay. The admission of a mobile node requires the dissemination of new transmission schedules to the nodes, and the overhead of this dissemination depends on the scheduling algorithm used. In particular, if bandwidth reservation for a new flow requires rescheduling existing flows, then the amount of control data disseminated increases with the number of mobile nodes admitted. Having this in mind, A-MARS has been designed to perform bandwidth reservation without modifying existing schedules.

Figures 20 and 21 show the admission delay achieved with the scheduling algorithms under various traffic patterns. Note that we did not present the results for DM and EDF strategies, as their performance is similar to that of LLF. The results indicate that the admission delay achieved with A-MARS is lower than that of other approaches, and its delay is independent of the number of mobile nodes and the characteristics of their generated flows.

Using MARS, when a request for flow admission is received at the GW, the algorithm reschedules all the existing flows with their deadline larger than that of the new flow; thereby, the admission delay depends on the number and characteristics of flow classes. As Figure 20 shows, the admission delay of MARS is the same as that of A-MARS when the traffic pattern is homogeneous. This is because the request for scheduling a data flow only requires the scheduling of that flow. In this case, the optimization described

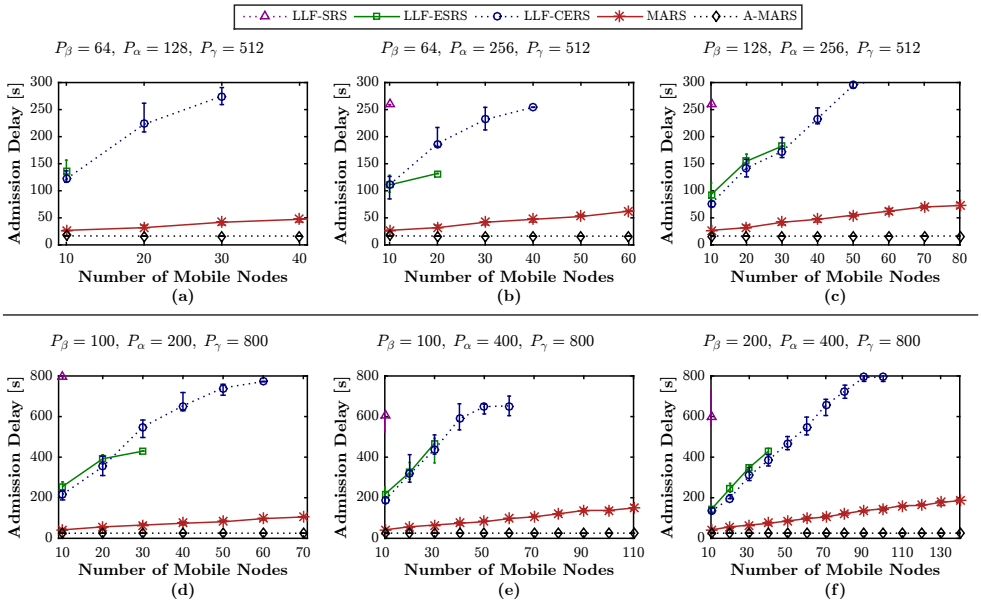


Fig. 21: Admission delay of the scheduling algorithms when the traffic pattern is heterogeneous. The values above each figure show the period (and deadline) of the three flow classes. The admission delay of A-MARS is always less than 20 seconds, and it is independent of the number of mobile nodes admitted.

in Section 5.1.2 may be used to reduce the admission delay. In contrast, the admission delay of MARS increases with the number of mobile nodes when the traffic pattern is heterogeneous, as Figure 21 shows. Note that since the flows' periods are multiplicative in this scenario, MARS is still able to use the aforementioned optimization. In sharp contrast to MARS, the admission delay of A-MARS is independent of the number of mobile nodes. This is because A-MARS requires only scheduling the flows of a newly admitted node. This has significant performance benefits, particularly when there are numerous mobile nodes. For example, in Figure 21(f), the admission delay of MARS is about 7x higher than that of A-MARS.

The significantly higher delay of the SRS, ESRS and CERS algorithms under both workloads is because these algorithms reschedule all the flows whenever a request for flow admission arrives. Therefore, they cannot benefit from the technique explained in Section 5.1.2 to reduce the overhead of schedule dissemination, as the scheduling patterns of a flow during its instances may not be similar. For these algorithms, in addition, Figures 20(a), (b), and (c) show that admission delay increases when flow period reduces. This is because when the ratio of hyper-period to data flow period increases, the number of instances in which a data flow needs to be scheduled increases as well; as a result, the amount of data disseminated by the GW is increased. For example, when the period of flow class equals 128 slots, the GW must distribute the schedule of each flow during its four instances ($512/128$), where 512 is the hyper-period duration. On the other hand, the schedule of only one flow instance must be disseminated when the period of flow class equals 512 slots. It is worth noting that admission delay can be reduced through shortening the period of beaconing flow (bc), request reception flow (rq), reporting flows (rp) and control flow (ct). However, reducing their periods introduces a higher level of network resources reserved for control purposes.

