2022 IEEE 28th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA) | 978-1-6654-5344-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/RTCSA55878.2022.00014

Distributed Successive Packet Scheduling for Multi-Channel Real-Time Wireless Networks

Dawei Shen[†*], Tianyu Zhang[†], Jiachen Wang[‡], Qingxu Deng[†], Song Han[‡], Xiaobo Sharon Hu[§]
[†]School of Computer Science and Engineering, Northeastern University, Shenyang, China
[†]Email: shendw@stumail.neu.edu.cn, tyzhang4@gmail.com, dengqx@mail.neu.edu.cn.
[‡]Dept. of Computer Science and Engineering, University of Connecticut, Storrs, CT, 06269
[‡]Email: {jc.wang, song.han}@uconn.edu
[§]Dept. of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, 46556
[§]Email: shu@nd.edu

***Abstract*—With the rapid growth of industrial Internet of Things (IIoT) applications, real-time wireless networks (RTWNs) are playing an increasingly important role in providing real-time, reliable, and secure communication services for these applications. A key challenge in RTWN management is to ensure real-time Quality of Services (QoS), especially in the presence of unexpected external (*i.e.*, application-side) and internal (*i.e.*, network-side) disturbances. This paper presents a novel framework, DS-PaS, to determine the packet transmission schedule for multi-channel multi-hop RTWNs at the data link layer in a distributed and dynamic fashion. DS-PaS is able to (i) handle external disturbances, (ii) support spatial reuse, (iii) meet deadlines of all critical tasks, and (iv) minimize the number of dropped non-critical packets. To avoid transmission collisions when using inconsistent information in a distributed framework, DS-PaS incorporates several key advances in both the data-link layer protocol and algorithm design so that individual nodes can build on-line schedules with only local interference information. Extensive evaluation based on both testbed implementation and simulation validates the correctness of the DS-PaS design and demonstrates its effectiveness compared to the state of the art.**

***Index Terms*—Real-time wireless networks, distributed successive packet scheduling, multi-channel, spatial reuse, disturbance**

I. Introduction

In recent years, we have witnessed growing development and deployment of real-time wireless networks (RTWNs) in various industrial applications [1], especially those with stringent requirements on real-time and reliable information delivery. Packet scheduling at data link layer plays a key role in meeting these requirements of RTWNs. Besides meeting these timing and reliability requirements, several other challenges must be dealt with when developing packet scheduling approaches. First, RTWNs often employ multiple channels along with spatial reuse in order to increase network capacity. Second, RTWNs often suffer from dynamic workload perturbations caused by unexpected disturbances, which are particularly severe in harsh industrial environment.

Unexpected disturbances in general can be classified into *internal* disturbances and *external* disturbances. *Internal* disturbances are casued by network changes (due to node or link failure, etc.). *External* disturbances are caused by unexpected changes in the environment being monitored and controlled (due to detection of an emergency, sudden pressure or temperature increase, etc.). Many approaches (*e.g.*, [2]–[4]) have been proposed to handle internal disturbances in RTWNs. Few studies address external disturbances in RTWNs. The focus of this paper is to develop an efficient and scalable packet scheduling solution for the data link layer to handle unexpected *external* disturbances (referred to as disturbances for brevity) in multi-channel RTWNs with spatial reuse.

Some researchers propose centralized methods to handle disturbances (*e.g.*, [5]–[7]). For example, OLS [7] relies on a centralized controller (*e.g.*, gateway) to generate a dynamic schedule for handling disturbance and broadcast the entire updated schedule to all the nodes in the network. To reduce the communication overhead for broadcasting the dynamic schedule, D^2-PaS [5] and RD-PaS [6] let the gateway propagate only a portion of schedule-related information and each node generates the dynamic schedule locally. However, these centralized approaches are subjected to single-point (*e.g.*, the gateway) failure, and do not scale well for large networks.

Some distributed methods (*e.g.*, [8]–[11]) have been proposed for RTWNs to handle external disturbances. In Orchestra [8], [10], DiGS [11], and FD-PaS [9], only single-channel protocols are considered. To the best of our knowledge, FD-PaS is the state-of-the-art approach to handling external disturbance for single-channel networks with no spatial reuse. The main idea of FD-PaS is to generate a dynamic schedule at each device node locally according to the task set information when disturbances happen. Directly extending FD-PaS to multi-channel RTWNs with spatial reuse, however, can lead to inconsistent schedules (to be explained in details in Sec. III) that may cause critical tasks to miss their deadlines. DistributedHART [12] is the most recent distributed approach for multi-channel scheduling in RTWNs supporting spatial reuse. Though DistributedHART can handle some workload changes (*e.g.*, a node can adjust its own schedule when the period of a packet at the node is changed but a feasible schedule can be found for all the packets to be handled), it cannot handle external disturbances that result in the network being overloaded. That is, DistributedHART can only work under the assumption that the network workload is always schedulable. However, to avoid costly over-design, typically RTWNs are allowed to be temporarily overloaded when dis-

[*]The first two authors have equal contribution to this work.

2325-1301/22/$31.00 ©2022 IEEE
DOI 10.1109/RTCSA55878.2022.00014

turbances happen. Therefore, handling external disturbances in multi-channel with spatial reuse in a distributed and dynamic fashion is valuable to explore.

