This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

Reliable Dynamic Packet Scheduling with Slot Sharing for Real-Time Wireless Networks

Tianyu Zhang, Tao Gong, Mingsong Lyu, Nan Guan, *Member, IEEE,* Song Han, *Member, IEEE,* Xiaobo Sharon Hu, *Fellow, IEEE*

Abstract—In order for real-time wireless networks (RTWNs) to achieve desired Quality of Service (QoS) for real-time sensing and control, effective packet scheduling algorithms play a critical role, especially in the presence of unexpected disturbances. Most existing solutions in the literature focus either on static or dynamic schedule construction to meet the desired QoS requirements, but have a common assumption that all wireless links are reliable. However, this assumption is not realistic in real-life settings. To address this drawback, this paper introduces a novel reliable dynamic packet scheduling framework, called RD-PaS. RD-PaS can not only construct static schedules to meet both the timing and reliability requirements of end-to-end flows in RTWNs, but also construct new schedules rapidly to handle abruptly increased network traffic induced by unexpected disturbances while minimizing the impact on existing network flows. Through judiciously sharing time slots among tasks, RD-PaS can significantly reduce the number of required time slots to meet the system reliability requirement and improve the network throughput. The functional correctness of the RD-PaS framework has been validated through its implementation and deployment on a real-life RTWN testbed. Extensive simulation-based experiments have also been performed to evaluate the effectiveness of RD-PaS, especially in large-scale network settings.

Index Terms—Real-time wireless networks (RTWNs), lossy links, dynamic packet scheduling, reliability, shared slots

1 Introduction

In recent years, real-time wireless networks (RTWNs) have been making their way into a wide range of industrial applications [1]. These applications are commonly featured with stringent timing and reliability requirements to ensure timely data collection and control decision making. Thus packet scheduling in RTWNs plays an important role to achieve the desired Quality of Service (QoS) in such applications. QoS here is often measured by how well the network delivers the packets by their deadlines. Although packet scheduling in RTWNs has been studied for a long time, how to handle abruptly increased network traffic in the presence of unexpected disturbances (*i.e.*, events causing more frequent sensing of the environment and processing of sensed data) remains a challenge. This challenge is further exacerbated by the lossy wireless links in typical industrial environments [2].

Most RTWNs adopt Time Division Multiple Access (TDMA) based data link layers to achieve deterministic real-time communication. Sensing and control tasks are abstracted as end-to-end (e2e) flows with specified timing and reliability requirements. A key challenge of meeting the e2e timing requirements comes from the existence of unexpected disturbances occurred at run time. Disturbances can trigger changes in network resource demands from certain tasks during the network operation for realizing some functionalities (*e.g.*, more frequent environment monitoring). The unpredictable occurrence of disturbances makes offline static scheduling approaches not effective in handling disturbances while satisfying the flows' timing requirements. The lossy nature of wireless links in the industrial environment raises another challenge in meeting the e2e reliability requirements in RTWNs. The lossy wireless links in the network cause unexpected packet losses, which can degrade the data freshness, lead to system instability and even cause catastrophe to the system. Thus, most RTWNs require a desired e2e Packet Delivery Ratio (PDR), *e.g.* 99% for all critical flows in the system.

To address the above challenges, this paper introduces a reliable dynamic packet scheduling framework, called RD-PaS, to meet both the *timing* and *reliability* requirements in packet scheduling in RTWNs in the presence of disturbances[1]. When no disturbance occurs (*i.e.*, in the static scenario), RD-PaS determines the *minimum* number of retransmission slots needed for each task to guarantee reliable e2e packet delivery, and constructs a communication schedule locally in a hybrid manner at individual nodes. The hybrid approach requires collaboration between a centralized controller and a local schedule generator running on individual nodes to keep a good tradeoff between bandwidth usage and QoS. When a disturbance occurs, RD-PaS generates a dynamic schedule to guarantee desired reliability of critical task(s) while judiciously degrade the reliability of packet transmissions for other tasks.

We present a formal formulation of the reliable dynamic packet scheduling problem to minimize such degradation, prove that this problem is NP-hard, and present an effective heuristic to solve it. To improve the network bandwidth usage, we further propose a novel shared slot transmission mechanism by allowing time slot sharing among multiple

• *T. Zhang, M. Lyu and N. Guan are with the Hong Kong Polytechnic University, Hong Kong. Email: {tianzhang, nan.guan}@polyu.edu.hk.*
• *T. Gong and S. Han are with the University of Connecticut, US. Email: {tao.gong, song.han}@uconn.edu.*
• *X. S. Hu is with the University of Notre Dame, US. Email: shu@nd.edu.*

1. An earlier version of the paper appears in [3].

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

flows, which can significantly reduce the required number of slots to meet the reliability requirement. The functional correctness of the RD-PaS framework has been validated through its implementation and deployment on a real-life RTWN testbed. Extensive simulation-based experiments have also been performed to evaluate the effectiveness of RD-PaS, especially in large-scale network settings. Our results show that RD-PaS can reduce the degradation in the e2e PDR by 58% on average compared to the state-of-the-art method.

The remainder of this paper is organized as follows. In Section 2, we review the related work. Section 3 describes the system model and problem definition, and gives an overview of the RD-PaS framework. Section 4 presents the details of RD-PaS for the Transmission-based Scheduling (TBS) model, including both static schedule construction and dynamic schedule adjustment in the presence of disturbances. These efforts are further extended to the Packet-based Scheduling (PBS) model in Section 5. In Section 6, we propose a novel shared slot transmission mechanism in order to reduce the demand on the network resource and improve the network bandwidth usage. In Section 7, we present the implementation and functional validation of RD-PaS on a real-life RTWN testbed. Performance evaluation from extensive simulation-based experiments is reported in Section 8. Finally, we conclude the paper and discuss future work in Section 9.

2 Related Work

Most packet scheduling algorithms designed for RTWNs in the literature focus on schedulability analysis and employ centralized and static (or infrequently updated) management frameworks (*e.g.*, [4], [5]). Those solutions may fit well for small-scale static RTWNs. They however often lead to significantly degraded QoS when the system becomes large and when deployed in harsh environments for monitoring and controlling complex physical processes where unexpected disturbances occur frequently.

To model and respond to disturbances in RTWNs, many dynamic scheduling approaches have been proposed. [6] supports admission control in response to adding/removing tasks for handling disturbances in the network. However, it does not consider scenarios when not all tasks can meet their deadlines. The protocol in [7] proposes to allocate reserved slots for occasionally occurring emergencies (*i.e.*, disturbances), and allow regular tasks to steal slots from the emergency schedule when no emergency presents. However, how to satisfy the deadlines of regular tasks in the presence of emergencies is not considered. [8] proposes a MAC protocol with a centralized reschedule scheme allowing on-line changes of active streams and network topology. However, the scheduler and the data format of the schedule distribution are not specified. A number of efficient distributed scheduling frameworks (*e.g., [9]–[13]*) are proposed in Time Slotted Channel Hopping (TSCH) networks to quickly react to network dynamics. However, all those works do not consider real-time constraints, i.e., ignore the hard deadlines associated with tasks running in the network. They only provide best effort but no guarantee on the end-to-end latency of each task.

Another thread of research significantly advances the state of the art by providing dynamic packet scheduling functions in RTWNs. Among these approaches, OLS in [14] relies on a centralized gateway to construct and disseminate a dynamic schedule to all the nodes in the network; D^2-PaS in [15], [16] offloads the schedule construction to individual nodes and only disseminates minimum information for the nodes to construct a dynamic schedule locally; and FD-PaS in [17], [18] further eliminates the need of a centralized gateway by notifying and handling the disturbances in a local and distributed manner. They, however, all assume perfect wireless links, which is not realistic especially in noisy and harsh industrial environments. To the best of our knowledge, none of the existing dynamic packet scheduling algorithms consider packet losses and thus many lead to significantly degraded QoS in real-life deployment.

On the other hand, a rich set of methods have been designed for RTWNs to improve the reliability of wireless packet transmission over lossy links. For instance, most RTWN solutions (*e.g.*, WirelessHART [19], ISA 100.11a [20], and 6TiSCH [21]) employ multiple channels and some frequency hopping mechanisms to minimize potential interference. Furthermore, [22] proposes a set of reliable graph routing algorithms in WirelessHART networks to explore path diversity to improve reliability. Those works are complementary to the approach to be introduced in this paper since this work focuses on single channel with pre-defined routing paths. [23] proposes an algorithm to allocate a necessary number of retransmision time slots for individual links to guarantee a desired success ratio of packet delivery in a star network topology. [24] extends the network model in [23] to allow multi-hop flows and proposes both link-centric and flow-centric retransmission policies. However, the policies proposed in [23], [24] tend to assign more retransmission slots than necessary, and thus require higher network bandwidth. By contrast, our approach proposed in this work results in an optimal retransmission slot assignment. Moreover, these retransmission policies assign dedicated time slots to either transmissions or flows and are inefficient to address unpredictable transmission failures. This is because the reserved retransmission timeslots can be wasted when the previous transmission is successful.

To support flexibility in timeslot allocation and improve retransmission efficiency of RTWNs, several shared-timeslot retransmission schemes are proposed. The IEEE 802.15.4 TSCH specification [25] supports both dedicated (contention-free) slots and shared (contention-based) slots with CSMA back-off. [26] introduces a hybrid timeslot design where a time slot can be used as a dedicated slot by a primary link and a shared slot by several backup links. [27] proposes an autonomous scheduling method relying on a mixture of dedicated slots and shared slots. To reduce transmission collision in shared slots, [28] presents a segmented slot assignment method in which multiple nodes compete for a shared slot by randomly choosing a CCA offset. By assigning each shared slot only to the links with a same receiver, [29] further reduces the contention on accessing these timeslots. However, all the approaches above suffer from the drawback that deterministic performance cannot be provided for the traffic in CSMA-based shared slots. On the other hand, both [30] and [31] are recent work proposing

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

contention-free shared slot transmission mechanisms. [30] allocates shared slots to tasks with common links along their routing paths and [31] assigns shared slots to transmissions with common receivers. Both approaches rely on a joint node to decide the transmission within each shared slot and such link-based slot sharing schemes all suffer from the state explosion problem when calculating the PDR values for multi-hop end-to-end packets.

3 Preliminaries

In this section, we first discuss the system model and then give an overview of the proposed RD-PaS framework.

3.1 System Model and Problem Definition

The system architecture of an RTWN studied in this work is modeled after RTWNs often found in industrial process control applications. Such an RTWN consists of multiple sensor and actuator nodes wirelessly connected to a single controller node either directly or through relay nodes. The network is described by a directed graph $G = (V, E)$, where the node set $V = \{V_0, V_1, \ldots, V_c\}$. V_c is the controller node and the rest are referred to as the device nodes. A direct link $e = (V_i, V_j) \in E$ represents a wireless link from node V_i to V_j with a Packet Delivery Ratio (PDR), λ_e^L, which represents the probabilistic transmission success rate on link e[2]. V_c connects to all the nodes via designated routes and is responsible for executing relevant control algorithms. V_c also contains a network manager which conducts network configuration and resource allocation. In this work, we focus on RTWNs with only one controller node. Networks with multiple controller nodes are left for future work.

We assume that the system executes a fixed set of control tasks $\mathcal{T} = \{\tau_0, \tau_1, \ldots, \tau_N\}$ where τ_i $(0 \leq i < N)$ is a unicast task and τ_N is a broadcast task. Each task τ_i is associated with a period P_i and deadline D_i, and follows a designated single routing path with H_i hops. We use $\overrightarrow{L_i} = [L_i[0], L_i[1], \ldots, L_i[H_i - 1]]$ to represent the routing path of task τ_i. For a unicast task, $L_i[h] \in E$ $(0 \leq h < H_i)$. Each unicast task periodically generates a packet originated at a sensor node, passing through the controller node and delivering a control message to the designated actuator node. For the broadcast task τ_N, each hop involves multiple links, thus $L_N[h] = (L_N[h](0), L_N[h](1), \ldots)$, where $L_N[h](i) \in E$. The broadcast task runs periodically in V_c and only generates packets when necessary. These packets are broadcast to each node directly or though some intermediate nodes on the broadcast path L_N. The j-th released instance of τ_i is referred to as packet $\chi_{i,j}$, with its release time, deadline, and finish time denoted as $r_{i,j}$, $d_{i,j}$ and $f_{i,j}$, respectively. We denote the transmission of packet $\chi_{i,j}$ at the h-th hop as transmission $\chi_{i,j}(h)$, $(0 \leq h < H_i)$.

Fig. 1 shows an example RTWN running 4 unicast tasks (τ_0, τ_1, τ_2 and τ_3) and 1 broadcast task (τ_5) on 7 nodes ($V_0, V_1, \ldots, V_5$ and V_c) where V_0, V_3, V_5 are the sensor nodes, V_1, V_4 are the actuator nodes, and V_2 is a combined sensor and actuator node. The routing paths of individual tasks are summarized on the right side of Fig. 1.

In applications such as crude oil refining, a disturbance, *e.g.*, a sudden change in temperature, may occur unexpectedly. When a disturbance occurs, the system usually requires the sensor nodes located within the range of the disturbance to monitor the environment more closely, and thus one or multiple tasks may demand more network bandwidth during the disturbance. To capture such abrupt increase in network resource demand upon the detection of a disturbance, we adopt the rhythmic task model [32] in this work[3]. In the rhythmic model, each task has two states: *nominal state* and *rhythmic state*. In the nominal state, τ_i releases packets following the nominal period P_i and each packet has a relative deadline $D_i \leq P_i$. In the rhythmic state, the period and relative deadline of τ_i adopt a series of new values specified by pre-designed vectors $\overrightarrow{P_i}$ and $\overrightarrow{D_i}$. Once τ_i returns to the nominal state, it starts to use P_i and D_i again. When a disturbance occurs and the corresponding tasks (denoted as $\mathcal{T}_{Rhy}$) enter their rhythmic states, we say the system switches to the *rhythmic mode*. The system returns to the *nominal mode* after the disturbance has been completely handled, *i.e.* all the corresponding tasks return to their nominal states. In Fig. 1, when the disturbance (in the yellow region) occurs, τ_0 and τ_2 (installed on nodes V_3 and V_0, respectively) will enter their rhythmic states and the system switches to the rhythmic mode. In the following, we first assume that at any time during the system operation, at most one disturbance can occur and needs to be detected and handled. We will then generalize the system model to discuss concurrent disturbances at the end of Section 4.2.

Following the industrial practice for RTWNs, we consider a synchronized network adopting a time-slotted schedule. The length of a time slot is typically 10ms. Within each time slot, at most one packet can be transmitted over the air from a sender to a receiver. The acknowledgement (ACK) is then sent back from the receiver to the sender in the same slot to notify the successful reception.