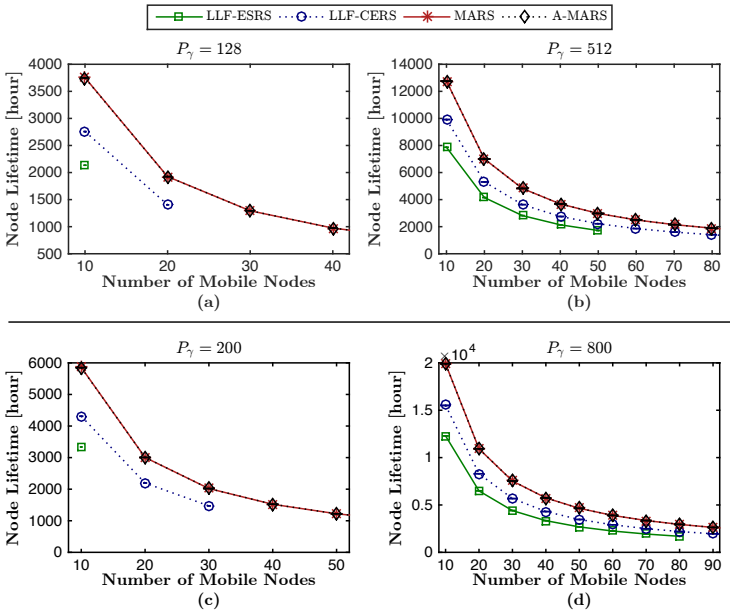


Fig. 22: Average node lifetime achieved by the scheduling algorithms when the traffic pattern is homogeneous.

6.2.3. Lifetime. In this section we show how the scheduling algorithms affect the energy consumption of nodes. Figures 22 and 23 present the average lifetime of infrastructure nodes versus the number of admitted mobile nodes. We do not report the lifetime of mobile nodes because it was higher than that of infrastructure nodes. We increased the number of mobile nodes in steps of 10 and measured steady-state energy consumption.

Given a flow class and a number of mobile nodes, the time spent in transmit mode is independent of the scheduling algorithm used because, as Section 3 shows, mobility results in the activation of one path during an instance of a flow. However, the scheduling algorithm employed affects the number of slots in which infrastructure nodes expect packet reception. In particular, the flow merging technique proposed in this paper reduces the number of slots in which infrastructure nodes turn on their radio to detect and receive potentially incoming packets. Additionally, the number of such slots is further reduced using the reverse scheduling technique. For example, in Figure 5, the number of slots in which A waits for receiving a packet of flow i is 3 using forward scheduling (see `row 3`), and this value is reduced to 1 using reverse scheduling (see `row 4`).

6.2.4. Effect of Two-Phase Scheduling on Reliability. As explained in Section 4, the lack of coordination between the scheduling of beacons and data flows may result in mobile nodes being unable to associate with infrastructure nodes over high-quality links.

Measuring the effect of two-phase scheduling on the reliability of packet transmission is not straightforward because, its effect is revealed only when the scheduling algorithm cannot schedule all mobile-to-infrastructure transmissions during one beaconing instance. This condition happens when the number of admitted nodes is beyond a certain number for a given configuration. Therefore, our measurement strategy is as follows: For a given beaconing period and data flow period, we measure the average

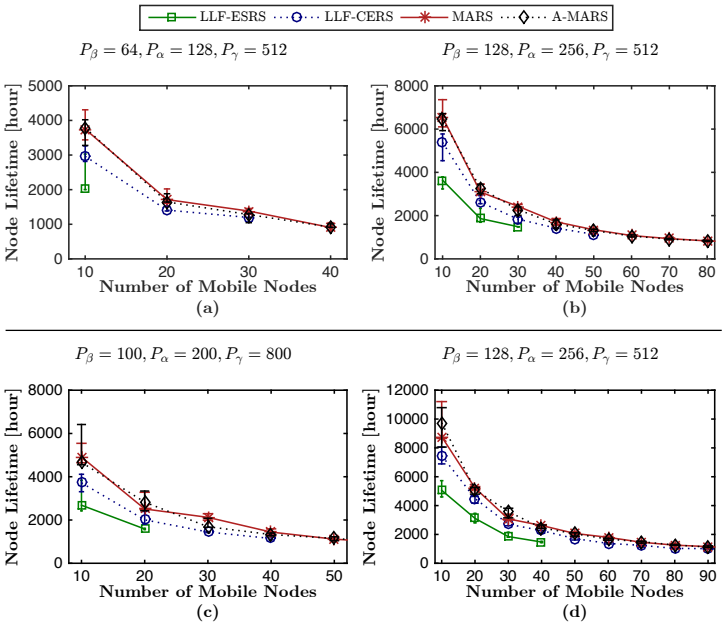


Fig. 23: Average node lifetime achieved by the scheduling algorithms when the traffic pattern is heterogeneous. The lifetime achieved with MARS algorithms is higher than that of other algorithms because MARS algorithms reduce the number of slots in which nodes wake up and listen for packet reception.

quality of the ten least-quality links over which mobile nodes associate with infrastructure nodes during each instance of the data flow. We adjust beacon period based on mobility speed to properly reflect neighborhood changes. The data flow period is set to be a multiple of the beaconing period to bring the possibility of placing the transmission schedules of an instance of data flow over multiple beaconing instances. Our evaluation results are presented in Figure 24. The beacon period has been set to 1000 and 500 to reflect the changes in neighborhood when the mobility speed is 1 m/s and 2 m/s, respectively.