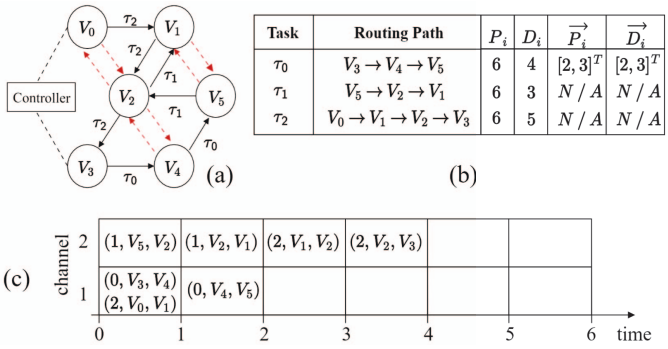
In this paper, we introduce a novel data-link layer packet scheduling framework called DS-PaS. To our best knowledge, DS-PaS is the first work that is able to (i) handle external disturbances, (ii) support multiple channels and spatial reuse, and (iii) guarantee to meet the deadlines of all critical tasks while dropping the minimum number of packets. To address the scalability issue and unexpected disturbances, DS-PaS lets individual nodes construct a *dynamic* schedule in a *distributed* way. To ease the memory demand for storing the complete interference information of a large RTWN, DS-PaS only requires each node to store limited local interference information. These setups together with multi-channel and spatial reuse introduce a number of unique challenges in the design of DS-PaS.

We make the following contributions to tackle the challenges. (i) We devise a successive scheduling mechanism in which a scheduling decision made by a node is propagated to the subsequent nodes along the routing path. This mechanism ensures that the local schedules generated by different nodes on the same routing path is always collision-free. (ii) To satisfy the timing requirements of critical tasks while dropping fewest non-critical packets when handling the extra workload caused by disturbances, we introduce an efficient heuristic to be executed by individual nodes locally to determine a temporary dynamic schedule. To validate the correctness of the DS-PaS design, we have implemented the DS-PaS framework on a 7-node RTWN testbed. We also developed a high-fidelity simulation tool to evaluate DS-PaS's effectiveness. Our extensive experimental studies show that compared to existing work, DS-PaS not only improves system scalability in terms of slower degradation in the number of feasible task sets and the packet drop ratio as the network size grows, but also provides fast and effective responses to external disturbances.

II. System Model

This paper considers a typical RTWN system, where multiple sensors, actuators and relay nodes (all referred to as device nodes) and a controller (referred to as the gateway) are wirelessly connected to form a multi-channel multi-hop RTWN. The network is modeled as a directed graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \{V_0, V_1, \dots\}$ is the device node set and $(V_i, V_j) \in \mathbf{E}$ represents a reliable link from V_i to V_j. Fig. 1(a) shows an example RTWN with 7 device nodes and a controller.

We use the concept of tasks to describe packet transmissions from sensor nodes to actuator nodes. Specifically, the system runs a fixed set of unicast tasks $\mathcal{T} = \{\tau_0, \tau_1, \dots, \tau_N\}$. Each task τ_i $(0 \le i \le N)$ follows a designated routing path with H_i hops. It periodically generates a packet which originates at a sensor node, passes through some relay nodes then reaches to an actuator. Each task τ_i consists of an infinite sequence of packets. The k-th packet of τ_i, $\chi_{i,k}$, is associated with release time $r_{i,k}$, deadline $d_{i,k}$ and finish time $f_{i,k}$. The delivery of



Task	Routing Path	P_i	D_i	$\overrightarrow{P_i}$	$\overrightarrow{D_i}$
τ_0	$V_3 \to V_4 \to V_5$	6	4	$[2,3]^T$	$[2,3]^T$
τ_1	$V_5 \to V_2 \to V_1$	6	3	N/A	N/A
τ_2	$V_0 \to V_1 \to V_2 \to V_3$	6	5	N/A	N/A

Fig. 1. (a) An example RTWN with 3 tasks running on 7 nodes. A black solid arrow indicates a reliable link from the source node to the destination node; a black dashed line indicates a reliable link between the controller and the device node; a red dashed arrow indicates that the destination node is within the interference range of the source node. (b) Parameters for the three tasks in Fig. 1(a). (c) Static schedule S (generated by the controller at the initialization) with a length of 6 time slots and 2 channels. A tuple (i, V_j, V_k) represents a transmission of task τ_i with sender V_j and receiver V_k.

packet $\chi_{i,k}$ at the h-th hop is referred to as a transmission denoted as $\chi_{i,k}(h)$ $(1 \le h \le H_i)$.

Our system model adopts a multi-channel Time Division Multiple Access (TDMA)-based data link layer with C channels. We assume that each node is capable of transmitting or receiving data on any one channel at a given time slot. An acknowledgement packet (ACK) is sent upon a successful transmission reception in the same time slot. To improve network capacity, spatial reuse is enabled such that concurrent transmissions involving different node pairs can take place within the same time slot on the same channel. Since the wireless medium is essentially shared among nodes, there is a potential that one transmission may affect another when both transmissions share the same time slot. We use the following terms to describe two situations that prevent two transmissions in the same time slot to be reliably received.

Definition 1 (Transmission conflict)**.** *Two transmissions A and B assigned to the same time slot experience transmission conflict if the senders/receivers of A/B share a same node.*

Definition 2 (Transmission interference)**.** *Two transmissions A and B assigned to the same time slot and on the same channel experience transmission interference if A's (or B's) signal interferes with B's (or A's) signal at B's (or A's) receiver.*

Transmission interference is caused by interference among nodes in RTWNs (as indicated by the red dashed arrows in Fig. 1). A set of transmissions on the same channel have no interference if all receivers are not in the other senders' interference range. The interference range is determined by the transmission power and distance. To capture the potential interference of each node with other nodes in the network, a local interference table can be built and stored at each node using existing methods (e.g., [13]) according to the task set information. Combining the local interference tables of all the nodes gives the global interference table which contains the

TABLE I
THRESHOLD VALUES ON SENSED DATA AND RESPONSE PLANS FOR AN UNDERGROUND MINE USE CASE.