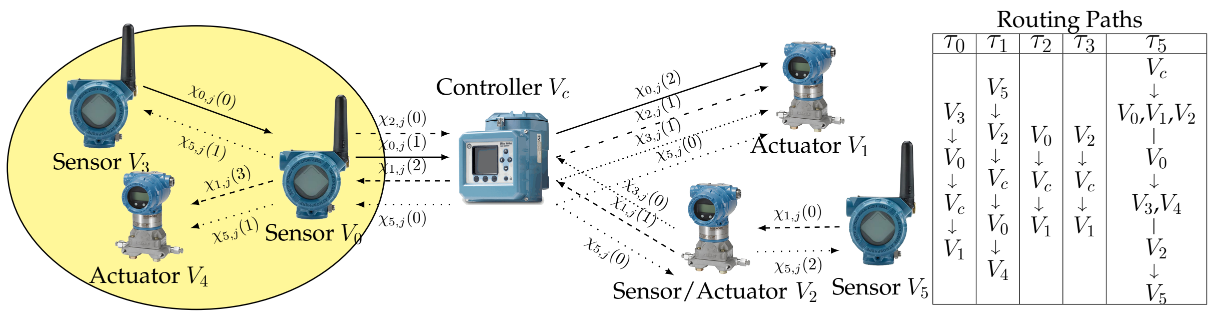
Traditional RTWNs employ Link-based Scheduling (LBS) to allocate time slots. In LBS, each time slot is allocated to a link by specifying the sender and receiver. If packets from different tasks share a common link and are all buffered at the same sender, their transmission order is decided by a node-specified policy (*e.g.*, FIFO). This approach introduces uncertainty in packet scheduling and may violate the e2e timing constraints on packet delivery. To tackle this problem, *Transmission-based Scheduling (TBS)* and *Packet-based Scheduling (PBS)* are proposed in [15] and [24], respectively, to construct deterministic schedules. Each of the two scheduling models has its own advantages and disadvantages and is preferred in different usage scenarios as discussed in [24]. Hence, we consider both models in our RD-PaS framework. Furthermore we focus on single-channel RTWNs in this work since it forms the basis for more advanced studies. Multi-channel networks are left for future work.

2. Link PDR λ_e^L is usually measured during the site survey and we apply a considerable lower PDR value for each link to further improve the system reliability in case the link quality slightly decreases. If the value of λ_e^L changes significantly, the new value will be reported to the network manager which will broadcast the information to all the nodes in the network.

3. RD-PaS is not limited to the rhythmic task model and can be applied to any task models capturing unexpected network resource demand changes.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922



τ_0	τ_1	τ_2	τ_3	τ_5
$V_3 \downarrow V_0 \downarrow V_c \downarrow V_1$	$V_5 \downarrow V_2 \downarrow V_c \downarrow V_0 \downarrow V_4$	$V_0 \downarrow V_c \downarrow V_1$	$V_2 \downarrow V_c \downarrow V_1$	$V_c \downarrow V_0,V_1,V_2 \mid V_0 \downarrow V_3,V_4 \mid V_2 \downarrow V_5$

Fig. 1: An example RTWN with 5 tasks running on 7 nodes. The sensor and actuator nodes are taken from a crude oil processing plant.

In the TBS model, each time slot is allocated to the transmission of a specfic packet $\chi_{i,j}$ at a particular hop h or kept idle. Once the network schedule is constructed, packet transmission in each time slot is unique and fixed. In the PBS model, each time slot is allocated to a specific packet $\chi_{i,j}$ or kept idle. Within each time slot assigned to $\chi_{i,j}$, every node along $\chi_{i,j}$'s routing path decides the action to take (*e.g.*, transmit, receive or idle), depending on whether the node has received $\chi_{i,j}$ or not.

Since each link e in the network may suffer packet losses, *i.e.*, $\lambda_e^L < 1$, packet transmissions may fail, which can significantly affect the timely delivery of real-time packets. To handle such cases, a retransmission mechanism is commonly employed in RTWNs [19], [21]. Specifically, if a sender node does not receive the ACK from the receiver node, it automatically retransmits the packet in the next possible time slot.

To quantify the reliability requirement of the e2e packet delivery for each task, a *required* e2e PDR for τ_i, denoted as λ_i^R, is introduced. For example, a control application can tolerate 0.01% packet loss, so λ_i^R is 99.99%. Based on λ_i^R, the transmission of any packet of τ_i is reliable if and only if the achieved e2e PDR of τ_i is larger than or equal to λ_i^R, *i.e.*, $\lambda_{i,j} \geq \lambda_i^R$. To simplify presentation, we assume that all tasks in the network share a common required e2e PDR value, denoted as λ^R. However, our proposed approach can be easily extended to support different λ^R's for different tasks. Table 1 summarizes the frequently used symbols in this paper.

Based on the above system model, we aim to design a general packet scheduling framework to provide specific timing and reliability guarantees for all the tasks running in the network in the presence of dynamic disturbances. To achieve this, we accomplish the framework design by following two steps corresponding to solving two subproblems as follows. **P1:** In the system nominal mode, construct a schedule such that both the e2e timing and reliability requirements of all tasks can be satisfied; **P2:** When disturbances occur and are detected, adjust the schedule in a dynamic and hybrid manner to still guarantee the reliable and timely transmissions of the rhythmic packets while achieving the minimum reliability degradation on other packets. More formally, we have the following problem formulation.

P1: Given RTWN $G = (V, E)$ where each link $e \in E$ has an associated PDR, and task set $\mathcal{T}$ in which each task τ_i has a single routing path $\overrightarrow{L_i}$, determine the nominal-mode schedule under which the following constraints are satisfied.

Constraint 1 $\forall i, j, \lambda_{i,j} \geq \lambda^R$. (e2e reliability requirements for all tasks)

Constraint 2 $\forall i, j, f_{i,j} \leq d_{i,j}$. (e2e timing requirements for all tasks)

P2*: Given the packet set, Γ, in the rhythmic mode under consideration, the PDR function of each task τ_i, and other network related constraints, determine the rhythmic-mode schedule such that $\sum_{\chi_{i,j} \in \Gamma} \max\{0, \lambda^R - \lambda_{i,j}\}$ is minimized, with the following constraints being satisfied.

Constraint 3 $\forall \tau_i \in \mathcal{T}_{Rhy}, \lambda_{i,j} \geq \lambda^R$. (e2e reliability requirements for rhythmic tasks)

Constraint 4 $\forall \tau_i \in \mathcal{T}_{Rhy}, f_{i,j} \leq d_{i,j}$. (e2e timing requirements for rhythmic tasks)

Here we use **P2*** instead of **P2** as we have not discussed the network constraints. They will be elaborated in Section 4 and 5 where formal definitions of **P2** will be given.

In this paper, we assume **P1** is solvable, *i.e.*, there exists a solution that can satisfy both Constraints 1 and 2 simultaneously. Otherwise, extra information should be provided from the application to distinguish tasks in terms of criticality according to their importance in the system model. Then, an optimization problem can be formulated to maximize the level of satisfying tasks' timing and reliability requirements. This thread of work will be explored in the future.

3.2 Overview of the RD-PaS Framework

We propose a reliable dynamic packet scheduling framework, referred to as RD-PaS, to address the questions raised above. An overview of the execution model of RD-PaS is shown in Fig. 2. Below we focus on a high-level discussion while leave the detailed explanation of the symbols in Section 4.

In the network initialization phase, each device node stores necessary specification information of all tasks (*i.e.*, H_i, D_i, P_i and λ^R) locally after receiving it from the network manager through broadcast packets. Each device node then calculates the number of time slots to be allocated to each task (for both transmission and retransmission) in order to achieve the required e2e PDR value λ^R.

After the network starts, each device node generates a static schedule locally, following which all tasks can meet

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

TABLE 1: Table of important symbols and notations

Notation	Definition	Notation	Definition
$V_0, V_1, \ldots$	Device nodes: sensor, actuator or relay node	λ^R	Required e2e packet delivery ratio (for all tasks)
V_c	Controller node	$\lambda^L_{L_i[h]}$	Measured link packet delivery ratio of link $L_i[h]$
$\mathcal{T}$, τ_i	Task set and task i	$\lambda_{i,j}$, $\overrightarrow{R}_{i,j}$	E2e PDR value and retry vector of $\chi_{i,j}$
H_i, P_i, D_i	Number of hops, period and deadline of τ_i	$R_{i,j}[h]$	Number of trials for h-th hop assigned by $\overrightarrow{R}_{i,j}$
$L_i[h]$	The h-th link on the routing path of τ_i $(0 \leq h < H_i)$	$\lambda^*_i(\cdot)$	Optimal PDR of τ_i as a function of number of assigned slots
$\chi_{i,j}$	The j-th released packet of τ_i	$\overrightarrow{R}^*_i(\cdot)$	Optimal retry vector of τ_i as a function of number of assigned slots
$r_{i,j}, d_{i,j}, f_{i,j}$	Absolute release time, deadline, finish time of $\chi_{i,j}$	w^+_i, w^*_i	The smallest w achieving $\lambda^*_i(w) \geq \lambda^R$ under DTM and STM
$W_{i,j}$	Total number of slots assigned for $\chi_{i,j}$	$[t_{sp}, t_{ep})$	Time duration of system rhythmic mode (dynamic schedule)
$\delta_{i,j}$	PDR degradation of $\chi_{i,j}$	Γ, ρ	Active packet set and updated packet set

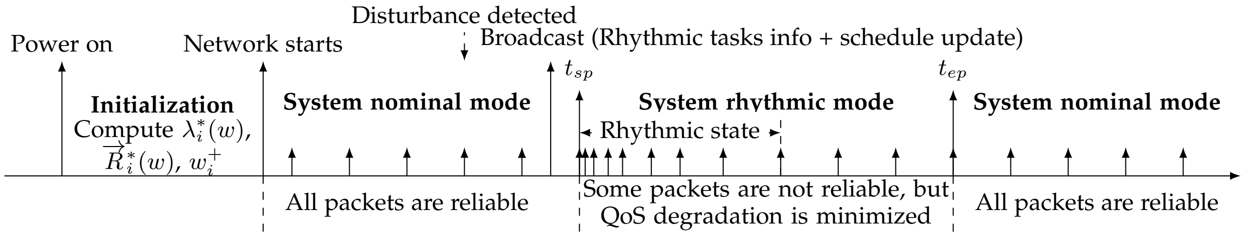


Fig. 2: Overview of the execution model of RD-PaS in both nominal and rhythmic modes. Short upward arrows represent the releases of the rhythmic packets.

their timing and reliability requirements. By locally generating a static schedule, no unnecessary bandwidth is wasted on transmitting the schedule from the gateway. When a disturbance occurs, several sensor nodes within the range may detect it and send a report to the controller node via the corresponding tasks. After the controller node receives the disturbance information from any of the sensor nodes, V_c first determines a time duration, denoted as $[t_{sp}, t_{ep})$, during which the system runs in the rhythmic mode using a temporary dynamic schedule.[4] As RD-PaS and D^2-PaS in [15] both require each node to generate schedule locally, RD-PaS adopts the same end point selection method in D^2-PaS to determine the system rhythmic mode duration $[t_{sp}, t_{ep})$. V_c then, checks whether all tasks can still be reliably delivered after the rhythmic tasks entering their rhythmic states. If so, V_c only broadcasts the rhythmic tasks information (task IDs and the corresponding $\overrightarrow{P_i}$ and $\overrightarrow{D_i}$) to the network. Otherwise, V_c needs to generate a dynamic schedule in which the number of time slots assigned to certain periodic packets are updated in order to accommodate the increased workload from the rhythmic tasks. V_c then piggybacks the information of the updated packet set as well as the rhythmic task information to a broadcast packet and disseminates it to all nodes in the network. After all the nodes receive the updates, the system switches to the rhythmic mode to handle the disturbance.

In the rhythmic mode, individual device nodes generate their own dynamic schedules locally and these local schedules collaboratively guarantee the timing and reliability requirements of the rhythmic packets while minimizing the total reliability degradation suffered by other periodic tasks. After executing the dynamic schedules, all the device nodes return to the nominal mode and re-employ the static schedule.

4. $[t_{sp}, t_{ep})$ is the time duration handling the disturbance instead of the last duration of the disturbance which is captured by the rhythmic period vector $\overrightarrow{P_i}$.

In the following, we first present the details of the RD-PaS framework under the Transmission-based Scheduling (TBS) model in Section 4. We then introduce required modifications to support the RD-PaS framework under the PBS model in Section 5.

4 Reliable Scheduling under TBS Model

This section focuses on reliable scheduling under the TBS model. We first describe how RD-PaS constructs a reliable static schedule in the system nominal mode. We then introduce how RD-PaS handles disturbances in the rhythmic mode.

4.1 Reliable Static Scheduling

An RTWN starts running in the nominal mode in which all tasks need to 1) be reliably scheduled to achieve the required e2e PDRs; and 2) meet the e2e timing constraints for all the packet transmissions. That is, we need to solve **P1** defined in Section 3.1. In the TBS model, each specific time slot is assigned to an individual packet transmission. Considering the lossy nature of wireless links, when a transmission is not successful, retransmissions are needed, which require extra time slots. To reduce the demand on network resources, we aim to minimize the number of extra slots for each task while satisfying the reliability requirement (*i.e.*, Constraint 1 in **P1**). On the other hand, we observe that Constraint 2 can be handled separately from Constraint 1 since satisfying Constraint 2 can be treated as a standard transmission scheduling problem once the number of extra time slots is determined for each task. Thus, we intend to first tackle the following sub-problem.

P1.1: Given RTWN $G = (V, E)$ where each link $e \in E$ has an associated PDR, and task set $\mathcal{T}$ in which each task τ_i has a single routing path $\overrightarrow{L_i}$, determine the minimum number of extra slots needed by each task τ_i to satisfy Constraint 1.

To solve **P1.1**, we propose to first determine whether a given number of extra time slots for each task can satisfy

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

Constraint 1 and then search for the optimal number of extra time slots for every task. We will prove later that this approach indeed leads to an exact solution for **P1.1**. We discuss our approach in detail below.

Let $\overrightarrow{R}_{i,j} = [R_{i,j}[0], R_{i,j}[1], \ldots, R_{i,j}[H_i - 1]]$ be the *retry vector* of packet $\chi_{i,j}$, where $R_{i,j}[h]$ denotes the number of time slots assigned to hop h of $\chi_{i,j}$. We use $W_{i,j}$ to denote the total number of time slots assigned to $\chi_{i,j}$, *i.e.*, $W_{i,j} = \sum_{h=0}^{H_i-1} R_{i,j}[h]$. Given the PDRs of all the links along the routing path of τ_i and the retry vector of $\chi_{i,j}$, the e2e PDR of $\chi_{i,j}$, $\lambda_{i,j}$, can be derived as:

$$\lambda_{i,j} = \prod_{h=0}^{H_i-1} \left[1 - (1 - \lambda^L_{L_i[h]})^{R_{i,j}[h]}\right]. \quad (1)$$

According to Constraints 1 and 2 in **P1**, all the packets released by τ_i must meet the same timing and reliability requirements in the system nominal mode. Thus, in the following discussion, we only consider parameter settings (including both the assigned number of slots and the retry vector) for each individual task τ_i instead of each packet $\chi_{i,j}$. For a given number of slots, say w, assigned to τ_i, the number of possible slot allocations, *i.e.* retry vectors, equals to $\binom{w-1}{H_i-1}$. We further introduce the following definitions.