Figure 24(a) shows that as soon as the period of data flow is increased to 3000, packets are sent over links with about 80% reliability when the two-phase scheduling technique is not employed. Reliability is further reduced as the period of data flow is increased because this enhances the chance of placing mobile-to-infrastructure transmissions over multiple beaconing instances. Figure 24(b) shows a similar behavior at shorter values of data flow period. These results confirm that reliability reduces as a function of mobility speed and the ratio of data flow period to beacon period (i.e., P_i/P_{bc}) when the two-phase scheduling technique is not employed. As both timeliness and reliability are the essential characteristics of mobile mission-critical applications, our results confirm the importance of coordination between beaconing and data scheduling.

Discussion. In this paper, we addressed reliability from the association point of view. Other mechanisms such as multi-path data forwarding (e.g., [12; 13]) can be integrated into REWIMO to further improve reliability. Such mechanisms may be necessary for situations such as an area not perfectly covered by infrastructure nodes, or a harsh industrial environment with high interference level. To this end, we can group nearby nodes and then schedule *mobile node-to-group transmissions* so that multiple

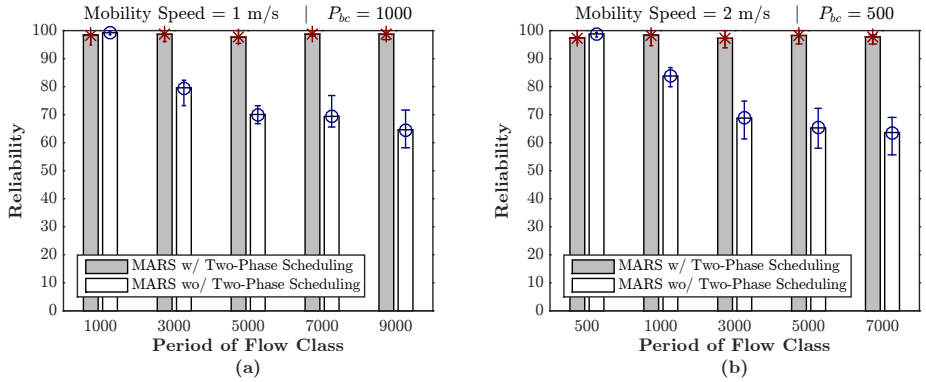


Fig. 24: Effect of two-phase scheduling on the reliability of data transmission. These results show the possibility of mobile-to-infrastructure transmissions over links with an unacceptable level of reliability when the two-phase scheduling technique is not enforced.

infrastructure nodes receive a data packet sent by a mobile node during a period of its data flow. In this case, for example, two mobile nodes A and B receive a packet concurrently and forward it towards the GW. A design consideration is to decide if the duplicates will be merged at some point before reaching the GW, or two copies are delivered to the GW separately. This design consideration affects the efficiency of flow merging and flow coordination techniques. For example, if two nodes A and B are allowed to forward to a node C during a period of a data flow i, then transmissions $(AC)_i$ and $(BC)_i$ cannot be combined. Note that if we do not apply a grouping strategy on the infrastructure nodes enabled to concurrently receive a packet from a mobile node, then we unnecessarily increase the maximum number of concurrently active paths up to the total number of infrastructure nodes. However, this is both unrealistic and unnecessary because only the infrastructure nodes near a mobile node can receive a packet concurrently. Nevertheless, such an unrealistic multi-path forwarding strategy will reduce the network capacity of REWIMO to that of the SRS algorithms. In addition to multi-path forwarding, an another method to enhance reliability is to extend the duration of time slots to include multiple retransmissions.

6.2.5. Algorithm Execution Time. Figure 25 shows the execution duration of algorithms using an i7-4980HQ processor[9]. Note that the reported values do not include the execution duration of A-MARS's Slot Ordering Algorithm as it is executed only once before the actual network operation begins. We varied the relationship between the period of beaconing and data flow to measure the effect of two-phase scheduling on schedule computation duration. In particular, the transmissions of an instance of a data flow may be placed over multiple beaconing instances when $P_{bc} = 128$ or $P_{bc} = 200$, and the algorithm may need to evaluate the schedulability of mobile-to-infrastructure transmissions during more than one beaconing instance.

The higher execution duration of A-MARS compared to MARS is due to its iterative evaluation of schedulability using the lists of ordered slots. For example, when a flow $i \in \gamma$, where $P_\gamma = 800$, arrives, A-MARS may need to evaluate the schedulability of this flow for up to 800 iterations using the slots in U_γ. Therefore, the number of mo-

[9]In real deployments, a component called Network Manager is responsible for schedule computation, and Gateway acts as an interface between the wireless and wired parts of the network. Both these components are referred to as the Gateway (GW) in this paper. As Network Managers are similar to (or even stronger than) regular PCs, we believe that these results are representative of a real deployment.