	Threshold values and response plans					
	Safe (green)		Transient (orange)		Unsafe (red)	
Sampling period (s)	30		15		5	
Temperature (°C)	≤ 28	1. Next reading	(28, 40)	1. Turn the auxiliary fan(s) on	≥ 40	1. Text a message to all to evacuate from unsafe place(s)
Humidity (%)	≤ 75		(75, 85)	2. Text a message to the shift supervisors	≥ 85	2. Next reading
CO_2 (ppm)	≤ 2000		(2000, 5000)	3. Next reading	≥ 5000	

entire network's interference information. Suppose an RTWN consists of M nodes and it requires b bytes to store the index of a node. Then the global interference table requires $(b \cdot M)^2$ bytes in the worst case while a local interference table only requires $b \cdot M$ bytes. Given that M can be a large number and a typical RTWN node is memory constrained (*e.g.*, 32KB RAM for TI CC2538), it is more practical to only store local interference tables at individual device nodes, which we adopt in this paper.

Our task model explicitly considers external disturbances. Table I gives an example use case of an underground mine monitoring system [14] to illustrate the actions required by RTWNs upon detecting such disturbances. In the RTWN, hundreds of sensor nodes are mounted in working areas to sense environment attributes (*e.g.*, temperature, humidity and gas concentration), and the working environment is assessed in three conditions including safe (green), transient (orange) and unsafe (red) according to the sensor reading values. From Table I, we can observe that (i) upon detecting a disturbance, RTWN applications would require more frequent sampling and control actions for the disturbance affected task(s), and (ii) the task(s) in transient or unsafe conditions are granted higher criticality since they need to deliver emergency information.

To capture the abrupt increase in network resource demand caused by disturbance, we adopt the rhythmic task model [15]. Since the workload change of a disturbed task in RTWNs generally takes the form of sampling rate variation, we let each unicast task τ_i to have two states with different period and deadline patterns. In the *nominal state*, τ_i follows the given nominal period P_i and nominal relative deadline D_i ($D_i \leq P_i$). When a disturbance occurs, τ_i enters the *rhythmic state* in which its period and relative deadline are first reduced in order to respond to the disturbance, and then gradually return to their nominal values by following some monotonically non-decreasing pattern. We use vectors $\overrightarrow{P_i} = [P_{i,x}, x = 1, \ldots, R]^T$ and $\overrightarrow{D_i} = [D_{i,x}, x = 1, \ldots, R]^T$ to represent the periods and relative deadlines of τ_i when it is in the rhythmic state. As soon as τ_i enters the rhythmic state, its period and relative deadline adopt sequentially the values specified by $\overrightarrow{P_i}$ and $\overrightarrow{D_i}$, respectively. τ_i returns to the nominal state when it starts using P_i and D_i again. Note that our DS-PaS framework is not limited to the rhythmic task model and is applicable to any task model that provides workload changing patterns for handling disturbances.

In this work, we assume that at most one task[1] can be in the rhythmic state at any time during the network operation. To simplify terminology, we refer to any task currently in the rhythmic state as *rhythmic task* and denote it as τ_0, and refer to task τ_i $(1 \leq i \leq N)$ currently not in the rhythmic state as *periodic task*. When task τ_0 enters the rhythmic state, we refer to the nodes on τ_0's routing path as rhythmic nodes. The set of rhythmic nodes is denoted by $\mathbf{V}_{rhy}$. When τ_0 enters the rhythmic state, we say that the system switches to the *rhythmic mode*. The system returns to the *nominal mode* when the disturbance has been completely handled, typically some time after τ_0 returns to the nominal state. As observation (ii) from the use case, disturbances may have catastrophic effects, and the rhythmic task is granted higher criticality in the rhythmic mode. Therefore, when the system enters the rhythmic mode, the deadlines of the rhythmic task are considered to be hard while periodic tasks can tolerate occasional deadline misses. Without loss of generality, we assume that τ_0 enters the rhythmic state at time slot $r_{0,m+1}$ (denoted as $t_{n\to r}$) and returns to the nominal state at $r_{0,m+R+1}$ (denoted as $t_{r\to n}$). Thus, τ_0 stays in the rhythmic state during $[t_{n\to r}, t_{r\to n})$. The system returns to the nominal mode at a time referred to as end point t_{ep} when the disturbance has been completely handled. Any packet of τ_0 released in the system rhythmic mode is referred to as a *rhythmic packet* while the packets of task τ_i $(1 \leq i \leq N)$ are *periodic packets*.

At a high level, the disturbance handling problem we aim to solve can be described as follows. Consider a multi-channel TDMA-based RTWN supporting spatial reuse. Each node stores its local interference table and follows a pre-determined static schedule to transmit periodic packets in the nominal mode. When there is no disturbance, the system is schedulable. Upon detecting a disturbance at t^* (some release time[2] of the rhythmic task τ_0), the system needs a *fast*, *distributed* and *effective* dynamic packet scheduling strategy to handle the disturbance. As a *fast* strategy, the system should start handling the disturbance within a short time (*e.g.*, one nominal period of the rhythmic task). As a *distributed* strategy, nodes must make local decisions to handle the disturbance without the aid of any centralized controller. As an *effective* strategy, deadlines of all rhythmic packets should be met while a minimum number of periodic packets should be dropped.

[1]Handling multiple concurrent rhythmic tasks is left for future work.

[2]We assume that disturbances can be detected only at the time when the sensor samples the environment data, *i.e.*, the release time of a certain packet.