Definition 1 Optimal Retry Vector $\overrightarrow{R}^*_i(w)$: *An optimal retry vector of task τ_i for a given number of slots w is the retry vector that leads to the largest PDR value for the given w, denoted as $\lambda^*_i(w)$, among all the possible allocations.*

Definition 2 Optimal Retry Vector Function $\overrightarrow{R}^*_i(\cdot)$: *The optimal retry vector function of task τ_i is the set of pairs $(w, \overrightarrow{R}^*_i(w))$ such that each $\overrightarrow{R}^*_i(w)$ is the optimal retry vector for the given number of slots w.*

Definition 3 Optimal PDR Function $\lambda^*_i(\cdot)$: *The optimal PDR function of task τ_i is the set of pairs $(w, \lambda^*_i(w))$ such that each PDR value $\lambda^*_i(w)$ corresponds to the optimal retry vector with the given number of slots w.*

As the first step towards satisfying Constraint 1, we present our solution to evaluate the optimal retry vector function $\overrightarrow{R}^*_i(\cdot)$ and the optimal PDR function $\lambda^*_i(\cdot)$ for each task τ_i. As both functions are only related to task τ_i itself, the computation for each task is independent of other tasks. For the sake of clarity, we create a PDR table for each task τ_i to store both $\overrightarrow{R}^*_i(\cdot)$ and $\lambda^*_i(\cdot)$ for all (needed) values of w in each node, such overhead in our implementation is given in Section 7. (An example PDR table can be found in Table 4 in Section 7.) Below, we describe our optimal PDR table generation algorithm, Alg. 1, and prove its optimality.

Alg. 1 iteratively constructs the PDR table. At each iteration, we add one time slot to τ_i at the h-th hop that yields the maximum PDR value λ^*_i and store the resulting retry vector $\overrightarrow{R}^*_i$ into the PDR table (Lines 5-7). The retry vector is initially set to $[1, 1, 1, \ldots]$ and the corresponding PDR value equals to $\prod_{h=0}^{H_i-1} \lambda^L_{L_i[h]}$ (Lines 1-3). Since the required PDR value is λ^R, the iterative process stops when $\lambda^*_i(w) \geq \lambda^R$. We use w^+_i to denote the minimum number of slots that guarantees the reliable delivery of τ_i and the time complexity of Alg. 1 is $O(H_i \cdot w^+_i)$.

Algorithm 1 PDR Table Construction under TBS for Task τ_i

```
Input: $G = (V, E)$, $\tau_i$, $\lambda^R$
Output: PDR table of $\tau_i$ and $w^+_i$
1: $w \leftarrow H_i$;
2: $\overrightarrow{R}^*_i(w) \leftarrow [1, 1, 1, \ldots]$;
3: $\lambda^*_i(w) \leftarrow \prod_{h=0}^{H_i-1} \lambda^L_{L_i[h]}$;
4: while $\lambda^*_i(w) < \lambda^R$ do
5:     $w \leftarrow w + 1$;
6:     Select the hop index $h$ which yields the maximum PDR
       value (computed by Eq. (1));
7:     Update $\overrightarrow{R}^*_i(w)$ and $\lambda^*_i(w)$ in PDR table;
8: end while
9: $w^+_i \leftarrow w$
```

Lemma 1 and Theorem 1 below affirm that Alg. 1 indeed results in the optimal retry vector function $\overrightarrow{R}^*_i(\cdot)$ and optimal PDR function $\lambda^*_i(\cdot)$.

Lemma 1 *For any particular hop h of a packet with w allocated slots, if we allocate both the $(w+1)$-th and $(w+2)$-th slots to h hop and the reliability values of the packet equal to $\lambda^*(w+1)$ and $\lambda^*(w+2)$, respectively, we have $\frac{\lambda^*(w+2)}{\lambda^*(w+1)} < \frac{\lambda^*(w+1)}{\lambda^*(w)}$.*

Proof of Lemma 1: Let $\mathcal{G}(R^*(w)[h], \lambda^L_{L[h]}) = \frac{\lambda^*(w+1)}{\lambda^*(w)}$ be a function of $R^*(w)[h]$ and $\lambda^L_{L[h]}$. Then Lemma 1 implies that when $\lambda^L_{L[h]}$ is set to an arbitrary value λ_0, $\mathcal{G}_{\lambda_0} = \mathcal{G}(R^*(w)[h], \lambda_0)$ is a monotonically decreasing function of $R^*(w)[h]$. If we update $\overrightarrow{R}^*(w)$ by allocating one slot at an arbitrary hop h-th, according to Eq. (1), we only need to update $\lambda^*(w)$ by replacing the term $1 - (1 - \lambda^L_{L[h]})^{R^*(w)[h]}$ by $1 - (1 - \lambda^L_{L[h]})^{R^*(w)[h]+1}$ to get $\lambda^*(w+1)$. That is,

$$\mathcal{G}(R^*(w)[h], \lambda^L_{L[h]}) = \frac{\lambda^*(w+1)}{\lambda^*(w)} = \frac{1 - (1 - \lambda^L_{L[h]})^{R^*(w)[h]+1}}{1 - (1 - \lambda^L_{L[h]})^{R^*(w)[h]}}$$

Thus, if $\lambda^L_{L[h]}$ is fixed to λ_0, we have:

$$\mathcal{G}'_{\lambda_0} = \frac{\partial \mathcal{G}(R^*(w)[h], \lambda_0)}{\partial R^*(w)[h]} = \frac{\lambda_0 \cdot (1 - \lambda_0)^{R^*(w)[h]} \log(1 - \lambda_0)}{\left((1 - \lambda_0)^{R^*(w)[h]} - 1\right)^2}$$

Since $0 < \lambda^L_{L[h]} < 1$ and $(1 - \lambda^L_{L[h]})^{R^*(w)[h]} > 0$, we have $\mathcal{G}'_{\lambda_0} < 0$. Further, $\mathcal{G}_{\lambda_0}$ decreases monotonically as $R^*(w)[h]$ increases. Thus, Lemma 1 holds. □

Theorem 1 *For any given number of time slots, w, no other retry vector can yield a larger PDR value than $\overrightarrow{R}^*_i(w)$ as computed by Alg. 1.*

Proof of Theorem 1: We prove the theorem by mathematical induction, *i.e.*, for any $w = H, H+1, \ldots, w^+$, the retry vector $\overrightarrow{R}^*(w)$ determined by Alg. 1 can achieve the largest PDR value $\lambda^*(w)$. (Here we omit the task index i since only one task is considered).

Base case: When $w = H$, the statement holds as only one possible retry vector exists, *i.e.*, $\overrightarrow{R}^*(H) = [1, 1, \ldots, 1]$.

Inductive step: Suppose the PDR value of $\overrightarrow{R}^*(w)$ is largest among that of all possible retry vectors when $w = k, k > H$, we should prove that the PDR value of $\overrightarrow{R}^*(k+1)$ obtained by Alg. 1, *i.e.* $\lambda^*(k+1)$ is also the largest. We prove this by contradiction.

Suppose there exists another retry vector (denoted as $\overrightarrow{R}^o(k+1)$) that leads to a larger PDR value, *i.e.*, $\lambda^*(k+1) <$

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

$\lambda^o(k+1)$. Since the total number of slots assigned to the task (*i.e.*, the sum of all elements in the retry vectors) both equal to $k+1$ and $\overrightarrow{R}^*(k+1) \neq \overrightarrow{R}^o(k+1)$, we can always find one hop at which the number of assigned slots in $\overrightarrow{R}^o(k+1)$ is larger than that in $\overrightarrow{R}^*(k+1)$. We use q to denote this hop index and $R^o(k)[q]$ to denote the number of slots assigned at the q-th hop in $\overrightarrow{R}^o(k)$. Then, $R^o(k+1)[q] > R^*(k+1)[q]$. Suppose $\overrightarrow{R}^*(k+1)$ is achieved by adding one slot at the p-th hop in $\overrightarrow{R}^*(k)$.

Case 1: $p = q$. In this case, $\overrightarrow{R}^*(k+1)$ and $\overrightarrow{R}^o(k+1)$ are both achieved by adding one slot at the p-th hop in $\overrightarrow{R}^*(k)$ and $\overrightarrow{R}^o(k)$, respectively. Then, according to Lemma 1, $\lambda^*(k+1)$ and $\lambda^o(k+1)$ can be rewritten with $\mathcal{G}(R^*(w)[h], \lambda^L_{L[h]})$ function as follows:

$$\lambda^*(k+1) = \lambda^*(k) \cdot \mathcal{G}(R^*(k)[p], \lambda^L_{L[p]}),$$

$$\lambda^o(k+1) = \lambda^o(k) \cdot \mathcal{G}(R^o(k)[p], \lambda^L_{L[p]}).$$

According to the assumption that the PDR value of $\overrightarrow{R}^*(k)$ is the largest, we have $\lambda^*(k) \geq \lambda^o(k)$. Since $R^*(k)[p] < R^o(k)[p]$, according to Lemma 1, we have $\mathcal{G}(R^*(k)[p], \lambda^L_{L[p]}) > \mathcal{G}(R^o(k)[p], \lambda^L_{L[p]})$. Then, $\lambda^*(k+1) > \lambda^o(k+1)$. This contradicts our assumption.

Case 2: $p \neq q$. In this case, $\lambda^*(k+1)$ and $\lambda^o(k+1)$ can be rewritten as:

$$\lambda^*(k+1) = \lambda^*(k) \cdot \mathcal{G}(R^*(k)[p], \lambda^L_{L[p]}),$$

$$\lambda^o(k+1) = \lambda^o(k) \cdot \mathcal{G}(R^o(k)[q], \lambda^L_{L[q]}).$$

As $\lambda^*(k+1) < \lambda^o(k+1)$ and $\lambda^*(k) \geq \lambda^o(k)$, it must holds that

$$\mathcal{G}(R^*(k)[p], \lambda^L_{L[p]}) < \mathcal{G}(R^o(k)[q], \lambda^L_{L[q]}). \quad (2)$$

Since $R^*(k)[q] < R^o(k)[q]$ according to the assumption, the following inequality holds:

$$\mathcal{G}(R^*(k)[q], \lambda^L_{L[q]}) > \mathcal{G}(R^o(k)[q], \lambda^L_{L[q]}). \quad (3)$$

Combining Eq. (2) and Eq. (3), we have $\mathcal{G}(R^*(k)[p], \lambda^L_{L[p]}) < \mathcal{G}(R^*(k)[q], \lambda^L_{L[q]})$. Further,

$$\lambda^*(k) \cdot \mathcal{G}(R^*(k)[p], \lambda^L_{L[p]}) < \lambda^*(k) \cdot \mathcal{G}(R^*(k)[q], \lambda^L_{L[q]}).$$

This means that if we allocate one slot at the q-th hop in $\overrightarrow{R}^*(k)$ instead of at the p-th hop, we can have a larger PDR value. This contradicts with Alg. 1 which allocates one slot at the hop which yields the largest PDR value at each iteration.

Since both cases lead to contradiction, the inductive step is proved. Thus, Theorem 1 holds for all values of w. □

Now with the functions $\overrightarrow{R}^*_i(\cdot)$ and $\lambda^*_i(\cdot)$ being determined, we have successfully solved **P1.1**. To satisfy Constraint 2 in **P1**, we need to create a static schedule, *i.e.*, specifying when a packet uses a slot, to ensure that real-time constraints are met. We introduce an observation that helps map the reliable static schedule generation problem, *i.e.*, **P1**, to a conventional real-time scheduling problem.

Observation 1 *Given task set $\mathcal{T}$ to be reliably scheduled, if we set the number of slots for τ_i to w^+_i according to $\lambda^*_i(\cdot)$*[5]*, w^+_i is then equivalent to the execution time of τ_i. Then, each task $\tau_i \in \mathcal{T}$ with P_i, D_i and w^+_i can be mapped to a task in a conventional real-time task set with the same period, deadline and execution time. Thus, a feasible schedule for the corresponding conventional real-time task set is also a feasible schedule under which all tasks in $\mathcal{T}$ can be reliably delivered.*

Given the schedule specifying the slot assignment for each task, each node can further allocate specific slots to the transmission at each hop according to the retry vector function $\overrightarrow{R}^*_i(\cdot)$. Thus, given a task set to be reliably scheduled in an RTWN, the network can adopt any conventional real-time scheduling algorithm (Earliest-Deadline-First (EDF) [33] in this work) to generate a static schedule that guarantees to meet all the constraints in **P1**. Since we allow at most one transmission within each timeslot, determining the nominal-mode schedule (*i.e.*, **P1**) can be mapped to a uni-processor scheduling problem.

4.2 Reliable Dynamic Scheduling

Our proposed solution for **P1** ensures that both timing and reliability requirements are met in the system nominal mode. However, upon the detection of any disturbance, the corresponding tasks enter their rhythmic states and follow new release patterns and deadlines as shown in Fig. 2. The static schedule may no longer be able to meet both requirements especially for all the critical rhythmic packets. Therefore, a well-designed reliable dynamic packet scheduling mechanism is needed to enable the system to adapt to any workload change after the detection of a disturbance.

In our RD-PaS framework, the network generates the static schedule by assigning w^+_i slots to each task τ_i according to the retry vector function. When a disturbance is detected and reported to the controller node, the system follows the execution model outlined in Section 3.2 to switch to the rhythmic mode. The main challenge here is to generate a temporary dynamic schedule when tasks cannot be reliably delivered after the rhythmic tasks (in $\mathcal{T}_{Rhy}$) enter their rhythmic states. That is, problem **P2*** needs to be solved. The dynamic schedule must be able to accommodate the increased rhythmic workload and minimizes the degradation of both timing and reliability performance of other periodic tasks. Specifically, all the rhythmic packets must meet their timing and reliability requirements. That is, Constraints 3 and 4 are satisfied.

To ensure this, we may have to sacrifice the reliability requirements, *i.e.* lowering the e2e PDR values of some periodic packets, or even sacrifice their timing requirements, *i.e.* dropping some periodic packets. That is, the number of slots assigned to each packet may need to be updated. Since the PDR table for each task containing both the retry vector function $\overrightarrow{R}^*_i(\cdot)$ and PDR function $\lambda^*_i(\cdot)$ is pre-calculated and stored at each node, V_c only needs to piggyback on a broadcast packet the information of the updated total number of slots ($W_{i,j}$) assigned to each periodic packet,

5. All the retry vectors for other w values stored in $\overrightarrow{R}^*_i(\cdot)$ are used in the dynamic schedule generation, which will be discussed in Section 4.2.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

and then each node can decode the updated retry vector accordingly, once it receives this information.[6]

To formally define the dynamic schedule generation problem, we introduce some concepts/notation. Let Γ denote the *active packet set* containing all the packets to be scheduled within the rhythmic mode duration $[t_{sp}, t_{ep})$. Since the payload size of a broadcast packet is bounded, we set an upper bound on the number of periodic packets whose $W_{i,j}$ can be changed, and denote it as α. To capture the reliability degradation for periodic packet $\chi_{i,j}$, let $\delta_{i,j}$ represent the difference between the required PDR λ^R and the updated PDR value $\lambda_{i,j} = \lambda_i^*(W_{i,j})$ in the dynamic schedule, *i.e.*, $\delta_{i,j} = \max\{0, \lambda^R - \lambda_{i,j}\}$. Note that the timing degradation of each packet can also be captured by $\delta_{i,j}$ where $\delta_{i,j} = \lambda^R$ if $\chi_{i,j}$ is dropped. Now the dynamic schedule generation problem, which is defined formally below, becomes finding $W_{i,j}$ for each periodic packet in Γ to satisfy Constraint 3 and 4.