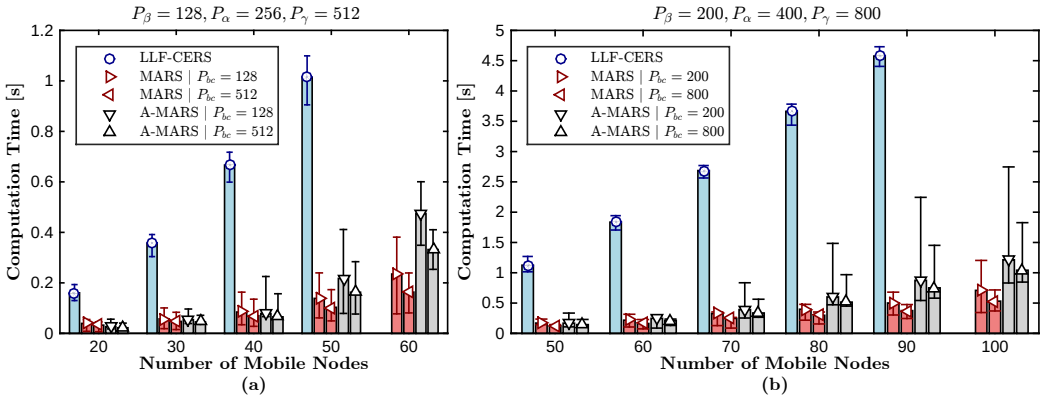


Fig. 25: The computation duration of the scheduling algorithms when multiple flow classes exist.

bile nodes and the characteristics of flow classes affect execution time. The variations observed in the execution duration of MARS are subject to flow arrival pattern. For example, when a flow $i \in \beta$, where $P_\beta = 200$, arrives, MARS needs to reschedule all the flows belonging to flow classes α and γ.

Figure 25 also shows that the execution time of MARS and A-MARS are higher when the schedules of an instance of a data flow may be placed over multiple beaconing instances. The reason is that if scheduling all mobile-to-infrastructure transmissions was not successful during a beaconing instance, the algorithm retries scheduling all these transmissions in the next beaconing instance. Therefore, as the number of admitted mobile nodes increases, the scheduling matrix becomes denser and the number of retries increases.

The significantly higher execution duration of CERS is due to two reasons: First, CERS (and also ESRS and SRS, which have not been shown in this figure) reschedules all the flows when a request for flow admission arrives. Second, these algorithms cannot benefit from the technique mentioned in Section 5.1.2 to reduce the overhead of schedule computation and schedule dissemination.

7. RELATED WORK

The problems of supporting mobility and providing performance guarantees have been studied extensively in isolation. The novel aspect of our work is to provide an integrated solution to both problems. Consistent with recent surveys on mission-critical wireless networks [14; 15; 16], prior work can be categorized into three broad classes: (i) approaches that support real-time communications in static multi-hop networks, (ii) approaches that support best-effort communications in mobile multi-hop networks, and (iii) approaches that support real-time communications for mobile nodes in single-hop networks.

7.1. Real-time Communications in Static Multi-Hop Networks

Several scheduling algorithms and techniques have been proposed for supporting real-time communications in static wireless networks. A common problem formulation is to consider the scheduling of real-time flows established between arbitrary sources and destinations. Saifullah et al. [33] proved the NP-hardness of this problem. The authors proposed an optimal scheduling algorithm based on the branch-and-bound technique and a necessary condition that helps to reduce the search space. Similarly, Pottner et

al. [5] use an exhaustive search over all possible tree topologies to identify the schedule that meets timeliness, energy, and reliability constraints. However, even when advanced techniques to prune the search space (as in [33]) or parallel computation (as in [5]) are used, the time necessary to find a feasible schedule remains large. As a consequence, it is common to forgo optimality for computationally efficient heuristics such as fixed priority scheduling [37], earliest deadline first [34; 33], or laxity-based scheduling [33]. Consequently, some alternative problem formulations exist. For example, Chipara et al. [37] studied the problem of real-time scheduling for real-time queries used to collect data at a sink. Demirel et al. [27] considered the problem of supporting multiple wireless control loops established between sensors, controllers, and actuators. In contrast to the above solutions that schedule packet transmissions, Zimmerling et al. [28] propose using flooding as a primitive for real-time communications.

Although the above-mentioned approaches support both timeliness and reliability in static networks, they cannot be deemed as suitable solutions for networks with mobile nodes. The same shortcoming holds for the works presented in [45; 46; 36; 42; 47]. In this paper, we referred to this work category as *Static Real-time Scheduling* (SRS) algorithms. Section 3 of this paper presented an in-depth analysis of these works when employed for supporting mobility in real-time networks, and our performance evaluation results confirmed the superior performance of REWIMO compared to these approaches.