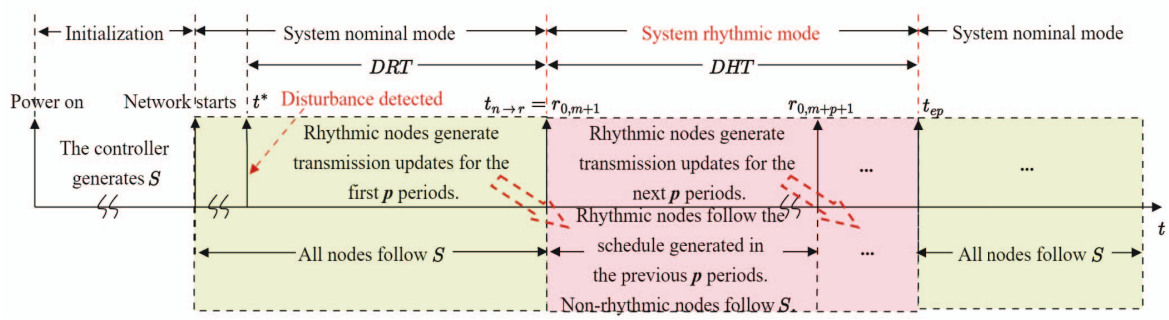


Fig. 2. An overview of the execution model of DS-PaS. DRT and DHT denote disturbance response time and disturbance handling time, respectively.

III. Overall Framework of DS-PaS

In this section, we present an overview of the proposed distributed successive packet scheduling framework, DS-PaS. One key challenge to solve the disturbance handling problem is how to generate consistent transmissions among all rhythmic nodes when a disturbance is detected. We first give the following schedule-related definitions below.

Definition 3 (Consistent Transmission)**.** *If the sender and the receiver of a transmission assign the same slot offset t and channel offset c to the transmission in their own local schedules, this transmission is a consistent transmission.*

Definition 4 (Static/Dynamic Schedule)**.** *A static/dynamic schedule $S/\tilde{S}$ is a set of slot assignment to be used in the system nominal/rhythmic mode. Each slot is either idle or allocated to one or multiple transmissions.*

The main reason that FD-PaS [9] cannot handle disturbance in multi-channel RTWNs with spatial reuse is that each node in $\mathbf{V}_{rhy}$ determines the dynamic schedule *independently*. Thus, the dynamic schedule generated by each node is not known by other nodes. Such an approach works if all nodes maintain the same information to generate the dynamic schedule. However, as discussed in the system model, it is more realistic for each node to only store a local interference table in which the interference information may differ among different nodes. To eliminate the inconsistency when nodes in $\mathbf{V}_{rhy}$ handle disturbance, we propose a distributed *successive* packet scheduling framework, referred to as DS-PaS.

The key idea of DS-PaS is to let each node in $\mathbf{V}_{rhy}$ only determine the slot and channel assignments for the rhythmic transmissions *sent by itself* in $\tilde{S}$, and we refer to such assignment for each transmission as a *transmission update*. Then, each node propagates the information related to these transmission updates to its subsequent node (receiver of these updated transmissions) along the rhythmic task's routing path. In this way, nodes in $\mathbf{V}_{rhy}$ are able to generate dynamic schedules in a successive fashion without inconsistency. We use $\tilde{s}_{0,k}(h)$ $(1 \le h \le H_0)$ to denote the transmission update for $\chi_{0,k}(h)$ generated by its sender V_h. By successively (*i.e.*, in the order of the sending and receiving nodes along the rhythmic task's routing path) generating and propagating the transmission updates $\tilde{s}_{0,k}(h)$, DS-PaS can avoid inconsistent dynamic schedule[3]. The execution model of DS-PaS is summarized below:

- In the system initialization phase (un-shaded part in Fig. 2), the controller generates a static schedule S (*e.g.*, Fig. 1(c)) using the global interference table[4] according to a certain scheduling policy (we use the earliest deadline first (EDF) scheduling policy in this paper). S contains the schedule of all the packets within one hyper-period and is propagated to all the nodes in the network. After receiving S, each node stores it locally and follows S (repeated every hyper-period) to transmit packets when no disturbance occurs.
- When a disturbance is detected at t^* (some release time of the rhythmic task τ_0), each sender node $V_h \in \mathbf{V}_{rhy}$ $(1 \le h \le H_0)$ along τ_0's routing path locally and successively determines the transmission updates in the dynamic schedule $\tilde{S}$ for the rhythmic transmissions sent by itself in the first p periods[5], denoted as $\tilde{s}_{0,m+1}(h), \ldots, \tilde{s}_{0,m+p}(h)$ $(1 \le p \le R)$, after the system enters the rhythmic mode at $t_{n\to r}$. This phase is shown in the green shaded region in Fig. 2. After the transmission updates are determined, V_h propagates the key information of $\tilde{s}_{0,m+1}(h), \ldots, \tilde{s}_{0,m+p}(h)$ to the next node V_{h+1} along τ_0's routing path. Note that all nodes still follow the static schedule S to transmit packets during the time from the disturbance is detected (*i.e.*, t^*) to the time that τ_0 enters its rhythmic state (*i.e.*, $r_{0,m+1}$).
- After the information of transmission updates $\{\tilde{s}_{0,k}(h)|(m+1 \le k \le m+p)\}$ is received by all receiver nodes $V_{h+1}(1 \le h \le H_0)$, τ_0 enters its rhythmic state at $r_{0,m+1}$. During $[r_{0,m+1}, r_{0,m+p+1})$, each node $V_h \in \mathbf{V}_{rhy}$ follows $\{\tilde{s}_{0,k}(h)|(m+1 \le k \le m+p)\}$ to transmit $\chi_{0,m+1}(h), \ldots, \chi_{0,m+p}(h)$ and all other nodes not in $\mathbf{V}_{rhy}$ keep following static schedule S (see the red shaded region in Fig. 2). Furthermore, during $[r_{0,m+1}, r_{0,m+p+1})$, V_h generates transmission updates $\{\tilde{s}_{0,k}(h)|(m+p+1 \le k \le m+2p)\}$ for the transmissions in the next p periods and propagates the corresponding information to V_{h+1} in a similar successive fashion. This pattern of (i) successively generating and propagating $\{\tilde{s}_{0,k}(h)|(m+p+1 \le k \le m+2p)\}$

[3]Note that the inconsistency between the static and dynamic schedules can be handled by the MP-MAC protocol [9] where more critical rhythmic packets can always be successfully transmitted.