P2: Given the active packet set Γ, the PDR function $\lambda_i^*(\cdot)$ of each task τ_i, the maximum allowed number of updated packets α, determine the updated packet set $\rho = \{W_{i,j} | \chi_{i,j} \in \Gamma\}$ such that i) the size of ρ is not larger than α, *i.e.*, $|\rho| \leq \alpha$, and ii) the total reliability degradation is minimized, *i.e.*, $\forall \chi_{i,j} \in \rho, \min \sum \delta_{i,j}$.

Given that determining the updated packet set (*i.e.* solving **P2**) is NP-hard [3], we propose a heuristic method to solve **P2** and the high-level idea is as follows. Since dropping any packet $\chi_{i,j}$ leads to a significant decrease in the PDR value of $\chi_{i,j}$, *i.e.*, $\delta_{i,j} = \lambda^R$, we prefer to always allocate at least the basic number of slots (*i.e.*, H_i) to each packet. If the network bandwidth is sufficient, we assign extra slots to periodic packets in a greedy manner according to their PDR degradation. Alg. 2 summarizes the updated packet set generation algorithm which uses the greedy extra slots assignment heuristics described in Alg. 3. Specifically, at each iteration, Alg. 3 adds one slot to the packet resulting in the minimum PDR degradation after an extra slot has been assigned. Using Alg. 2 and Alg. 3, the updated packet set can be determined in $\mathcal{O}(\alpha \cdot W_{max})$ time where W_{max} is the maximum w_i^+ among all the tasks.

Handling disturbance in RD-PaS introduces extra time overhead including computation delay and communication delay. The computation delay at the gateway is negligible given the efficient heuristics in Alg. 2 and Alg. 3. The communication delay consists of the disturbance report delay and dynamic schedule broadcast delay. The former part is bounded within one period of the reporting task. The latter part is determined by the network scale and the broadcast mechanism deployed in the system which is not the focus of this work. Note that the proposed RD-PaS framework can be readily extended to handle concurrent disturbances in RTWNs, following the similar way as elaborated in [16].

5 Reliable Scheduling under PBS Model

In this section, we discuss how to support the RD-PaS framework under the packet-based scheduling (PBS) model. At the highest level, reliable scheduling under the PBS

6. In the system rhythmic mode, we adjust the assigned number of slots for each packet instead of each task for more flexibility.

Algorithm 2 Updated Packet Set Generation

```
Input: $\Gamma, \alpha, \lambda_i^*(w)$
Output: $\rho$
 1: Schedule the rhythmic packets in $\Gamma$ using $w_0^+$;
    //Suppose $n$ is the number of periodic packets in $\Gamma$
 2: if all periodic packets in $\Gamma$ can be reliably scheduled then
 3:     No packet needs to be updated;
 4: else
 5:     Find the first $n - \alpha$ schedulable periodic packets with
        the minimum $w_i^+$ using the packet-dropping heuristic in
        [15];
 6:     if Such $n - \alpha$ periodic packets can be found then
 7:         if the $\alpha$ packets can be scheduled using $H_i$ then
 8:             Assign extra slots to the $\alpha$ packets by Alg. 3;
 9:         else
10:             Determine the dropped packet set (suppose $m$ pack-
                ets) using the dropping heuristics in [15];
11:             Assign extra slots to the $\alpha - m$ packets by Alg. 3;
12:         end if
13:     else
14:         Drop all the periodic packets;
15:     end if
16: end if
```

Algorithm 3 Extra Slots Assignment

```
 1: $\mathcal{S}_{extra} \leftarrow$ {Packets to be assigned extra slots};
 2: while $\mathcal{S}_{extra} \neq \emptyset$ do
 3:     Add one slot to packet $\chi_s$ if doing so leads to the
        minimum PDR degradation;
 4:     if the system is schedulable then
 5:         if $\chi_s$ is already reliable then
 6:             Remove $\chi_s$ from $\mathcal{S}_{extra}$;
 7:         end if
 8:     else
 9:         Reduce one slot from $\chi_s$;
10:         Remove $\chi_s$ from $\mathcal{S}_{extra}$;
11:     end if
12: end while
```

model has two main differences from that under the TBS model. First, since each time slot is assigned to a specific packet instead of a dedicated hop, retry vector $\overrightarrow{R}_{i,j}$ and its function $\overrightarrow{R}_i^*(\cdot)$ are no longer needed. Second, the computation for PDR function $\lambda_i^*(\cdot)$ is different because the time slot allocation mechanism has changed.

Since the PDR function is a key parameter in checking reliability, we first describe how to compute the PDR value for a task with a given number of slots in PBS. Let $Pr_i(0, w)$ denote the probability of a packet of τ_i staying in the source node within w slots; $Pr_i(h, w)$ denote the probability of a packet of τ_i being transmitted to the receiver of the h-th hop along the routing path ($1 \leq h \leq H_i$), and have not been successfully forwarded, within w slots. $Pr_i(h, w)$ can be computed by:

$$Pr_i(h,w) = \begin{cases} 1 & w = h = 0 \\ (1 - \lambda^L_{L_{i[h]}}) Pr_i(h, w-1) & w > h = 0 \\ \lambda^L_{L_{i[h-1]}} Pr_i(h-1, w-1) & w = h > 0 \\ Pr_i(h, w-1) + & \\ \lambda^L_{L_{i[h-1]}} Pr_i(h-1, w-1) & w > h = H_i \\ (1 - \lambda^L_{L_{i[h]}}) Pr_i(h, w-1) + & \\ \lambda^L_{L_{i[h-1]}} Pr_i(h-1, w-1) & \text{otherwise.} \end{cases} \quad (4)$$

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

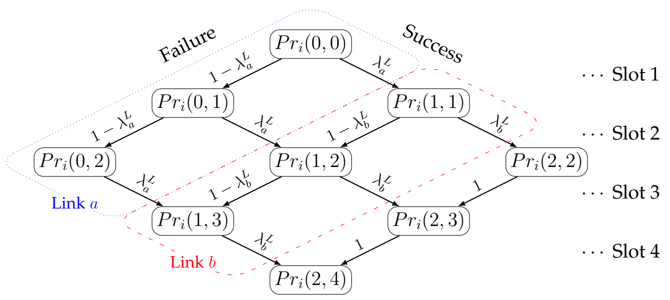


Fig. 3: PDR computation for a task with two hops under the PBS model.

The computation of $Pr_i(h,w)$ is iteratively conducted based on the status of τ_i at its previous transmission, i.e., $Pr_i(h^*,w-1)$ when τ_i is assigned with $w-1$ time slots. Specifically, when $w=0$, h must equal to 0 since it indicates that a packet of τ_i stays in the source node without any assigned time slot. In this case, $Pr_i(0,0)$ equals 1 as no transmission happens. When $w>0$, $Pr_i(h,w)$ is computed according to the different cases of h and the status of the previous transmission. If $w>h=0$, it indicates that a packet of τ_i experiences w transmission failures and stays at the source node. If $w=h>0$, it indicates that a packet of τ_i has been successfully transmitted for each hop $1,2,...,h$. If $w>h>0$, it indicates that a packet of τ_i has been successfully transmitted for h hops using w slots. Note that, depending on whether $h=H_i$, the computation of $Pr_i(h,w)$ is slightly different. Below, we use an example task with 2 hops (links a and b with PDR λ_a^L and λ_b^L, respectively) and 4 slots to describe the computation of $Pr_i(h,w)$ in Fig. 3.

As shown in the figure, $Pr_i(h,w)$ can be either reached by $Pr_i(h-1,w-1)$, followed by a successful transmission ($\lambda^L_{L_i[h-1]}$), or $Pr_i(h,w-1)$, followed by a failed transmission ($1-\lambda^L_{L_i[h]}$), except for boundary conditions. These boundary conditions include the following:

Case 1: When $w=h=0$, the source node generates a packet ($Pr_i(0,0)=1$).
Case 2: When $w>h=0$, it is not possible for $Pr_i(h,w)$ to be reached by $Pr_i(h-1,w-1)$ ($Pr_i(0,1)$, $Pr_i(0,2)$ in the figure). Thus only $Pr_i(h,w-1)$ is considered.
Case 3: When $w=h>0$, it is not possible for $Pr_i(h,w)$ to be reached by $Pr_i(h,w-1)$ ($Pr_i(1,1)$, $Pr_i(2,2)$ in the figure). Thus only $Pr_i(h-1,w-1)$ is considered.
Case 4: When $w>h=H_i$, $Pr_i(h,w-1)$ always reaches $Pr_i(h,w)$ ($Pr_i(2,3)$, $Pr_i(2,4)$ in the figure).

Different from TBS, which finds the optimal PDR values by using retry vectors for a given w, the PDR values in PBS is solely determined by w, *i.e.*, $\lambda_i^*(w)=Pr_i(H_i,w)$. Based on Eq.(4), we propose a dynamic programming algorithm (Alg. 4) to compute $Pr_i(h,w)$ and finally $\lambda_i^*(w)$. In Alg. 4, the iteration starts from $w=1$, and stops when λ^R is reached. In each iteration, it computes all $Pr_i(h,w)$ for $0\le h\le H_i$, and stores them to $\lambda_i^*(\cdot)$ if $w\ge H_i$.

After the PDR function is computed, we can apply the same method proposed in Section 3.2 and 4 to generate reliable static and dynamic schedules, respectively. More specifically, we use Observation 1 with computed PDR function to generate a reliable static schedule, and use Alg. 2 and

Algorithm 4 PDR Table Computation under PBS for Task τ_i

```
Input: $G=(V,E)$, $\tau_i$, $\lambda^R$
Output: The PDR function of $\tau_i$ and $w_i^+$
1: $w \leftarrow 0$;
2: while $\lambda_i(w) < \lambda^R$ or $w < H_i$ do
3:     $w \leftarrow w+1$;
4:     for $h=0$ to $H_i$ do
5:         Compute $Pr_i(h,w)$ following Eq.(4);
6:     end for
7:     if $w >= H_i$ then
8:         $\lambda_i^*(w) \leftarrow Pr_i(H_i,w)$;
9:     end if
10: end while
11: $w_i^+ \leftarrow w$
```

Alg. 3 to determine the updated W in the rhythmic mode. Note that, the retransmission mechanism for the broadcast task under both TBS and PBS models shows some slight differences and readers are referred to [3] for the details.

6 Shared Slot Transmission Mechanism

In the previous sections, we meet the tasks' reliability requirements by assigning a number of retransmission slots which are specifically assigned to either each hop or each packet in the TBS model or PBS model, respectively. To improve the network slot utilization, in this section, we propose a novel shared slot transmission mechanism, called TG-STM, for both TBS and PBS models. Below we first present a motivational example to illustrate that slots may be significantly wasted under the dedicated slot transmission mechanism (DTM) and then present an overview of TG-STM. After that, we describe the details of TG-STM for the TBS model and the PBS model, respectively.

6.1 Motivation and Overview of TG-STM

Consider task τ_1 with a required e2e PDR value $\lambda^R=0.99$. Its routing path consists of two hops, each of which corresponds to a link with PDR value $\lambda^L=0.9$. According to Alg. 1 and Alg. 4, τ_1 should be allocated with 4 and 2 retransmission slots to satisfy λ^R under TBS and PBS models, respectively. Table 2 shows the probability of each dedicated time slot being wasted (not used for transmitting a packet of τ_1). For example, the first hop of τ_1 can be successfully transmitted within 2 time slots (Slot 1 and Slot 2) under the TBS model with a probability of 99%. This means that the third slot (Slot 3) allocated to τ_1's first hop has a probability of 99% being wasted. Note that such slot waste happens for every packet released by τ_1. Taking the task set running in our later RTWN testbed as an example (the task and network specifications will be provided later in Table 3 and Fig. 5, respectively), the number of wasted slots with a probability higher than 90% is 126 (56) within each hyperperiod (360 slots) under the TBS (PBS) model.

TABLE 2: Time Slot Waste under the TBS/PBS Models.

	Slot 1	**Slot 2**	**Slot 3**	**Slot 4**	**Slot 5**	**Slot 6**
TBS	1 hop			2 hop		
	0%	90%	99%	0%	90%	99%
PBS	0%	0%	81%	97%	-	-

As a consequence of the slot waste, more time slots have to be allocated to each task (*i.e.*, a larger w^+) to satisfy the

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

reliability requirement. This leads to an unnecessarily higher system workload. Even worse, if the system is overloaded, Constraint 1 and 2 in **P1** cannot be satisfied simultaneously. This motivates us to explore more efficient packet transmission mechanisms in the RD-PaS framework to reduce the number of wasted slots, and thus to improve the network resource utilization.

As observed in the motivational example, the main reason behind such slot waste is that each slot is dedicated to a particular hop (packet) of a task in the TBS (PBS) model. If the transmission of a hop (packet) succeeds, all the following allocated slots are wasted. Therefore, an intuitive insight is to let each slot be shared by different tasks. In this case, the probability of a slot being wasted can be significantly reduced. However, realizing such slot sharing is non-trivial especially given the **deterministic** communication requirement by RTWNs. By deterministic, we mean that under a certain scheduling policy, we are able to determine the reliability value that can be achieved by each task τ_i (*i.e.*, λ_i) prior to the network operation.

Designing a deterministic shared slot transmission mechanism in RD-PaS needs to tackle the following challenges. First, RD-PaS relies on a local schedule generation framework where each node determines the schedule locally based on the information stored at the node. Given the limited memory resource at each node, the additional required information to be stored locally to support the shared slot mechanism should be affordable. Second, the number of slots that can be used as shared slots varies among packets released by individual tasks. This is because each packet can only use the slots (i) scheduled for other tasks as the shared slots and (ii) within the time duration between the packet's release time and deadline. Such a varing number of shared slots requires pessimistic estimation on the number of required retransmission slots for each task. Third, we need to design a method to calculate the achieved reliability value for each task under the shared slot mechanism before the network starts operation.

To overcome the above challenges, we design a *task-level* and *grouped* shared slot transmission mechanism, referred to as TG-STM, for both TBS and PBS models. As a task-level mechanism, TG-STM follows DTM to assign a same number of transmission slots (*i.e.*, w_i^+) for all packets released by each task τ_i. As a grouped mechanism, TG-STM groups all the tasks into multiple task groups where slots can only be shared among the tasks within the same group. TG-STM guarantees that the number of shared slots supplied to all the packets from each task remains the same. To achieve the required functions of TG-STM, the following questions need to be answered. i) How to group all the tasks into different task groups? ii) How to design the slot sharing rules among tasks within a group to guarantee a same required number of shared slots among packets released by a task? iii) How to calculate the achieved reliability value for each task under TG-STM? In the following, we present our solution under the TBS model and the PBS model in Section 6.2 and Section 6.3, respectively.

6.2 TG-STM under TBS Model

As elaborated in Section 6.1, TG-STM is acceptable only if the following two requirements are satisfied. (i) Each time slot is shared by tasks in a deterministic manner. That is, tasks sharing a same slot should be granted access to the slot in a deterministic order during run time. (ii) The specification of each shared slot, including which tasks (or even which hops in the TBS model) sharing this slot and the sharing order, is pre-determined and the achieved reliability of each task under TG-STM can be calculated before the network starts.