Particularly focused on reliability, Han et al. [12] proposed mechanisms to establish multiple paths for both uplink and downlink data forwarding. Similarly, Yan et al. [13] employed multi-path packet forwarding as well as per-link retransmissions to overcome reliability concerns in mission-critical applications. Unfortunately, these approaches only improve the reliability of data forwarding in a static multi-hop network, and they cannot be used to ensure reliable association of mobile nodes with infrastructure nodes, which is an important reliability concern as presented in this paper.

7.2. Supporting Mobility without Timeliness and Reliability Guarantees

MS-MAC [17] is an enhancement of S-MAC [48] that provides mobility support by dynamically adjusting the beaconing interval of S-MAC in response to mobility. MS-MAC detects mobile nodes by monitoring the received signal strength values and establishes Active Zones around the mobile nodes. Nodes in an active zone frequently broadcast synchronization messages, which are required for establishing a connection with other nodes. Although MS-MAC reduces communications delay with mobile nodes, it does not provide any guarantee of timeliness and reliability.

M-LMAC [18] uses a hierarchical network architecture that is similar to REWIMO. The network is composed of infrastructure and mobile nodes, and TDMA techniques (similar to those of [49]) are used to schedule infrastructure-to-infrastructure communications. However, in contrast to REWIMO, M-LMAC handles the traffic variations introduced by mobility through a contention-based approach. Specifically, each time slot includes a contention period during which mobile nodes can contend to communicate with infrastructure nodes. A limitation of M-LMAC is that its operation requires careful tuning of the contention window based on both the topology and mobility pattern of mobile users. As a consequence, M-LMAC does not provide any performance guarantee. This can be observed from the simulation results in [18].

M-TDMA [19] partitions the network into non-overlapping clusters that form a multi-hop infrastructure. Each cluster-head schedules three types of slots: slots assigned to mobile nodes, slots shared between cluster-heads to support mobility, and free slots for future allocation. Movement of nodes between clusters is supported through slots that are shared among clusters and slots that are free for future allocation. If a mobile node joins a cluster that has less than one free slot available,

M-TDMA reduces the current bandwidth assigned to the mobile nodes in that cluster; therefore, the resources assigned to a mobile node may be less than its demand. In contrast, REWIMO employs an admission mechanism and admits a mobile node only if its requested traffic demand can always be satisfied. Additionally, M-TDMA employs an on-demand bandwidth reservation approach that may result in packet loss and delay as mobile nodes associate with different cluster-heads as they move. REWIMO addresses this issue by using an on-join reservation strategy.

MCMAC [21] has been designed to support mobility for applications in which nodes move in groups, such as body area networks. Nodes are categorized into static nodes and mobile clusters. Static nodes communicate using time slots assigned uniquely within two-hop neighborhoods, and mobile nodes communicate using CSMA. MCMAC does not employ multiple frequency scheduling to improve throughput and reliability.

MMAC [22] structures medium access by frames that include both CSMA and TDMA access mechanisms. The protocol uses a localization-based approach to predict the location of mobile nodes and adapt the size of the transmission frame accordingly. In contrast with MMAC, whose performance depends on the accuracy of motion prediction, we have designed REWIMO not to rely on motion models since we focus on the strict guarantee of real-time and reliable communications. The problem of incorporating probabilistic motion prediction models in real-time protocols is an open question that we plan to explore in the future.

MobiSense [20] divides the network into clusters that use non-overlapping frequencies. A distributed scheduling mechanism is employed to organize time frames into admission slots, uplink and downlink slots, and beaconing slots. All the cluster-heads (infrastructure nodes) broadcast beacon messages on a common channel to advertise their used channel and the timing for channel access. When a mobile node intends to join the network or perform handover, it listens to beacon messages, chooses a cluster-head, and then picks a random admission slot to communicate with the cluster-head. The uplink and downlink slots are used for communication with the sink node. Although MobiSense employs a TDMA access mechanism, mobile-to-infrastructure communications are not collision-free and the likelihood of collision depends on factors such as mobility pattern and the traffic rate of mobile nodes. In addition, MobiSense's bandwidth reservation strategy does not achieve end-to-end real-time data delivery.

Unfortunately, non of the above-mentioned works can guarantee real-time and reliable end-to-end packet delivery because they rely on CSMA [17], TDMA [18; 19; 20], or a combination of both [21; 22; 18]. While the unpredictability of CSMA has been widely investigated, the unpredictability of distributed TDMA has been proved as well [23; 24]. In fact, these approaches do not perform end-to-end bandwidth reservation based on the demand of mobile nodes. Therefore, both timeliness and reliability are affected by the number, traffic intensity and mobility pattern of mobile nodes. Furthermore, comparing the capacity and reliability of REWIMO against these approaches would not be fair as their channel access mechanisms were not designed to satisfy the timeliness and reliability required for mission-critical applications.