[4]We assume that the controller can collect the local interference table of each device node to build the global interference table.

[5]We will discuss the significance and impact of p in Sec. V-B.

in the current p rhythmic periods along the routing path of τ_0, and (ii) following $\{\tilde{s}_{0,k}(h)|(m+p+1 \leq k \leq m+2p)\}$ in the next p rhythmic periods, is repeated until all the rhythmic packets are processed.

For ease of discussion, in the rest of the paper, we refer to the time duration from the time when the disturbance is detected (t^*) to the start time of the system rhythmic mode ($t_{n\to r} = r_{0,m+1}$) as *disturbance response time* (DRT), and the time duration of the system rhythmic mode as *disturbance handling time* (DHT).

Given the execution model of DS-PaS, one critical question to answer is how DS-PaS ensures successful avoidance of inconsistent transmissions when constructing the dynamic schedule by each node only based on their local interference information. Furthermore, according to the problem definition at the end of Sec. II, DS-PaS should be able to generate the dynamic schedule that results in the minimum number of dropped periodic packets when disturbances cause the system to be overloaded. Finally, DS-PaS should be light-weight so it can be easily implemented on resource-constrained RTWN nodes. In the follow sections, we will present the details on how we resolve these issues in the DS-PaS design.

IV. Avoiding Inconsistent Transmissions

DS-PaS aims to eliminate inconsistent transmissions in the constructed dynamic schedule $\tilde{S}$. The key to achieve this goal is to ensure that the sender and the receiver of a transmission allocate the same time slot and channel to the transmission. DS-PaS achieves this by performing the successive packet scheduling process (as outlined in Sec. III) obeying the following rules.

Rule 1. *Each sender node along the routing path of the rhythmic task, τ_0, determines the transmission updates for the packets in the* subsequent *periods instead of the current period.*

Rule 2. *Each sender node V_h of $\chi_{0,k}(h)$ generates transmission update $\tilde{s}_{0,k}(h)$ in the system rhythmic mode, and propagates $\tilde{s}_{0,k}(h)$ to its receiver V_{h+1} on τ_0's routing path in a sequential manner. Each receiver node does not modify the assigned time slots and channels of the transmissions determined by the sender node.*

Rule 1 ensures that the newly generated transmission update can be propagated from the first node to the last node along τ_0's routing path before the update is used in the subsequent periods. Rule 2 guarantees that any pair of sender and receiver nodes always use the same time slot and channel assignment for its transmissions.

There are two key questions to be answered to enforce Rule 1 and Rule 2. (i) What information should be passed from each sender node to its receiver node? The answer to this question fulfills the design requirement of our successive scheduling mechanism. (ii) When should the transmission update be generated within the current period of τ_0? The answer to this question impacts the response time of the system to handle disturbance (*i.e.*, DRT).

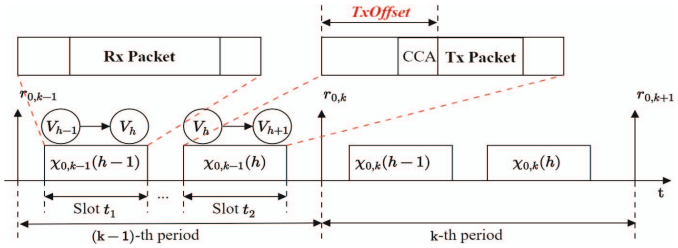


Fig. 3. Illustration of key time points for generating dynamic schedule.

For the ***first*** question, we consider node V_h as the receiver of transmission $\chi_{0,k}(h-1)$ and the sender of transmission $\chi_{0,k}(h)$, and we discuss what information should be piggybacked in transmission $\chi_{0,k-1}(h-1)$ received by V_h (see Fig. 3). To determine the transmission update $\tilde{s}_{0,k}(h)$ for $\chi_{0,k}(h)$ in the next period, V_h (as the receiver) needs the following information from the sender (V_{h-1}) through transmission $\chi_{0,k-1}(h-1)$ in the current period. (i) To successfully receive $\chi_{0,k}(h)$'s preceding transmission $\chi_{0,k}(h-1)$, the slot index and the channel index of $\tilde{s}_{0,k}(h-1)$ are necessary for V_h because our successive scheduling mechanism requires that each receiver must follow the transmission updates generated by the sender. (ii) To determine $\tilde{s}_{0,k}(h)$, V_h needs to know which periodic packets are dropped when scheduling the previous rhythmic transmissions before $\chi_{0,k}(h)$ such that V_h can utilize these released time slots from the dropped periodic packets. Therefore, the information to be passed along with $\chi_{0,k-1}(h-1)$ from the sender (V_{h-1}) to the receiver (V_h) consists of slot index and channel index of transmission $\chi_{0,k}(h-1)$, and task/packet IDs of the dropped periodic packets[6] determined by the previous transmission updates. Note that, according to the successive scheduling mechanism, each node V_h can only receive information from all the predecessor nodes along the routing path, *i.e.*, $V_1, \ldots, V_{h-1}$. Hence, previous transmission updates include all $\tilde{s}_{0,\{1,\ldots,k\}}(\{1,\ldots,h-1\})$.

Now we analyze and demonstrate that the payload overhead for propagating the above information is acceptable. When $p = 1$, it requires 2 bytes to store the slot index (12 bits) and channel index (4 bits) for one transmission. Similarly, 2 bytes are needed to store the task ID (7 bits) and packet ID (9 bits) for one dropped periodic packet. The Maximum Transmission Unit (MTU) in the IEEE 802.15.4 physical layer equals 127 bytes including 29 bytes for MAC header/footer and 2 bytes for slot and channel offsets. Thus, 96 bytes are available for representing 48 dropped periodic packets, which are sufficient in most cases to store the information needed for one rhythmic transmission to determine a feasible cell to transmit. Moreover, our simulation results show that the average packet drop rate under DS-PaS is within 2% (see details in Sec. VI). That is, 2 out of 100 periodic packets on average need to be dropped in the system rhythmic mode, which is far smaller than the 48 dropped packets allowed.