To satisfy the first requirement, we adopt Multi-Priority MAC (MP-MAC), an enhancement to the IEEE 802.15.4e standard proposed in [17], to support prioritization of task transmissions. MP-MAC enables the transmitter to indicate the priority of the transmission by adjusting the Start-Of-Frame (SOF) time offset in the time slot. Tasks sharing the same slot are granted access to the channel according to their MP-MAC priorities. If the transmission of the task with the highest priority has been successfully transmitted before this shared slot, the task(s) with lower priorities can use this shared slot to transmit. Therefore, different from traditional shared slot transmission mechanisms in RTWNs (*e.g.*, [25]) which only provide *opportunistic* transmissions by allowing links (or nodes) to compete for a shared slot to transmit packets, TG-STM relies on MP-MAC to support tasks to share each time slot in a deterministic manner.

Satisfying the second requirement is more complicated. Below, we first explain the reason of designing TG-STM as a task-level and isolated shared slot transmission mechanism to satisfy the second requirement. Then we present the details of TG-STM under the TBS model.

First, the granularity of slot sharing among tasks (*i.e.*, how many slots are shared by which tasks) impacts the network bandwidth required by the system to be schedulable. For example, if we allow all tasks running in the system to share every slot in the slotframe, the utilization of each time slot is high and the system schedulability may be satisfied using the minimum number of slots. However, such mechanism is impractical since it requires a huge number of shared slot information to be stored in each memory constrained device node (*e.g.*, 32K RAM in TI CC2538 SoC) to generate the schedule locally. Particularly, besides the task specifications, each node needs to record the priority index of every transmission of all the tasks in each time slot within the slotframe[7]. Since the slotframe length can be relatively long and storing the slot offset is memory consuming, this motivates us to design a mechanism under which slot sharing rule of transmissions of each task is fixed for all its released packets. In this case, the number of transmission slots needed by each packet of a same task remains the same. We call this mechanism task-level slot sharing mechanism, *i.e.*, each task τ_i is assigned with a fixed number of transmission slots for all its released packets.

Second, each packet $\chi_{i,j}$ released by task τ_i can only use the slots assigned to other tasks within its own time window (*i.e.*, $[r_{i,j}, d_{i,j})$) as shared slots. Thus, for a given schedule, the number of shared slots can be different among the packets released by a same task. Similarly, the relative positions between these shared slots and $\chi_{i,j}$'s dedicated

7. Since different transmissions of each task correspond to particular links with different PDRs, each time slot can be shared by an arbitrary combination of all tasks' transmissions and the priorities of all transmissions in different slots also vary according to the schedule.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

slots can also be different. According to the PDR table generation under both TBS and PBS models, this results in different achieved reliability values for different packets $\tau_{i,j}$. To satisfy the task-level slot sharing requirement mentioned above, we have to set the maximum required number of dedicated and shared slots among all $\chi_{i,j}$ as w_i^+ for task τ_i. This introduces significant pessimism on the network bandwidth demand by the task set to satisfy its reliability requirement. Furthermore, since the shared slots assigned to each packet $\chi_{i,j}$ are different, each node on τ_i's routing path must store the shared slot information for each $\chi_{i,j}$. Apparently, this posts high memory overhead on each device to store the slot offsets of shared slots and priorities of all transmissions released by each packet. Therefore, we propose to address the above two issues by designing a grouped shared slot transmission mechanism to guarantee that the shared slots allocated to all the packets by a task remain the same.

The key challenge in designing TG-STM mainly lies on the fact that the relative positions of transmissions from different tasks keep changing in the schedule. Below, we use an example to illustrate such challenge and introduce an important observation to help us design the task-level and grouped TG-STM.

Example 1 *Consider three tasks τ_1, τ_2 and τ_3 with periods $P_1 = 5$, $P_2 = 13$ and $P_3 = 15$, respectively. Assume that τ_1, τ_2 and τ_3 are allocated with 2, 3 and 2 dedicated slots, respectively. τ_2 and τ_3 use the dedicated slots of τ_1 as shared slots to transmit. The system adopts the EDF scheduler. Fig. 4 shows the shared slots (from τ_1) of τ_2 and τ_3. According to EDF, 6 dedicated slots assigned to τ_1 fall into the time window of τ_2's first packet $\tau_{2,1}$ but only 5 dedicated slots of τ_1 fall into the time window of packet $\tau_{2,2}$. However, each packet of τ_3 (i.e., $\tau_{3,1}$ and $\tau_{3.2}$) shares the same number of dedicated slots (i.e., 6 slots) from τ_1. This is because that τ_1 and τ_3 have harmonic periods (i.e., $P_1 = 5$, $P_3 = 15$) such that a fixed number of slots from the task with the shorter period fall into the time window of each packet of the task with the larger period.*

On the other hand, although the number of shared slots can be used by each packet of τ_3 remains the same (6 slots from τ_1), the relative positions between these shared slots and the dedicated slots for τ_3 are different among different packets in the schedule (see Fig. 4). This results in different shared slots assigned to different packets of a same task. One can further observe that the relative positions between the transmissions of $\tau_{1,1}$ and $\tau_{3,1}$ ($\tau_{1,4}$ and $\tau_{3,2}$) remain unchanged. That is, the transmissions of $\tau_{1,1}$ ($\tau_{1,4}$) are always scheduled before those of $\tau_{3,1}$ ($\tau_{3,2}$) under the EDF scheduler because the deadline of any packet of τ_1 is always before the deadline of a simultaneously released packet of τ_3. This example gives us the following observation.

Observation 2 *Given two tasks with harmonic periods, the transmissions of any packet released by the task with the shorter period are always scheduled before the transmissions of a simultaneously released packet (if any) from the task with the larger period.*

According to Observation 2, the assigned dedicated slots for packets of τ_i can be used as shared slots by the simultaneously released packets of τ_j, if τ_i and τ_j are harmonic tasks and $P_i < P_j$. This guarantees that i) the number of

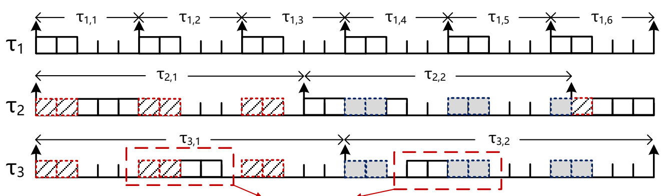


Fig. 4: Illustration of τ_1's dedicated slots and the shared slots of τ_2 and τ_3 in Example 1. The solid blocks denote tasks' dedicated slots and the dashed blocks in different colors denote the shared slots assigned to different packets of a same task.

shared slots assigned to each packet of τ_j is the same, and ii) the relative positions of the dedicated and shared slots keep unchanged for all the packets of τ_j. Therefore, we can group tasks into harmonic task groups, each of which consists of tasks with harmonic periods[8] and slots can be shared among tasks within a group in a predetermined manner.

Intuitively, one may simply group all the tasks with harmonic periods into one group so that shared slots can be utilized by tasks as many as possible. However, it is more advantageous to limit the number of tasks within each harmonic task group to be two due to the following three considerations. First, the probability of a shared slot being used by a certain task during run time decreases significantly as the number of tasks sharing this slot increases. For example, suppose three transmissions with a same corresponding link PDR of 0.8 share one time slot and the first transmission uses this slot as a dedicated retransmission slot. Then, the probability of this slot being used by the second transmission equals to 0.8, and the probability of this slot being used by the third transmission decreases to 0.64. Second, for all the transmissions being assigned in a shared slot, both of their senders and receivers need to remain active[9] thus consume additional energy. Third, MP-MAC only supports a limited number of priority levels, *i.e.*, the number of tasks that can share a time slot. The more priority levels to be enabled in MP-MAC, the longer slot length is needed which may reduce the network throughput.

Therefore, TG-STM groups (unicast) tasks in the system into harmonic task groups, where each group $HG_k (1 \leq k \leq I)$ contains two tasks and I is the total number of harmonic task groups. Within each HG_k, the dedicated slots allocated to the task with the shorter period can be used as the shared slots by the task with the larger period.[10] Tasks not belonging to any harmonic task group (*e.g.*, tasks with prime periods) are stored in a special group HG^*. To enable a larger total number of shared slots among the tasks in the system, we aim to maximize the number of harmonic task groups (*i.e.* I) and thus formulate the following task grouping problem.

8. Harmonic periods have been widely used in industrial applications ranging from radar dwell tasks and robotics to control systems with nested feedback loops [34]. If all tasks are with inharmonic periods, some existing methods (*e.g.*, [35]) can be applied to tune the applications' original periods into harmonic ones.

9. In a shared slot, the sender makes a transmission attempt within its assigned *TxOffset* and the receiver keeps listening for a potential transmission.

10. If two tasks have the same period, the earlier scheduled task makes it's dedicated slots as shared slots to be used by the later scheduled one.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

Algorithm 5 Task Grouping Heuristic

```
Input: $\mathcal{T} = \{\tau_0, \tau_1, \ldots, \tau_n\}$
Output: $\mathcal{T}' = \{HG_{k(1 \le k \le I)}, HG^*\}$
 1: for $\tau_i \in \mathcal{T}$ do
 2:   if $P_i$ is a prime number then
 3:     $HG^* \leftarrow HG^* \bigcup \{\tau_i\}$;
 4:     $\mathcal{T} \leftarrow \mathcal{T} \setminus \{\tau_i\}$;
 5:   end if
 6: end for
 7: Sort tasks in $\mathcal{T}$ in the ascending order of tasks' periods;
 8: $k \leftarrow 1$, $Flag \leftarrow$ false, $\mathcal{T}' \leftarrow \{HG^*\}$;
 9: for $\tau_i \in \mathcal{T}, i = 0, 1, 2, \ldots$ do
10:   for $\tau_j (j > i) \in \mathcal{T}$ do
11:     if $P_i$ and $P_j$ are harmonic numbers then
12:       $HG_k \leftarrow \{\tau_i, \tau_j\}$;
13:       $\mathcal{T} \leftarrow \mathcal{T} \setminus \{\tau_i, \tau_j\}$;
14:       $\mathcal{T}' \leftarrow \mathcal{T}' \bigcup \{HG_k\}$;
15:       $Flag \leftarrow$ true;
16:       $k \leftarrow k + 1$;
17:       Break;
18:     end if
19:   end for
20:   if $Flag$ is true then
21:     Continue;
22:   end if
23:   $\mathcal{T} \leftarrow \mathcal{T} \setminus \{\tau_i\}$;
24:   $HG^* \leftarrow HG^* \bigcup \{\tau_i\}$;
25: end for
26: return $\mathcal{T}'$;
```

Task Grouping Problem. Given a task set $\mathcal{T} = \{\tau_0, \tau_1, \ldots, \tau_N\}$, any two tasks with harmonic periods can form a harmonic task group $HG_k (1 \le k \le I)$. The objective of the task grouping problem is to maximize I, *i.e.* the total number of harmonic task groups in $\mathcal{T}$.

We design a greedy heuristic to solve the task grouping problem[11]. Alg. 5 gives the pseudo code of the proposed algorithm. Initially, all tasks with prime periods are removed from $\mathcal{T}$ and stored in group HG^* (Lines 1-6). All the remaining tasks in $\mathcal{T}$ are sorted in the ascending order of tasks' periods (Line 7). Then, for each task $\tau_i \in \mathcal{T}, i = 0, 1, 2, \ldots$, we select the task with the minimum harmonic period with P_i to form a harmonic group HG_k. If there is no task with harmonic period with τ_i, τ_i is added to group HG^* (Lines 9-25). Finally, the task set $\mathcal{T}'$ containing harmonic task groups $HG_k (1 \le k \le I)$ and the special task group HG^* is returned (Line 26). The time complexity of Alg. 5 is $O(N)$ where N is the number of unicast tasks in $\mathcal{T}$.

Given task set $\mathcal{T}$ being grouped into harmonic task groups, we describe how TG-STM generates the PDR table for the task being assigned with shared slots, *i.e.* the task with the larger period in each harmonic task group HG_k. The task with the shorter period is assigned with only dedicated time slots, and its PDR table is calculated according to Alg. 1. In the following, we denote τ_l as the task using shared slots and τ_s as the task contributing shared slots in each harmonic task group HG_k. We use w'_l to represent the total number of slots allocated to each packet of τ_l (including both the shared slots from τ_s and the dedicated slots of τ_l) and $\overrightarrow{R}'_l = [R'_l[0], R'_l[1], \ldots, R'_l[H_l - 1]]$ to represent the corresponding retry vector.

Due to the existence of shared slots, generating the PDR table for τ_l (*i.e.*, determining w'_l and $\overrightarrow{R}'_l$) under TG-STM is much more complicated. Specifically, each shared slot contributed by τ_s is associated with a probability specifying the likelihood that this slot is not used by τ_s and thus can be used as a shared slot by τ_l. This possibility value of each shared slot impacts both w'_l & $\overrightarrow{R}'_l$, and the calculation of the achieved reliability of τ_l, denoted as λ'_l. We use $P_s(h_s, k)(0 \le h_s < H_s)$ to denote the probability for the k-th dedicated slot assigned to τ_s's h_s hop being used as a shared slot of τ_l. To determine w'_l and $\overrightarrow{R}'_l$ for τ_l, we need to tackle the following issues. i) How to calculate $P_s(h_s, k)$ for each shared slot and how to determine the number of shared slots that can be used by τ_l? ii) How to calculate the achieved reliability λ'_l with given w'_l and retry vector $\overrightarrow{R}'_l$? iii) How to determine the minimum needed w'_l and $\overrightarrow{R}'_l$ to have $\lambda'_l \ge \lambda^R$? Below, we present the answers to these questions.

According to the definition of $P_s(h_s, k)$, we have

$$P_s(h_s, k) = 1 - (1 - \lambda^L_{L_s[h_s]})^{k-1} \tag{5}$$

where $\lambda^L_{L_s[h_s]}$ represents the link PDR for delivering the h_s hop transmission of τ_s. According to Eq. (5), we know $P_s(h_s, 1) = 0$ which indicates that the first slots assigned to each hop of τ_s are always used to transmit τ_s's packets and only the retransmission slots allocated for each hop (*i.e.*, $k > 1$) can be used as shared slots by τ_l. Thus, the number of shared slots contributed by τ_s that can be used by τ_l equals to $w^+_s - H_s$.

According to Observation 2, all the shared slots from τ_s are scheduled before the dedicated slots (denoted as w^*_l) assigned to τ_l under TG-STM based on EDF[12]. That is, given w^*_l, the sequence of all the slots used by τ_l is determined and we have $w'_l = (w^+_s - H_s) + w^*_l$. Further, if a retry vector $\overrightarrow{R}'_l$ is also given, the slot assignment for each packet of τ_l can be determined, *i.e.* the slot assignment (including both the shared slots from τ_s and the dedicated slots assigned to τ_l) to each hop of τ_l. Below, we first present the calculation of the achieved reliability λ'_l with given w^*_l and $\overrightarrow{R}'_l$ by Eq. (6).

$$\lambda'_l = \prod_{h=0}^{H_l-1} \left[1 - \prod_{R'_l[0]}^{R'_l[H_l-1]} \left(1 - P_s(h_s, k) * \lambda^L_{L_l[h]} \right) \right]. \tag{6}$$

We now discuss how to determine w^*_l and $\overrightarrow{R}'_l$ according to Eq. (6) to satisfy τ_l's reliability requirement, *i.e.* $\lambda'_l \ge \lambda^R$. Given that the number of shared slots w^+_s is fixed and a minimum number of dedicated slots w^*_l is desired to satisfy $\lambda'_l \ge \lambda^R$, the high-level idea to determine w^*_l and $\overrightarrow{R}'_l$ is as follows. We first check whether λ^R can be satisfied only using $w^+_s - H_s$ shared slots by traversing all the possible slot assignments (*i.e.*, retry vectors). If not, an extra dedicated slot is assigned to τ_l (*i.e.*, $w^*_l = 1$) and the above process repeats until a feasible $\overrightarrow{R}'_l$ satisfying $\lambda'_l \ge \lambda^R$ is found. The searching space of such method can be significantly reduced based on the following theorem.