7.3. Mobility Support with Real-time Communications over a Single-Hop

MBStar [50] has been designed for single-hop body area networks where a user carries multiple devices. The core of the protocol is a scheduling approach that adjusts the phases of real-time flows (referred to as "offset-free scheduling") to reduce the number of collisions. The advantage of this lightweight approach is that it requires little control traffic to adjust the schedules. Similar to REWIMO, MBStar aims to reduce the overhead of schedule dissemination. However, the offset-free scheduling approach cannot be employed in multi-hop networks. In addition, if we assume that a set of infrastruc-

ture nodes (connected through wires[10]) communicate with mobile nodes, an algorithm is required to orchestrate the operation of infrastructure nodes to achieve collision-free scheduling. Such an approach is missing in [50] as nodes in a body area network usually communicate with a single infrastructure node.

Sadi et al. [4; 51] propose a heuristic scheduling approach for single-hop communications between sensors and controllers. This work assumes that wireless sensor nodes generate two types of traffics: (i) periodic time-triggered traffic, and (ii) aperiodic event-triggered traffic. In addition to scheduling, and in order to facilitate the accommodation of event-triggered data, the authors proposed a novel power control and rate adaptation approach to minimize the time required by sensor nodes to concurrently transmit and deliver their data without violating the timeliness, reliability and energy efficiency requirements.

RT-WiFi [52] is a real-time enhancement of WiFi over a single-hop, mainly addressing implementation issues rather than focusing on the algorithmic aspects of real-time scheduling. Similar to the aforementioned approaches, [53] and [54] address real-time scheduling over a single hop. However, none of the works discussed in this category can be employed in a mobile network using a wireless infrastructure (such as REWIMO) because: (i) they do not address end-to-end bandwidth reservation for mobile nodes over a multi-hop infrastructure, and (ii) they do not propose any solution for handover of mobile nodes between infrastructure nodes.

8. CONCLUSION

This work presented the design and performance evaluation of REWIMO – a mobile wireless network solution to provide real-time and reliable data exchange with mobile nodes. REWIMO has a hierarchical network architecture that is composed of fixed infrastructure nodes and mobile nodes. The delivery of packets to the gateway is divided into two parts: from the mobile node to an infrastructure node, and from the infrastructure node to the gateway. REWIMO has three salient features: (i) improving real-time capacity through incorporation of transmission scheduling techniques to cope with path uncertainty introduced by mobility, (ii) reliability guarantee through real-time scheduling as well as beacon and data transmission coordination, (iii) incorporation of MARS and A-MARS to provide different performance trade-offs. MARS is designed to maximize real-time capacity. In contrast, A-MARS is designed to quickly adapt to network dynamics including the addition and removal of real-time flows.

We have developed a sophisticated simulator to evaluate the performance of MARS and A-MARS in a realistic and repeatable manner. Our results show that MARS and A-MARS significantly increase (14x) the number of mobile nodes admitted to the network, compared to the algorithms designed for static real-time networks. A-MARS, in particular, results in a fixed admission delay for mobile nodes, with less than 15% reduction in the number of mobile nodes admitted. In terms of reliability, the two-phase scheduling technique employed by MARS and A-MARS results in up to 30% higher reliability when the period of data flow is nine times longer than the beaconing period.

ACKNOWLEDGMENTS

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

[10]Note that the infrastructure nodes in REWIMO communicate wirelessly.

REFERENCES

[1] K. Lorincz and M. Welsh, "MoteTrack: A robust, decentralized approach to RF-based location tracking," *Personal and Ubiquitous Computing*, vol. 11, no. 6, pp. 489–503, 2007.

[2] MoteTrack, "A Robust, Decentralized Approach to RF-Based Location Tracking." http://www.eecs.harvard.edu/~konrad/projects/motetrack/, 2006.

[3] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Springer US, second ed., 2011.

[4] Y. Sadi and S. Coleri Ergen, "Energy and delay constrained maximum adaptive schedule for wireless networked control systems," *IEEE Transactions on Wireless Communications*, vol. 14, no. 7, pp. 3738–3751, 2015.

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

[6] N. Baccour, A. Koubâa, L. Mottola, M. A. Zúñiga, H. Youssef, C. A. Boano, and M. Alves, "Radio link quality estimation in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 8, no. 4, pp. 1–33, 2012.

[7] M. Radi, B. Dezfouli, K. A. Bakar, S. A. Razak, and M. Lee, "Network Initialization in Low-Power Wireless Networks: A Comprehensive Study," *The Computer Journal, Oxford*, vol. 57, pp. 1238–1261, aug 2014.

[8] K. Srinivasan and P. Levis, "RSSI is Under Appreciated," in *Third Workshop on Embedded Networked Sensors (EmNets)*, 2006.

[9] N. Baccour, A. Koubâa, H. Youssef, and M. Alves, "Reliable link quality estimation in low-power wireless networks and its impact on tree-routing," *Ad Hoc Networks*, vol. 27, pp. 1–25, apr 2015.

[10] M. Z. Zamalloa and B. Krishnamachari, "An analysis of unreliability and asymmetry in low-power wireless links," *ACM Transactions on Sensor Networks*, vol. 3, pp. 63–81, jun 2007.

[11] O. Chipara, G. Hackmann, C. Lu, W. D. Smart, and G.-C. Roman, "Practical modeling and prediction of radio coverage of indoor sensor networks," in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, p. 339, 2010.