The ***second*** question is *when* transmission update $\tilde{s}_{0,k}(h)$ should be generated by V_h after V_h receives the information

[6]The task/packet IDs of dropped periodic packets determined by each hop is incrementally added into the packet.

stated above from previous node V_{h-1}. As shown in Fig. 3, suppose in the $(k-1)$-th period, V_h receives the transmission update information from V_{h-1} (or detects the disturbance if V_h is the sensor node) at time slot t_1 and the transmission update $\tilde{s}_{0,k-1}(h-1)$ (or the static schedule) assigns time slot t_2 to transmission $\chi_{0,k-1}(h)$ in the $(k-1)$-th period. If the transmission update $\tilde{s}_{0,k}(h)$ for $\chi_{0,k}(h)$ sent by V_h can be generated before transmitting $\chi_{0,k-1}(h)$ in t_2, $\tilde{s}_{0,k}(h)$ can be piggybacked to the receiver node V_{h+1} in $\chi_{0,k-1}(h)$ and followed by V_h and V_{h+1} in the k-th period. In this case, the transmission updates for all the transmissions $\chi_{0,m+1}$ in the first rhythmic period can be generated and received by all the nodes along the routing path in the period during which the disturbance is detected. Therefore, $DRT = P_0$, one nominal period. Otherwise, if $\tilde{s}_{0,k}(h)$ is generated *after* transmitting $\chi_{0,k-1}(h)$ in the $(k-1)$-th period, V_h has to propagate $\tilde{s}_{0,k}(h)$ to V_{h+1} in the k-th period. That is, each node requires one period to generate the transmission update plus one additional period to propagate it. Since the propagation of $\tilde{s}_{0,k}(h)$ followed by the generation of $\tilde{s}_{0,k}(h+1)$ can be accomplished within one period, the entire transmission updates for a packet $\chi_{0,k}$ require a total of $(H_0+1)\times P_0$ periods in order to be received by all H_0+1 nodes along the routing path, *i.e.*, $DRT = (H_0+1)\times P_0$.

Next we analyze whether it is possible to generate the transmission update for each transmission before transmitting the packet in the current period, *i.e.*, achieving $DRT = P_0$. As shown in Fig. 3, packet transmission of V_h happens in the **Tx Packet** within time slot t_2. Thus, any sufficiently long idle time duration before **Tx Packet** can be used by V_h to generate the dynamic schedule. Apparently, an idle slot[7] (15ms) for V_h is eligible for generating the transmission update. If such an idle slot does not exist before V_h sending the packet in the $(k-1)$-th period at t_2, the time duration ***TxOffset*** before **Tx Packet** (see Fig. 3) can be another candidate for V_h to generate the transmission update. Through extensive experimental evaluation, we demonstrate that the length of ***TxOffset*** (3180μs) is sufficient for a typical RTWN device (TI CC2538) to generate the transmission update for the transmission in the following rhythmic period, thus achieving $DRT = P_0$.

Theorem 1. *DS-PaS guarantees that all rhythmic transmissions are consistent transmissions.*

Proof. According to the definition of consistent transmission in Def. 3, we only need to show that for any rhythmic transmission $\chi_{0,k}(h)$, its sender V_h and receiver V_{h-1} both allocate $\chi_{0,k}(h)$ to slot offset t and channel offset c at time t.

According to Rule 1 in DS-PaS, sender V_h determines $\chi_{0,k}(h)$'s transmission update $\tilde{s}_{0,k}(h)$ (*i.e.*, slot offset t and channel offset c) in a preceding period of packet $\chi_{0,k}$. Suppose V_h determines $\tilde{s}_{0,k}(h)$ at t' in the $(k-q)$-th period. Then we have $t' < t$. According to Rule 2, receiver V_{h-1} receives $\tilde{s}_{0,k}(h)$ from sender V_h at time t'' and does not modify slot offset t and channel offset c for $\chi_{0,k}(h)$. Based on our previous analysis, V_h is able to complete the generation of $\tilde{s}_{0,k}(h)$ before transmitting the packet in the same period either through an idle slot or a ***TxOffset*** duration. That is, V_{h-1} receives $\tilde{s}_{0,k}(h)$ in the $(k-q)$-th period. Thus, we have $t'' < t$. Since $t' < t$ and $t'' < t$, sender V_h and receiver V_{h-1} both transmit and receive $\chi_{0,k}(h)$ in slot t and channel c. Therefore, $\chi_{0,k}(h)$ is a consistent transmission. □

[7]Any slot within which V_h is not sending or receiving a packet is an idle slot for V_h.

V. Constructing Dynamic Schedule

Under the successive scheduling mechanism discussed in the previous sections, once a disturbance is detected, each node in $\mathbf{V}_{rhy}$ determines a transmission update locally and propagates it along the routing path of the rhythmic task. The construction of the transmission update, *i.e.*, the dynamic schedule in the system rhythmic mode must guarantee that (i) rhythmic transmission assignment made by preceding nodes remain unchanged, (ii) all rhythmic packets meet their deadlines, (iii) a minimum number of periodic packets are dropped, and (iv) the system can reuse the static schedule after the rhythmic mode ends and all packets meet their nominal deadlines. Below, we elaborate how DS-PaS determines the transmission updates at each node in $\mathbf{V}_{rhy}$.