11. By formulating the task grouping problem as a maximum matching on a graph $G = (V, E)$, we can apply the Blossom algorithm [36] to solve this problem. However, our simulation result shows that the Blossom algorithm with a higher time complexity ($O(E \times V^2)$) demonstrates a similar performance with the greedy heuristic.

12. Observation 2 holds under any scheduling policy as long as τ_s and τ_l are harmonic tasks.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

Algorithm 6 PDR Table Computation under TBS

```
Input: $G = (V, E)$, $\tau_s$, $\tau_l$, $\lambda^R$, $w_s^+$, $P_s(h_s, k)$
Output: PDR table of $\tau_l$
1: Generate $w_l^+$ and the corresponding retry vector $\overrightarrow{R}_l = [R_l[0], R_l[1], \dots, R_l[H_l - 1]]$ using Alg. 1;
2: $w_l' \leftarrow w_l^+$;
   // $w_l^* = w_l^+ - (w_s^+ - H_s)$ if $w_s^+ - H_s < w_l^+$. Otherwise, $w_l^* = 0$
3: $\overrightarrow{R}_l' \leftarrow [R_l[0], R_l[1], \dots, R_l[H_l - 1]]$;
4: Calculate $\lambda_l'$ by Eq. (6);
5: while $\lambda_l' < \lambda^R$ do
6:     $w_l' \leftarrow w_l' + 1$;
       // Assign a shared slot if $w_l' < w_s^+ - H_s$. Otherwise assign an extra dedicated slot
7:     Select the hop index $h$ which yields the maximum PDR value (computed by Eq. (6));
8:     Update $\overrightarrow{R}_l'$ and $\lambda_l'$ in PDR table;
9: end while
```

Theorem 2 *Suppose under DTM, w_l^+ dedicated slots are assigned to τ_l using Alg. 1 and the corresponding achieved PDR value equals to λ_l. If TG-STM assigns the same number of slots to τ_l* (i.e. $w_l^+ = (w_s^+ - H_s) + w_l^*$)*, the achieved PDR value λ_l' is less than or equal to λ_l.*

The proof of Theorem 2 is straightforward. Since $\forall k > 1, P_s(h_s, k) < 1$, the theorem directly holds by comparing Eq. (1) and Eq. (6). According to Theorem 2, we must assign at least $\left\|w_l^+ - (w_s^+ - H_s)\right\|^0$ number of dedicated slots to τ_l in order to satisfy $\lambda_l' \geq \lambda^R$. That is,

$$w_l^* \geq \left\|w_l^+ - (w_s^+ - H_s)\right\|^0. \tag{7}$$

Similarly, given the retry vector $\overrightarrow{R}_l = [R_l[0], R_l[1], \dots, R_l[H_l - 1]]$ generated by Alg. 1 under DTM, TG-STM should assign at least $R_l[h]$ number of slots to each hop of τ_l to satisfy $\lambda_l' \geq \lambda^R$. That is,

$$\forall h, R_l'[h] \geq R_l[h]. \tag{8}$$

Based on Eq. (7) & Eq. (8), the searching space of PDR table generation for τ_l can be reduced since the numbers of possible values for both w_l^* and $R_l'[h]$ reduce. The pseudo code for PDR table computation under the TBS model is given in Alg. 6. We first compute the number of dedicated slots w_l^+ and the corresponding retry vector $\overrightarrow{R}_l$ for τ_l under DTM (Line 1). If $w_s^+ - H_s < w_l^+$, the number of shared slots can be used by τ_l is less than the number of dedicated slots needed by τ_l under DTM. Thus, we allocate $w_l^+ - (w_s^+ - H_s)$ dedicated slots to τ_l according to Eq. (7). Otherwise, we allocate the first w_l^+ shared slots to τ_l, where the number of shared slots allocated to each hop follows the specification in $\overrightarrow{R}_l$ (Line 3) and then calculate the achieved PDR value λ_l' (Line 4). As long as the required PDR value of τ_l is not satisfied (*i.e.*, $\lambda_l' < \lambda^R$), we assign an additional shared/dedicated slot to the h-th hop that yields the maximum PDR value and update $\overrightarrow{R}_l'$ and λ_l' in the PDR table (Lines 5-9).

6.3 TG-STM under PBS Model

TG-STM for the PBS model bares only one difference from that under the TBS model. That is, the PDR table computation for task τ_l uses shared slots contributed by τ_s in each harmonic task group HG_k. In the PBS model, since dedicated slots are assigned to individual packet of a task, we only need to determine the number of slots allocated to τ_l, *i.e.* w_l', under TG-STM but not the retry vector. In this case, as the number of shared slots from τ_s usable by τ_l is fixed and equals to $w_s^+ - H_s$, we only need to determine w_l^*, the number of dedicated slots needed by τ_l. Below, we discuss how to calculate the achieved PDR value $\lambda_l'(w)$ if w dedicated slots are assigned to τ_l. Then, we can derive w_l^* by gradually increasing w until $\lambda_l'(w) \geq \lambda^R$.

According to the PDR computation for task τ_i under DTM (discussed in Section 5), $\lambda_i(w) = Pr_i(H_i, w)$, and represents the probability of a packet of τ_i being delivered within w slots. Since TG-STM assigns certain number of shared slots from τ_s to τ_l, we need to first compute the probability of these shared slots being available to τ_l. Then we can calculate $\lambda_l'(w)$ based on $Pr_i(H_i, w)$. We use $P_s(k)$ to denote the probability that k shared slots are available to τ_l (*i.e.*, τ_s is successfully transmitted using $w_s^+ - k$ slots). According to the definition of $Pr_i(H_i, w)$, we have

$$P_s(k) = Pr_s(H_s, w_s^+ - k) - Pr_s(H_s, w_s^+ - k - 1) \tag{9}$$

Since τ_s needs at least H_s slots to be delivered, we calculate $P_s(k)$ for $0 \leq k \leq w_s^+ - H_s$. For a certain number of shared slots k and dedicated slots w_l^*, the probability of a packet being delivered within these $k + w_l^*$ slots equals to $P_s(k) \cdot Pr_l(H_l, k + w_l^*)$. Thus, with a number of dedicated slots w_l^*, the achieved PDR value, $\lambda_l'(w_l^*)$, is the sum of the above probabilities for all the cases of $k \in [0, w_s^+ - H_s]$. That is,

$$\lambda_l'(w_l^*) = \sum_{k=0}^{w_s^+ - H_s} P_s(k) \cdot Pr_l(H_l, k + w_l^*) \tag{10}$$

As the delivery of a packet of τ_l needs at least H_l slots, w_l^* is initialized to $\left\|H_l - (w_s^+ - H_s)\right\|^0$ and we then gradually increase w_l^* until $\lambda_l'(w) \geq \lambda^R$.

Below, we use an example to illustrate the process of TG-STM under both the TBS/PBS model.

Example 2 *Consider our RTWN testbed (in Fig. 5 in Section 7) as an example and the task specifications are given in Table 3. We first group all unicast tasks in the system into task groups according to Alg. 5, and we get $HG_1 = \{\tau_0, \tau_3\}$ and $HG^* = \{\tau_1, \tau_2\}$ (τ_4 and τ_5 are not unicast tasks). Thus, the retransmission slots of τ_0 can be used by τ_3 as shared slots.*

In the TBS model, τ_0 and τ_3 are assigned with 10 and 6 dedicated slots, respectively, under DTM (i.e. $w_0^+ = 10, w_3^+ = 6$)*, and the corresponding retry vectors are $\overrightarrow{R}_0 = [4,3,3]$, $\overrightarrow{R}_3 = [3,3]$ to satisfy $\lambda^R = 99\%$. According to Alg. 6, since the number of shared slots from τ_0 ($w_0^+ - H_0 = 10 - 3 = 7$) is larger than the dedicated slots required by τ_3 under DTM ($w_3^+ = 6$), TG-STM assigns the first 6 shared slots to τ_3 ($w_3' = 6$) with no additional dedicated slots,* i.e. $w_3^* = 0$. *The retry vector follows $\overrightarrow{R}_3$,* i.e. *$\overrightarrow{R}_3' = [3,3]$. We calculate the achieved PDR value $\lambda_3' = 96.41\%$ which is less than the reliability requirement $\lambda^R = 99\%$. Then, we need to assign an additional slot to τ_3 to increase its PDR value. As another available shared slot exists ($w_0^+ - H_0 = 7$), we assign this slot to τ_3. Among the two possible slot assignments $\overrightarrow{R}_3' = [4,3]$ and $\overrightarrow{R}_3' = [3,4]$, the former achieves a higher PDR value $\lambda_3' = 99.39\% > \lambda^R$. Therefore, no dedicated slots are*

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

needed by τ_3 using TG-STM under the TBS model to satisfy the reliability requirement while $w_3^+ = 6$ if DTM is applied.

In the PBS model, τ_0 and τ_3 are assigned with 7 and 4 dedicated slots, respectively, under DTM (i.e. *$w_0^+ = 7, w_3^+ = 4$). Similarly, TG-STM first assigns all available shared slots to τ_3 ($w_3' = w_0^+ - H_0 = 4$) with no additional dedicated slots,* i.e. *$w_3^* = 0$, and calculates the achieved PDR value $\lambda_3' = 91.80\%$ according to Eq. (10). Since $\lambda_3' < \lambda^R$, TG-STM assigns additional dedicated slots to τ_3 until $w_3^* = 2$ and $\lambda_3' = 99.49\%$. Therefore, only 2 dedicated slots are needed by τ_3 using TG-STM under the PBS model to satisfy the reliability requirement while $w_3^+ = 4$ if DTM is applied.*

More comprehensive experimental comparisons between DTM and TG-STM for the both TBS and PBS models are given in Section 8.

6.4 TG-STM in Dynamic Scheduling

Upon the detection of any disturbance, the corresponding tasks enter their rhythmic states and RD-PaS generates a temporary dynamic schedule to accommodate the increased rhythmic workload. Since shared slots are allocated to certain tasks under TG-STM, we need to accordingly modify the generation and update of task PDR table (storing the PDR function and retry vector), which involve shared slot allocation, in the dynamic scheduling process. As described in Sec. 6.2, PDR table generation consists of task grouping (Alg. 5) and PDR table computation (Alg. 6). Below we discuss the needed updates on PDR table computation for a task τ_0 entering its rhythmic state under different cases after performing task grouping.

Case 1: $\tau_0 \in HG^*$. In this case, all the time slots assigned to τ_0 are dedicated slots and none of these slots are used as shared slots by other tasks. Therefore, when τ_0 enters its rhythmic state and follows a new release pattern, the PDR table generation for all the other periodic tasks remains unchanged.

Case 2: $\tau_0 \in HG_k$. In this case, either τ_0 shares a number of slots from the other task τ_p in HG_k (τ_0 having a larger period), or a portion of τ_0's dedicated slots are shared by τ_p (τ_0 having a shorter period). For the former situation, since the rhythmic task is granted with highest criticality in the disturbed network, we assign dedicated slots to τ_0 without any shared slots for the sake of guaranteed reliability. For the latter situation, since the release pattern of rhythmic task τ_0 changes during its rhythmic state, τ_0 and τ_p become inharmonic tasks which violates Observation 2 for allocating shared slots among them. Therefore, we assign dedicated slots to both τ_0 and τ_p in HG_k in this case according to Alg. 1 (TBS model) and Alg. 4 (PBS model).

To satisfy the timing and reliability requirements of τ_0 after it enters the rhythmic state, we need to minimize the performance degradation of other periodic packets, *i.e.* solving the dynamic schedule generation problem **P2**. According to the high-level idea of Alg. 2, a basic number of slots are first allocated to each periodic packet to avoid significant reliability degradation. Then, extra slots are gradually assigned to each packet to maximize the system reliability. Below, we discuss the necessary modifications required by the above two steps under TG-STM.

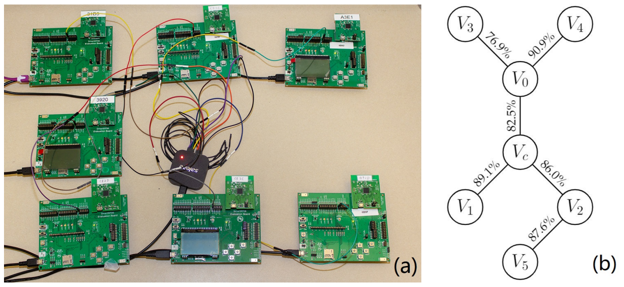


Fig. 5: (a): the RTWN testbed with 7 CC2538 evaluation boards; (b): the testing topology with emulated link PDR values.

In the first step, if all the slots allocated to task τ_i are dedicated slots in the system nominal mode (*i.e.*, either $\tau_i \in HG^*$, or τ_i has a shorter period in a task group HG_k), we assign H_i dedicated slots to τ_i to achieve the minimum non-zero reliability value. If τ_i is assigned with shared slots (*i.e.*, τ_i having larger period in a task group), we do not assign any dedicated slot to τ_i in the system rhythmic mode. Such modification is due to the following reason. If a task τ_l is assigned with shared slots, its PDR table can be generated only if the dedicated slot allocation for task τ_s within the same task group has been determined. Thus, the basic number of slots required by τ_l to satisfy its minimum reliability value cannot be determined before any shared slots are allocated to the harmonic task τ_s.

In the second step, extra dedicated slots are gradually allocated to periodic packets in a greedy manner to maximize the system reliability. Note that, the only difference introduced by TG-STM is the PDR value update in each iteration. Specifically, when a dedicated slot is assigned to a task with a shorter period in task group HG_k, this dedicated slot becomes a shared slot allocated to the other task with the larger period and thus its PDR value also needs to be updated.

7 Testbed Implementation and Validation

To validate the functionality of the proposed RD-PaS framework in real-life RTWNs, we implemented RD-PaS on a 7-node RTWN testbed (see Fig. 5) running the 6TiSCH protocol [21]. The testbed consists of seven CC2538 evaluation boards. One of these boards is configured as the controller node, while the others are configured as device nodes. A 16-channel 802.15.4 sniffer and an 8-channel logic analyzer are used to capture and analyze the activities of each device node. Our modified 6TiSCH stack utilizes 5KB more ROM and 2KB more RAM space for implementing RD-PaS (in TBS and PBS). These are relatively small compared to the original 6TiSCH stack which needs 69KB ROM and 6KB RAM. Due to the page limit, the implementation details of the RD-PaS framework is omitted. Below, we focus on discussing the functional validation of RD-PaS on the testbed.