[12] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon, "Reliable and Real-Time Communication in Industrial Wireless Mesh Networks," in *17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, (Chicago, IL), pp. 3–12, apr 2011.

[13] 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.

[14] P. Bartolomeu, M. Alam, J. Ferreira, and J. Fonseca, "Survey on low power real-time wireless MAC protocols," *Journal of Network and Computer Applications*, vol. 75, pp. 293–316, nov 2016.

[15] P. Suriyachai, U. Roedig, and A. Scott, "A survey of MAC protocols for mission-critical applications in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 14, pp. 240–264, jan 2012.

[16] Q. Dong and W. Dargie, "A survey on mobility and mobility-aware MAC protocols in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 88–100, 2012.

[17] H. Pham and S. Jha, "An adaptive mobility-aware MAC protocol for sensor networks (MS-MAC)," in *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pp. 558–560, 2004.

[18] L. van Hoesel, A. Tuysuz-Erman, and P. Havinga, "Ideas on node mobility support in schedule-based medium access," in *International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pp. 539–544, dec 2008.

[19] A. Jhumka and S. Kulkarni, "On the design of mobility-tolerant TDMA-based media access control (MAC) protocol for mobile sensor networks," in *4th International Conference on Distributed Computing and Internet Technology (ICDCIT)*, pp. 42–53, 2007.

[20] A. Gonga, O. Landsiedel, and M. Johansson, "MobiSense: Power-efficient micro-mobility in wireless sensor networks," in *International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pp. 1–8, jun 2011.

[21] M. Nabi, M. Geilen, T. Basten, and M. Blagojevic, "Efficient Cluster Mobility Support for TDMA-Based MAC Protocols in Wireless Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 10, pp. 1–32, jun 2014.

[22] M. Ali, T. Suleman, and Z. Uzmi, "MMAC: a mobility-adaptive, collision-free MAC protocol for wireless sensor networks," in *24th IEEE International Performance, Computing, and Communications Conference (PCCC)*, pp. 401–407, 2005.

[23] B. Dezfouli, M. Radi, K. Whitehouse, S. A. Razak, and H.-P. Tan, "DICSA: Distributed and concurrent link scheduling algorithm for data gathering in wireless sensor networks," *Ad Hoc Networks*, vol. 25, pp. 54–71, feb 2015.

[24] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: Distributed Randomized TDMA Scheduling for Wireless Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 8, pp. 1384–1396, oct 2009.

[25] WirelessHART, "HART Communication Protocol and Foundation." www.en.hartcomm.org, 2016.

[26] ISA1000, "Wireless Systems for Automation- ISA." www.isa.org, 2016.

[27] B. Demirel, Z. Zou, P. Soldati, and M. Johansson, "Modular design of jointly optimal controllers and forwarding policies for wireless control," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3252–3265, 2015.

[28] M. Zimmerling, P. Kumar, F. Ferrari, and L. Thiele "Adaptive real-time communication for wireless cyber-physical systems," *ETH Zurich, Tech. Rep*, 2016.

[29] 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 *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 377–386, apr 2008.

[30] S. Petersen and S. Carlsen, "WirelessHART Versus ISA100.11a: The Format War Hits the Factory Floor," *IEEE Industrial Electronics Magazine*, vol. 5, pp. 23–34, dec 2011.

[31] T. O'donovan, W.-B. Pöttner, U. Roedig, J. S. Silva, R. Silva, C. J. Sreenan, V. Vassiliou, T. Voigt, L. Wolf, Z. Zinonos, J. Brown, F. Büsching, A. Cardoso, J. Cecílio, J. D. Ó, P. Furtado, P. Gil, and A. Jugel, "The GINSENG system for wireless monitoring and control," *ACM Transactions on Sensor Networks*, vol. 10, no. 1, pp. 1–40, 2013.

[32] Haibo Zhang, P. Soldati, and M. Johansson, "Optimal link scheduling and channel assignment for convergecast in linear WirelessHART networks," in *7th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*, pp. 1–8, IEEE, jun 2009.

[33] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-Time Scheduling for WirelessHART Networks," in *31st IEEE Real-Time Systems Symposium (RTSS)*, pp. 150–159, nov 2010.

[34] C. Wu, M. Sha, D. Gunatilaka, A. Saifullah, C. Lu, and Y. Chen "Analysis of EDF scheduling for wireless sensor-actuator networks." in *IEEE 22nd International Symposium of Quality of Service (IWQoS)*, pp. 31-40, 2014.

[35] P. Suriyachai, J. Brown, and U. Roedig, "Time-Critical Data Delivery in Wireless Sensor Networks," in *6th IEEE international conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 216–229, 2010.

[36] 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.

[37] O. Chipara, C. Lu, and G.-C. Roman, "Real-Time Query Scheduling for Wireless Sensor Networks," in *28th IEEE International Real-Time Systems Symposium (RTSS)*, pp. 389–399, dec 2007.

[38] O. Chipara, C. Lu, T. C. Bailey, and G.-C. Roman, "Reliable clinical monitoring using wireless sensor networks," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, p. 155, 2010.