A. Problem Formulation

In DS-PaS, the network starts operation by following the static schedule S which guarantees that all tasks meet their nominal deadlines if no disturbance occurs. Schedule S is generated by the controller node (*i.e.*, gateway) which has the global interference information. We denote the static schedule S as a set of tuples, *i.e.* $S = \{(t, c, i, h)\}$, which represents that channel c at time slot t is assigned to task τ_i's h-th hop. For brevity, we refer to a pair of given time slot t and channel c as a *cell*, and denote the above assignment as $a[t, c] = \{(i, h)\}$. If two transmissions (i, h_i) and (j, h_j) have no shared node and no interference, these two transmissions can be scheduled in the same cell without impacting each other, *i.e.*, $a[t, c] = \{(i, h_i), (j, h_j)\}$. If no transmissions are assigned to a cell, the cell is idle and is denoted as $a[t, c] = \{(-1, -1)\}$.

As shown in Fig. 2, when disturbance is detected at t^*, τ_0 enters its rhythmic state at $r_{0,m+1}$, *i.e.*, $t_{n\to r} = r_{0,m+1}$. A dynamic schedule $\tilde{S}$ consisting of transmission updates $\tilde{s}_{0,m+l}(h)$ $(1 \le l \le R, 1 \le h \le H_0)$ is thus needed for each node V_h $(1 \le h \le H_0+1)$ to handle the increased rhythmic packets before the system switches back to the nominal mode and resumes the use of S. The system rhythmic mode starts from $t_{n\to r}$ and ends at a carefully chosen end point t_{ep} to achieve guaranteed disturbance handling time (DHT).

When a disturbance occurs, the increased rhythmic workload needs more resource to guarantee the deadlines of all rhythmic packets. Hence, some periodic transmissions have to be dropped. Since any node $V \notin V_{rhy}$ keeps following static schedule S to transmit periodic packets, we choose not to adjust the assignment of periodic transmissions in $\tilde{S}$. Thus,

if any periodic transmission $\chi_{i,j}(h_i)$ in S is dropped due to an updated rhythmic transmission in $\tilde{S}$ (we will discuss what conditions cause a dropped periodic transmission in Sec. V-B), we say periodic packet $\chi_{i,j}$ is dropped. Then, the question is which periodic transmissions should be dropped to achieve the desired dynamic schedule $\tilde{S}$ consisting of transmission updates $\tilde{s}_{0,m+l}(h)$ $(1 \leq l \leq R, 1 \leq h \leq H_0)$. Formally, to satisfy the four requirements listed at the beginning of this section, we aim to solve the following problem.

Problem 1 – Dynamic Schedule Generation: Determine transmission updates $\tilde{s}_{0,m+l}(h)(1 \leq l \leq R, 1 \leq h \leq H_0)$ at each sender node V_h on τ_0's routing path such that the total number of dropped periodic packets in the system rhythmic mode, denoted as $\rho[t_{n\to r}, t_{ep})$, is minimized and the following two constraints are satisfied.

Constraint 1. $f_{0,k}(h) \leq d_{0,k}(h)$.

Constraint 2. *In the dynamic schedule $\tilde{S}$, any periodic transmission $\chi_{i,j}(h_i)(1 \leq i \leq N)$ originally occupying cell $a[t,c]$ in S,* i.e.*, $a[t,c] = \{(i,h_i)\}$, can be (i) replaced by a rhythmic transmission,* i.e.*, $a[t,c] = \{(0,h_0)\}$, (ii) transmitted in parallel with a rhythmic transmission,* i.e.*, $a[t,c] = \{(i,h_i),(0,h_0)\}$, if they do not interfere with each other, (iii) removed due to transmission conflict with a rhythmic transmission,* i.e.*, $a[t,c] = \{(-1,-1)\}$, or (iv) kept unchanged.*

The choice of t_{ep} in **Problem 1** directly impacts the disturbance handling time, DHT. According to [9], each actual release time of τ_0 after $f_{0,m+R}$ can be a feasible end point candidate. To ensure a short DHT, we select the first release time of τ_0 after $f_{0,m+R}$ as the end point, *i.e.*, $t_{ep} = r_{0,m+R+1}$.

B. Dynamic Schedule Generation

Generating transmission updates at each rhythmic sender node $V_h(1 \leq h \leq H_0)$ in a successive manner while guaranteeing minimum number of dropped periodic packets is challenging, especially given that each node V_h only maintains a local interference table. The key issue is how to let each node select a valid cell in the static schedule to transmit the rhythmic transmission so as to (i) guarantee the deadline of the rhythmic packet and (ii) drop the minimum number of periodic packets. To determine transmission update $\tilde{s}_{0,k}(h)$ for rhythmic transmission $\chi_{0,k}(h)$, V_h needs to select a certain cell in S within $[r_{0,k}, d_{0,k})$ to transmit $\chi_{0,k}(h)$. Below, we introduce a greedy heuristic running at each node V_h to perform such cell selection. The key idea of the heuristic is to select the cell resulting in the minimum number of dropped periodic packets according to the local interference table. The heuristic is efficient and can be readily implemented on a typical RTWN device.

To solve **Problem 1**, at the highest level, V_h needs to (i) determine a feasible time duration for allocating $\chi_{0,k}(h)$ in the dynamic schedule to satisfy the deadline of $\chi_{0,k}$, and (ii) select a cell within the time duration to minimize the number of dropped periodic packets. According to Sec. IV, V_h receives the slot index and channel index assigned to transmission $\chi_{0,k}(h-1)$ by V_{h-1}, denoted as cell $a[t_{h-1}, c_{h-1}]$), in the transmission update $\tilde{s}_{0,k}(h-1)$. Since transmission $\chi_{0,k}(h)$ cannot be transmitted before receiving $\chi_{0,k}(h-1)$, $\chi_{0,k}(h)$ can only be scheduled after time slot t_{h-1}[8]. On the other hand, to meet the deadline of $\chi_{0,k}$, $\chi_{0,k}(h)$ cannot be transmitted later than $d_{0,k} - (H_0 - h)$ since each subsequent transmissions $\chi_{0,k}(h+1), \ldots, \chi_{0,k}(H_0)$ requires at least one time slot to be transmitted. Therefore, any cell within time duration $(t_{h-1}, d_{0,k} - (H_0 - h)]$ can be a candidate to schedule rhythmic transmission $\chi_{0,k}(h)$.