The testbed topology is shown in Fig. 5(b). To attain the link PDRs as specified in the topology, we implemented a random packet dropper at the MAC layer of each device node. Six tasks are installed in the testbed and the task specifications are summarized in Table 3. The desired e2e PDR for all the tasks, λ^R, is set to 99%. τ_0, τ_1, τ_2 and τ_3 are unicast tasks, τ_5 is a broadcast task, and τ_4 is a task that

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

handles all network management packets. Since we always allocate two *shared* slots at the beginning of τ_4's period, we set $D_4 = 2$. For simplicity, only τ_0 enters the rhythmic state when a rhythmic event occurs.

7.1 Validation of reliable static scheduling

To validate the static schedule construction in RD-PaS, we run the specified task set on the testing topology in the nominal mode under both TBS and PBS models. The PDR tables computed by the testbed are exactly the same as those obtained from simulation. The PDR table for task τ_1 is given in Table 4 (while others are not shown due to the page limit). The highlighted rows indicate the corresponding w_i^+'s for TBS ($w_i^+ = 13$) and PBS ($w_i^+ = 7$) when λ^R is reached.

We further test 5000 packets for each unicast and broadcast task under both models, and compare the actual e2e PDR values collected from the testbed with the simulated values from Alg. 1 and Alg. 4. These results are summarized in Table 5. τ_4 is omitted in the table since it is a task dedicated for network management packets. It can be concluded from the table that the reliable static scheduling function in RD-PaS executes correctly as the actual e2e PDRs are improved to the desired values ($\geq 99\%$) in both models in the presence of specified packet loss. The slight differences between the measured and predicted e2e PDR values are expected due to the limited sample size.

7.2 Validation of reliable dynamic scheduling

To validate the functional correctness of reliable dynamic scheduling in RD-PaS on our testbed, we let the network trigger rhythmic events, and use the logic analyzer to capture the radio activities through a physical pin on each device node and plot the waveforms. We configure the network to enter the rhythmic mode at slot 720. The hyperperiod of the task set is 360 according to Table 3. (Rhythmic events can happen at any time. We chose this integer multiple of the hyperperiod to simplify the waveform demo.) Fig. 6 illustrates a sample waveform for 240 consecutive slots (slot 600-840) in the TBS model. (Both TBS and PBS models are validated. We present the results in the TBS model here for ease of explanation.) The network runs in the nominal mode for the first 120 time slots (Fig. 6b) and then switches to the rhythmic mode in the next 120 slots (Fig. 6c). Seven waveforms represent the radio activities, either transmitting, receiving, or listening, for all the 7 nodes, as labeled on the left side of the figures. Each rising and falling edge in the *Slot* row (lower part of the figures) mark the start of a new time slot. In the *schedule* row (lower part of the figures), slot assignments are indicated using different colors.

TABLE 3: Parameters of the task set deployed on the testbed.

Task	Routing Path	$P_i(D_i)$	$\overrightarrow{P}_i = \overrightarrow{D}_i$
τ_0	$V_3 \to V_0 \to V_c \to V_1$	30 (30)	$[20, 20, 20, 20, 20, 20]$
τ_1	$V_5 \to V_2 \to V_c \to V_0 \to V_4$	45 (45)	-
τ_2	$V_0 \to V_c \to V_1$	40 (40)	-
τ_3	$V_2 \to V_c \to V_1$	60 (60)	-
τ_4	-	60 (2)	-
τ_5	$V_c \to (V_0, V_1, V_2), V_0 \to (V_3, V_4), V_2 \to (V_5)$	120 (120)	-

TABLE 4: PDR table for task τ_1 in TBS and PBS models.

w	PDR Table in TBS		PDR Table in PBS
	$\lambda_i^*(w)$	$\overrightarrow{R}_i^*(w)$	$\lambda_i^*(w)$
4	0.564963	1,1,1,1	0.564963
5	0.663832	1,1,2,1	0.864394
6	0.756769	1,2,2,1	0.964613
7	0.850608	2,2,2,1	0.991720 ($\lambda_i^*(w_i^+)$)
8	0.928013	2,2,2,2	
9	0.952201	2,2,3,2	
10	0.968572	2,3,3,2	
11	0.981822	3,3,3,2	
12	0.989274	3,3,3,3	
13	0.993672 ($\lambda_i^*(w_i^+)$)	3,3,4,3	

TABLE 5: Reliable static schedule validation in TBS and PBS models on the testbed.

	Task	$\overrightarrow{R}_i^*$	$\lambda_i^*(w_i^+)$	Measured PDR
TBS Model	τ_0	[4,3,3]	99.01%	99.21%
	τ_1	[3,3,4,3]	99.37%	99.61%
	τ_2	[3,3]	99.34%	99.41%
	τ_3	[3,3]	99.60%	99.71%
	τ_5	[4,4,3]	99.38%	100%
	Task	w_i^+ or $\overrightarrow{R}_i^*$	$\lambda_i^*(w_i^+)$	Measured PDR
PBS Model	τ_0	7	99.68%	99.22%
	τ_1	7	99.17%	99.65%
	τ_2	5	99.80%	99.34%
	τ_3	4	99.29%	99.65%
	τ_5	[4,4,3]	99.38%	100%

From Fig. 6b, we observe that each task τ_i releases its packets according to P_i, and w_i^+ number of slots are allocated to each packet before its deadline (shown in the *schedule* row). In each scheduled slot, the sender attempts to transmit the packet and may succeed (marked by the arrows). Although some attempts fail, all the packets are still delivered to the destination node because of the right amount of retransmission slots as determined by the reliable static scheduling function. In Fig. 6c, τ_0 enters the rhythmic state, and its period is reduced according to $\overrightarrow{P}_0$ given in Table 3. Also as shown in the *schedule* row, the $W_{i,j}$ values for τ_0 do not change, while those for τ_1, τ_2, τ_3, τ_5 are reduced to $[9, 9, 9]$, $[4, 5, 5]$, $[4, 4]$, $[7]$, respectively. The $\overrightarrow{R}_{i,j}$ vectors are also selected correctly by the updated $W_{i,j}$ values in the rhythmic mode, and all the packets from the rhythmic task (τ_0) are successfully delivered to the destination. The captured results match the results from the simulation, and this validates the correctness of the reliable dynamic scheduling function in RD-PaS.

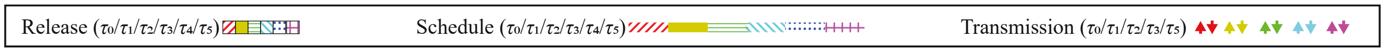
8 Simulation-based Evaluation

In this section, we evaluate the performance of RD-PaS through extensive simulations and compare RD-PaS with a state-of-the-art dynamic approach, D^2-PaS.[13] The first three sets of simulations compare the packet delivery ratio, network bandwidth usage and number of extra slots produced by RD-PaS with those by D^2-PaS. The 4th set of simulations

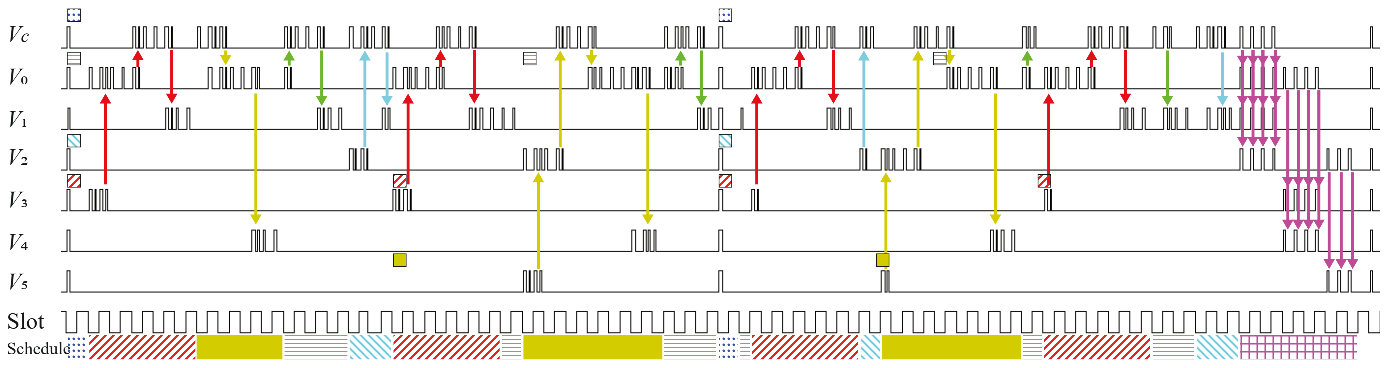
13. [17] shows that D^2-PaS has a clear advantage in packet dropping performance compared to the fully distributed scheduling framework FD-PaS, so we omit the comparison between RD-PaS and FD-PaS. Also, since we have proved the optimality of our retransmission slots assignment in Sec. 4.1, we omit to compare with the retransmission mechanism in [24] in the static setting.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

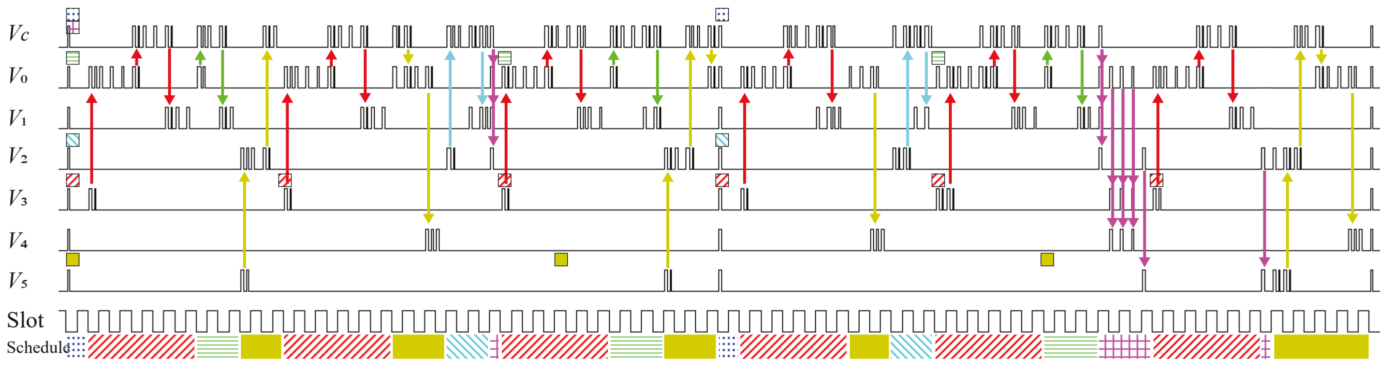
This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922


Release ($\tau_0/\tau_1/\tau_2/\tau_3/\tau_4/\tau_5$)
Schedule ($\tau_0/\tau_1/\tau_2/\tau_3/\tau_4/\tau_5$)
Transmission ($\tau_0/\tau_1/\tau_2/\tau_3/\tau_5$)

(a) Legend.



(b) Radio activities in slots 600 to 720 (nominal mode).



(c) Radio activities in slots 720 to 840 (rhythmic mode). Task τ_0 is in the rhythmic state and releases packets following $\overrightarrow{P}_0$ given in Table 3.

Fig. 6: Slot information and radio activities in the reliable dynamic scheduling test case captured by the logic analyzer.

studies the behavior of the rhythmic mode. We evaluate the reliability degradation by comparing RD-PaS with D^2-PaS on handling disturbances in RTWNs. In the last set of simulations, we evaluate the performance improvement of TG-STM in terms of network bandwidth utilization and number of extra slots.

8.1 Comparison of Packet Delivery Ratio

As RD-PaS utilizes retransmission slots to guarantee the required e2e PDR value for each task, there is no doubt that the system reliability will be improved compared with a traditional scheduling framework not considering reliability. To quantify such improvements, we calculate the e2e PDR resulted from applying D^2-PaS in lossy links with randomly generated link PDRs. Since the e2e PDR for each task is independent, we use different settings to randomly generate tasks and compute the PDR value for each task. The number of hops for a task, H, is drawn from the uniform distribution over $\{1, 2, ..., 10\}$ and the PDR value of each link on the routing path is randomly generated by controlling the average value of link PDR, λ^L, following a uniform distribution in $\{0.5, 0.55, ..., 0.95\}$. As periods and deadlines do not affect the packet delivery ratio, we only study PDR's dependency on H and λ^L. Fig. 7 shows the e2e PDR of a task as a function of λ^L and H. Because RD-PaS can always guarantee the required PDR value, its results are always at the ceiling (above 99%) of the figure and are thus omitted. From Fig. 7, we can observe the large gap between RD-PaS and D^2-PaS (60.6% on average) in guaranteeing the e2e PDR of the task.

8.2 Comparison of Network Bandwidth Usage

Allocating extra retransmission slots can significantly improve the reliability of packet delivery. However, higher network bandwidth is required which may affect system schedulability. In this set of experiments, we study the efficiency of using time slots to deliver packets, in different scheduling frameworks, according to the performance metric *throughput*. Throughput is defined as the number of packets delivered per slot (PPS) and is the ratio between the e2e PDR value and the number of allocated slots assigned to the task, *i.e.* $\frac{\lambda_i^*(w)}{w}$. The parameter settings of this set of experiments are the same as that in Section 8.1.

Fig. 8 summarizes throughputs for different scheduling frameworks with varied average link PDR λ^L and the number of hops, H, for the generated task. From the results, we can observe that D^2-PaS has a higher throughput when H is small and when λ^L is close to 1. However when the link PDR drops and H increases, RD-PaS (in both TBS and PBS models) gains better throughput. This is mainly due to the fact that using a time slot for retransmission can gain more throughput than transmitting a new packet in these cases. The simulation results also show that RD-PaS in the PBS model can always achieve a better throughput than in the TBS model. The reason is that the PBS model can always achieve same PDR with less number of slots, compared

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

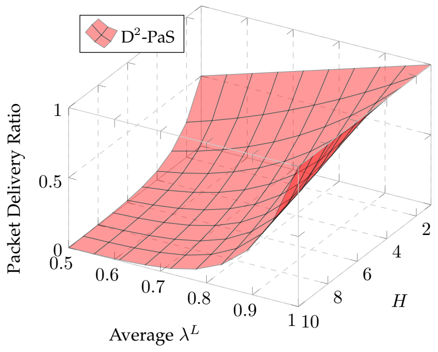


Fig. 7: PDR in D^2-PaS framework.

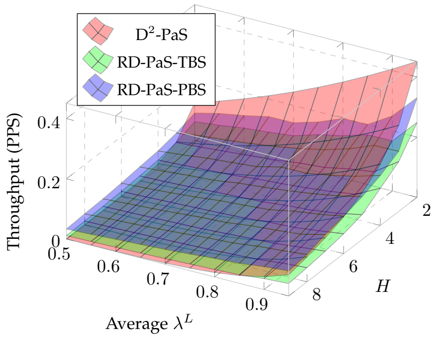


Fig. 8: Throughput comparison.

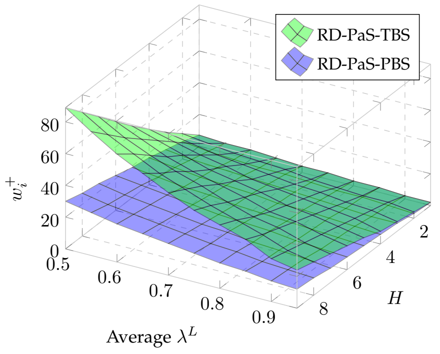


Fig. 9: Comparison of w_i^+ in TBS and PBS.