[39] S. Banerjee and D. O. Wu, "Final report from the NSF Workshop on Future Directions in Wireless Networking," tech. rep., National Science Foundation, 2013.

[40] EuropeanCommission, "Factories of the Future (FoF)." http://ec.europa.eu/research/ industrial_technologies/ factories-of-the-future_en.html, 2015.

[41] K. S. J. Pister and L. Doherty, "TSMP: Time synchronized mesh protocol," in *Parallel and Distributed Computing Systems (PDCS)*, pp. 391–398, 2008.

[42] H. Zhang, P. Soldati, and M. Johansson, "Performance bounds and latency-optimal scheduling for convergecast in WirelessHART networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 6, pp. 2688–2696, 2013.

[43] W. Torfs and C. Blondia, "TDMA on commercial of-the-shelf hardware: Fact and fiction revealed," *International Journal of Electronics and Communications*, vol. 69, no. 5, pp. 800–813, 2015.

[44] P. Suriyachai, U. Roedig, and A. Scott, "Implementation of a MAC protocol for QoS support in wireless sensor networks," in *IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, pp. 1–6, mar 2009.

[45] E. Toscano and L. Lo Bello, "Multichannel superframe scheduling for IEEE 802.15.4 industrial wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 337–350, 2012.

[46] K. Dang, J. Z. Shen, L. D. Dong, and Y. X. Xia, "A graph route-based superframe scheduling scheme in WirelessHART mesh networks for high robustness," *Wireless Personal Communications*, vol. 71, no. 38, pp. 2431–2444, 2013.

[47] X. Kang, W. Wang, J. J. Jaramillo, and L. Ying, "On the Performance of Largest-Deficit-First for Scheduling Real-Time Traffic in Wireless Networks," *IEEE/ACM Transactions on Networking*, vol. 24, pp. 72–84, feb 2016.

[48] J. Heidemann, D. Estrin, and W. Ye, "An energy-efficient MAC protocol for wireless sensor networks," in *IEEE International Conference on Computer Communications (INFOCOMM)*, vol. 3, pp. 1567–1576, 2002.

[49] L. van Hoesel and P. Havinga, "Collision-free Time Slot Reuse in Multi-hop Wireless Sensor Networks," in *International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pp. 101–107, 2005.

[50] X. Zhu, S. Han, P.-C. Huang, A. K. Mok, and D. Chen, "MBStar: A Real-time Communication Protocol for Wireless Body Area Networks," *23rd Euromicro Conference on Real-Time Systems*, pp. 57–66, 2011.

[51] Y. Sadi and S. Coleri Ergen, "Optimal power control, rate adaptation, and scheduling for UWB-based intravehicular wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 1, pp. 219–234, 2013.

[52] 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 *IEEE 34th Real-Time Systems Symposium (RTSS)*, pp. 140–149, dec 2013.

[53] I.-H. Hou, "Scheduling Heterogeneous Real-Time Traffic Over Fading Wireless Channels," *IEEE/ACM Transactions on Networking*, vol. 22, pp. 1631–1644, oct 2014.

[54] I.-H. Hou and P. R. Kumar, "Utility-optimal scheduling in time-varying wireless networks with delay constraints," in *ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, p. 31–40, 2010.

[55] CC2420, "2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver." http://www.ti.com/product/cc2420, 2016.

[56] CC2500, "Single Chip Low Cost Low Power RF Transceiver." www.ti.com/product/cc2500, 2016.

[57] CC2650, "SimpleLink multi-standard 2.4 GHz ultra-low power wireless MCU." http://www.ti.com/product/CC2650, 2016.

[58] M. Nobre, I. Silva, and L. Guedes, "Routing and Scheduling Algorithms for WirelessHART Networks: A Survey," *Sensors*, vol. 15, no. 5, pp. 9703–9740, 2015.

[59] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "End-to-end communication delay analysis in industrial wireless networks," *IEEE Transactions on Computers*, vol. 64, no. 5, pp. 1361–1374, 2015.

[60] D. Chen, M. Nixon, and A. Mok, *WirelessHART: Real-Time Mesh Network for Industrial Automation*. Springer US, 1st ed., 2010.

[61] OMNeT++, "The OMNeT++ Network Simulation Framework." http://www.omnetpp.org, 2016.

[62] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis, "CTP: An efficient, robust, and reliable collection tree protocol for wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 10, pp. 1–49, nov 2013.

[63] B. Dezfouli, M. Radi, S. A. Razak, K. Whitehouse, K. A. Bakar, and H.-P. Tan, "Improving broadcast reliability for neighbor discovery, link estimation and collection tree construction in wireless sensor networks," *Computer Networks, Elsevier*, vol. 62, pp. 101–121, apr 2014.

[64] B. Dezfouli, M. Radi, S. A. Razak, H.-P. Tan, and K. A. Bakar, "Modeling low-power wireless communications," *Journal of Network and Computer Applications*, vol. 51, pp. 102–126, may 2015.

Received x 2016; revised x 2016; accepted x 2016