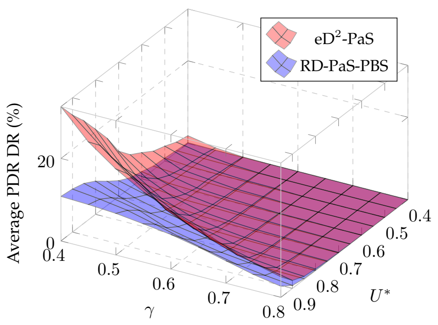


Fig. 10: Comparison of the PDR degradation rate.

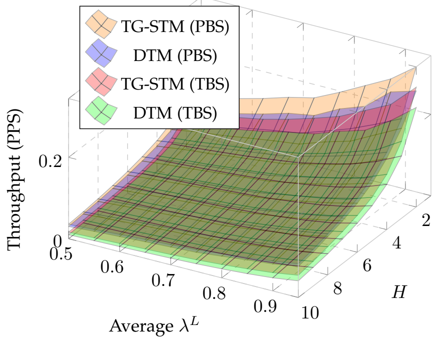


Fig. 11: Throughput improvement of TG-STM

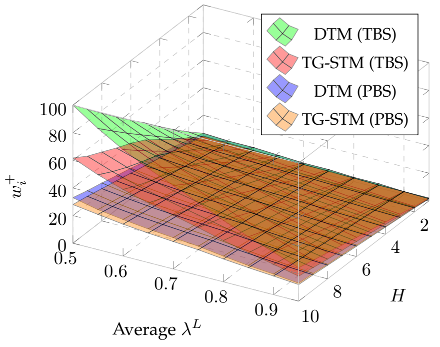


Fig. 12: w_i^+ improvement of TG-STM

to the TBS model due to the PBS's ability in sharing slots among transmissions of a packet.

8.3 Comparison of Required Numbers of Slots

In this set of experiments, we make further evaluation on RD-PaS in TBS and PBS models. As discussed in Section 5, the PBS model provides more flexibility on the retransmission slot assignment, and a less number of slots, w_i^+, is required to achieve the same λ^R as compared to the TBS model. Fig. 9 gives the comparison on the required number of slots under different settings of average λ^L and H, and the required end-to-end PDR value λ^R is set to 99%. As can be observed, tasks in PBS model require less number of slots than in TBS model, when $H > 1$. The required number of slots in the PBS model is 55.0% less on average compared to that in TBS model. This is consistent with the observation that one packet requires less number of slots to achieve the same λ^R in the PBS model.

8.4 Effectiveness in Handling Rhythmic Events

To evaluate the performance of RD-PaS in handling rhythmic events, we compare the *degradation rate (DR)* between RD-PaS and D^2-PaS. DR is defined as the ratio between the sum of reliability degradation (*i.e.*, $\delta_{i,j}$) from all periodic packets and the total number of generated periodic packets in the rhythmic mode. As D^2-PaS does not consider unreliable wireless links, we first extend D^2-PaS to support reliable transmission, denoted as eD^2-PaS. Specifically, all packets in eD^2-PaS are reliably transmitted using w_i^+ slots in the static schedule. In the dynamic schedule, transmission and retransmission slots assigned for each packet are not differentiated, *i.e.*, each packet can either be reliably scheduled or dropped.

To better control the system workload, we vary the nominal utilization of the task set. Specifically, we use a random periodic task set generated according to a target nominal utilization U^*. The generation of each random task τ_i is controlled by the following parameter settings: i) the number of hops H_i is drawn from the uniform distribution over $\{2, 3, ..., 16\}$, ii) the nominal period P_i is equal to deadline D_i and follows a uniform distribution in $\{50, 51, ...100\}$. As the simulation results in the last sub-section have shown, the PBS model requires less total number of slots to achieve the same transmission reliability. Thus, here we use the PBS model to generate the PDR function $\lambda_i^*(\cdot)$ for each task τ_i.

After a task set is generated, we randomly select two tasks to be the rhythmic tasks. To better control the workload of the rhythmic event, we assume that all the rhythmic periods (deadlines) are the same in $\overrightarrow{P}_i(\overrightarrow{D}_i)$ and the number of elements in $\overrightarrow{P}_i$ equals to 10. The value of each element $P_{i,R}$ is thus controlled by the rhythmic period ratio, $\gamma = \frac{P_{i,R}}{P_i}$.

Fig. 10 shows the results of DR as a function of both the nominal task set utilization U^* and the rhythmic period ratio γ. Each data point is the average value of $1,000$ trials. From Fig. 10, we can observe that RD-PaS has a lower PDR degradation rate (58.4% on average) over eD^2-PaS. The main reason is that eD^2-PaS either schedules or drops any packet $\chi_{i,j}$, *i.e.* $W_{i,j} \in \{0, w_i^+\}$. However, RD-PaS has more flexibility on tuning the number of slots assigned to $\chi_{i,j}$, *i.e.* $W_{i,j} \in \{0, H_i, \ldots, w_i^+\}$.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

8.5 Effectiveness of TG-STM

In this set of experiments, we evaluate performance improvements of TG-STM over DTM in terms of throughput and required number of slots w_i^+. The parameter settings remain the same as those in the previous sets of experiments. Fig. 11 and Fig. 12 summarize the throughput and w_i^+ results for both DTM and TG-STM, respectively. We can observe that all the simulation results of TG-STM under both the TBS and PBS models show a same trend with those of DTM. However, the performance of both throughput and required number of slots improve by varying degrees. Specifically, the required number of slots in the TBS model is significantly reduced using TG-STM while the improvement in the PBS model is smaller. This is due to the ability of the PBS model in sharing slots among transmissions of each packet such that a number of slots have already been saved, hence fewer opportunities for TG-STM.

9 Conclusion and Future Work

In this paper, we present RD-PaS, a reliable dynamic packet scheduling framework for RTWNs. RD-PaS provides guaranteed reliability of packet delivery in RTWNs for both transmission-based scheduling model and packet-based scheduling model in a hybrid fasion. In the presence of unexpected disturbances, RD-PaS makes dynamic schedule adjustment judiciously to guarantee timely and reliable delivery of the critical rhythmic packets while minimizes reliability degradation for noncritical packets. An optimal algorithm (for the static case) as well as a heuristic (for the dynamic case) are introduced for realizing RD-PaS. Extensive testbed and simulation based experiments are conducted to validate the correctness and effectiveness of RD-PaS. Our experimental results show that RD-PaS can significantly improve the QoS (in terms of reliability) compared with the state-of-the-art approaches. As future work, we will extend RD-PaS to further support RTWNs with multi-channel scheduling and multi-path routing capabilities, and evaluate its performance in large-scale RTWN testbeds.

References

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, Nov 2018.

[2] V. C. Gungor, G. P. Hancke *et al.*, "Industrial wireless sensor networks: Challenges, design principles, and technical approaches," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 4258–4265, Oct 2009.

[3] T. Gong, T. Zhang, X. S. Hu, Q. Deng, M. Lemmon, and S. Han, "Reliable dynamic packet scheduling over lossy real-time wireless networks," in *31st Euromicro Conference on Real-Time Systems (ECRTS)*, 2019.

[4] 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, 2014.

[5] A. Saifullah, D. Gunatilaka, P. Tiwari, M. Sha, C. Lu, B. Li, C. Wu, and Y. Chen, "Schedulability analysis under graph routing in WirelessHART networks," in *2015 IEEE Real-Time Systems Symposium*, Dec 2015, pp. 165–174.

[6] M. Zimmerling, L. Mottola, P. Kumar, F. Ferrari, and L. Thiele, "Adaptive real-time communication for wireless cyber-physical systems," *ACM Transactions on Cyber-Physical Systems*, vol. 1, no. 2, pp. 1–29, 2017.

[7] B. Li, L. Nie, C. Wu, H. Gonzalez, and C. Lu, "Incorporating emergency alarms in reliable wireless process control," in *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, 2015, pp. 218–227.

[8] F. Terraneo, P. Polidori, A. Leva, and W. Fornaciari, "TDMH-MAC: Real-time and multi-hop in the same wireless mac," in *2018 IEEE Real-Time Systems Symposium (RTSS)*, Dec 2018, pp. 277–287.

[9] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled tsch," in *Proceedings of the 13th ACM conference on embedded networked sensor systems*, 2015, pp. 337–350.

[10] S. Kim, H.-S. Kim, and C. Kim, "Alice: Autonomous link-based cell scheduling for tsch," in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, 2019, pp. 121–132.

[11] S. Oh, D. Hwang, K.-H. Kim, and K. Kim, "Escalator: An autonomous scheduling scheme for convergecast in tsch," *Sensors*, vol. 18, no. 4, p. 1209, 2018.

[12] S. Jeong, H.-S. Kim, J. Paek, and S. Bahk, "Ost: On-demand tsch scheduling with traffic-awareness," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 69–78.

[13] J. Shi, M. Sha, and Z. Yang, "Distributed graph routing and scheduling for industrial wireless sensor-actuator networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1669–1682, 2019.

[14] S. Hong, X. S. Hu, T. Gong, and S. Han, "On-line data link layer scheduling in wireless networked control systems," in *2015 27th Euromicro Conference on Real-Time Systems*, July 2015, pp. 57–66.

[15] T. Zhang, T. Gong, S. Han, Q. Deng, and X. S. Hu, "Distributed dynamic packet scheduling framework for handling disturbances in real-time wireless networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 11, pp. 2502–2517, 2018.

[16] ——, "Distributed dynamic packet scheduling framework for handling disturbances in real-time wireless networks," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2018.

[17] T. Zhang, T. Gong, Z. Yun, S. Han, Q. Deng, and X. S. Hu, "FD-PaS: A fully distributed packet scheduling framework for handling disturbances in real-time wireless networks," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2018, pp. 1–12.

[18] T. Zhang, T. Gong, S. Han, Q. Deng, and X. S. Hu, "Fully distributed packet scheduling framework for handling disturbances in lossy real-time wireless networks," *IEEE Transactions on Mobile Computing*, 2019.

[19] 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 *2008 IEEE Real-Time and Embedded Technology and Applications Symposium*, April 2008, pp. 377–386.

[20] ISA Standard, "Wireless systems for industrial automation: process control and related applications," *ISA-100.11 a-2009*, 2009.

[21] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert, "6TiSCH: deterministic ip-enabled industrial internet (of things)," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 36–41, December 2014.

[22] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon, "Reliable and real-time communication in industrial wireless mesh networks," in *2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*, April 2011, pp. 3–12.

[23] Y. Chen, H. Zhang, N. Fisher, L. Y. Wang, and G. Yin, "Probabilistic per-packet real-time guarantees for wireless networked sensing and control," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2133–2145, May 2018.

[24] R. Brummet, D. Gunatilaka, D. Vyas, O. Chipara, and C. Lu, "A flexible retransmission policy for industrial wireless sensor actuator networks," in *2018 IEEE International Conference on Industrial Internet (ICII)*, Oct 2018, pp. 79–88.

[25] I. . W. Group *et al.*, "Ieee standard for local and metropolitan area networks—part 15.4: Low-rate wireless personal area networks (lr-wpans)," *IEEE Std*, vol. 802, pp. 4–2011, 2011.

[26] R. Tavakoli, M. Nabi, T. Basten, and K. Goossens, "Hybrid timeslot design for ieee 802.15. 4 tsch to support heterogeneous wsns," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2018, pp. 1–7.

[27] A. Elsts, X. Fafoutis, J. Pope, G. Oikonomou, R. Piechocki, and I. Craddock, "Scheduling high-rate unpredictable traffic in ieee 802.15. 4 tsch networks," in *2017 13th International Conference on*

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3196922

Distributed Computing in Sensor Systems (DCOSS). IEEE, 2017, pp. 3–10.

[28] D. Yang, Y. Xu, H. Wang, T. Zheng, H. Zhang, H. Zhang, and M. Gidlund, "Assignment of segmented slots enabling reliable real-time transmission in industrial wireless sensor networks," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 3966–3977, 2015.

[29] R. Tavakoli, M. Nabi, T. Basten, and K. Goossens, "Topology management and tsch scheduling for low-latency convergecast in in-vehicle wsns," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1082–1093, 2019.

[30] M. Hashimoto, N. Wakamiya, M. Murata, Y. Kawamoto, and K. Fukui, "End-to-end reliability-and delay-aware scheduling with slot sharing for wireless sensor networks," in *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*. IEEE, 2016, pp. 1–8.

[31] R. Brummet, O. Chipara, and T. Herman, "Recorp: Receiver-oriented policies for industrial wireless networks," in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2020, pp. 135–141.

[32] J. Kim, K. Lakshmanan, and R. R. Rajkumar, "Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems," in *2012 IEEE/ACM Third International Conference on Cyber-Physical Systems*, April 2012, pp. 55–64.

[33] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, 1973.

[34] M. Mohaqeqi, M. Nasri, Y. Xu, A. Cervin, and K.-E. Årzén, "On the problem of finding optimal harmonic periods," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, 2016, pp. 171–180.

[35] M. Nasri and G. Fohler, "An efficient method for assigning harmonic periods to hard real-time tasks with period ranges," in *2015 27th Euromicro Conference on Real-Time Systems*. IEEE, 2015, pp. 149–159.

[36] V. Kolmogorov, "Blossom v: a new implementation of a minimum cost perfect matching algorithm," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 43–67, 2009.

Tianyu Zhang received the MS and PhD degrees both from Northeastern Univerity, China, in 2013 and 2018, respectively. He is currently a postdoc fellow at the Department of Computing, The Hong Kong Polytechnic University. His research interests include real-time systems, cyber-physical systems and wireless sensor-actuator networks.

Tao Gong received the BS degree from Beihang University, China, in 2013 and the Ph.D. degree from the University of Connecticut in 2020. His research interests include real-time and embedded systems, industrial wireless networks, and distributed real-time data analytics.

Mingsong Lyu received the Ph.D. degree in computer science from Northeastern University, Shenyang, China. He is currently a research assistant professor at the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His research interests include timing analysis of real-time systems and intermittent computing.

Nan Guan is currently an associate professor at the Department of Computing, The Hong Kong Polytechnic University. He received the BS and MS degrees from Northeastern University, China and the PhD degree from Uppsala University, Sweden in 2013. His research interests include real-time embedded systems and cyber-physical systems. He received SIGBED Early Career Researcher Award in 2020.

Song Han received the BS degree from Nanjing University in 2003, the M.Phil. degree from the City University of Hong Kong in 2006, and the Ph.D. degree from the University of Texas at Austin in 2012, all in Computer Science. He is currently an associate professor in the Department of Computer Science and Engineering at the University of Connecticut. His research interests include cyber-physical systems, real-time and embedded systems, and wireless networks.

Xiaobo Sharon Hu received the BS degree from Tianjin University, China, the MS degree from the Polytechnic Institute of New York, and the PhD from Purdue University. She is a professor in the Dept. of Computer Science and Engineering at the University of Notre Dame. Her research interests include real-time embedded systems, low-power system design, and computing with emerging technologies. She has authored more than 250 papers in related areas.

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: UNIVERSITY OF CONNECTICUT. Downloaded on September 28,2022 at 15:19:17 UTC from IEEE Xplore. Restrictions apply.