Next, we describe how to select a cell leading to the minimum number of dropped periodic packets in time duration $(t_{h-1}, d_{0,k} - (H_0 - h)]$. From the definitions of transmission conflict and transmission interference, we know that any periodic transmission having transmission conflict (*resp.*, interference) with $\chi_{0,k}(h)$ cannot be scheduled within a same time slot (*resp.*, a same cell). Hence, according to Constraint P 1.2 in **Problem 1**, if V_h selects cell $a[t,c]$ to transmit rhythmic transmission $\chi_{0,k}(h)$, any periodic transmission $\chi_{i,j}(h_i)$ meeting one or both of the following two conditions needs to be dropped.

Condition 1: $\chi_{i,j}(h_i)$ is scheduled in a cell with slot index t, and has transmission conflict with rhythmic transmission $\chi_{0,k}(h)$.

Condition 2: $\chi_{i,j}(h_i)$ is scheduled in cell $a[t,c]$, and has transmission interference with rhythmic transmission $\chi_{0,k}(h)$ according to V_h's local interference table.

According to Sec. V-A, if any periodic transmission $\chi_{i,j}(h_i)$ is dropped, periodic packet $\chi_{i,j}$ is dropped. Note that each packet can only have one transmission in one time slot. Thus, the number of dropped periodic packets equals to the number of dropped periodic transmissions caused by the cell selection of $\chi_{0,k}(h)$ according to the above two conditions. Further, the subsequent periodic transmissions of $\chi_{i,j}$ (*i.e.*, $\chi_{i,j}(h_i+1), \ldots, \chi_{i,j}(H_i)$) are also dropped such that their assigned cells in the static schedule can be used by the subsequent nodes $V_{h+1}, \ldots, V_{H_0}$ to transmit rhythmic transmissions $\chi_{0,k}(h+1), \ldots, \chi_{0,k}(H_0)$. Generally, more dropped packets at V_h indicate that more cells in the static schedule are released by the dropped periodic packets and can be used to transmit rhythmic packet $\chi_{0,k}(h+1)$. Hence, intuitively a larger p (*i.e.*, the number of periods that each node V_h uses to determine the transmission updates and propagate the dropped packet information to V_{h+1}) can lead to better performance in terms of fewer dropped packets. However, our simulations show that increasing p does not significantly decrease the total number of dropped periodic packets in the system rhythmic mode. The reason is that the cells released by the dropped packets determined by V_h may not be used by V_{h+1} to transmit $\chi_{0,k}(h+1)$ due to transmission interference within these cells according to V_{h+1}'s local interference table. That is, due to different interference information stored at each

[8]If V_h is the source node (*i.e.*, $h = 1$), $t_{h-1} = r_{0,m+1}$.

Algorithm 1 Heuristic for Cell Selection

Input: $\chi_{0,k}(h)$, static schedule S, dropped periodic packets
Output: $a[t,c]$

```
a, b ← 0, N_{d_min} ← N; //N is the total number of periodic tasks in the system
Update S within (t_{h-1}, d_{0,k}−(H_0−h)] according to the dropped periodic packets;
for t ∈ (t_{h-1}, d_{0,k} − (H_0 − h)] do
    a ← the number of periodic transmissions having conflict with χ_{0,k}(h) at slot t;
    for c ∈ [1, C] do
        b ← the number of periodic transmissions having interference with χ_{0,k}(h) in cell a[t,c];
        N_d(V_h, a[t,c]) ← a + b;
        if N_d(V_h, a[t,c]) = 0 then
            // no transmission conflict and interference
            return a[t,c];
        end if
        if N_d(V_h, a[t,c]) < N_{d_min} then
            //update the cell assignment with the minimum number of dropped packets
            N_{d_min} ← N_d(V_h, a[t,c]);
            a_min ← a[t,c];
        end if
    end for
end for
return a_min;
```

node, V_h does not necessarily need to consider transmission updates determined by its subsequent nodes $V_{h+1}, \ldots, V_{H_0}$ when it determines $\tilde{s}_{0,k}(h)$. Therefore, we design a greedy cell selection heuristic running at each node as follows.

We use $N_d(V_h, a[t,c])$ to denote the number of dropped periodic packets if V_h selects cell $a[t,c]$ to transmit $\chi_{0,k}(h)$. Our greedy heuristic selects the cell with the minimum number of dropped periodic packets, denoted as N_{d_min}, in time duration $(t_{h-1}, d_{0,k} - (H_0 - h)]$. Alg. 1 gives the pseudo code of the heuristic. The time complexity of the cell selection algorithm is $O(P_0 \cdot C)$ where P_0 is the period of the rhythmic task and C is the number of channels.

VI. Performance Evaluation

In this section, we present key results obtained from both testbed and simulation experiments to evaluate the performance of DS-PaS framework in RTWNs.

A. Testbed Implementation and Evaluation

We implemented DS-PaS on a RTWN testbed to validate its correctness in achieving prompt response to unexpected external disturbances. Our RTWN testbed consists of 7 wireless devices[9] (TI CC2538 SoC + SmartRF evaluation board) running the OpenWSN stack. OpenWSN is an open source implementation of the 6TiSCH protocol. A typical OpenWSN network consists of an OpenWSN Root and several OpenWSN devices, as well as an optional OpenLBR (Open Low-Power Border Router) to connect to IPv6 Internet. In our testbed setup, one device is configured as the root (functioning as the controller) and the rest are configured as device nodes to form a multi-hop RTWN. We set the slotframe length to 30 time slots (15ms for each slot) and enable two channels in IEEE 802.15.4e. The transmission power of the devices is set to be -25 dbm to support spatial reuse even when devices are deployed in a small area. The testbed topology and interference information are given in Fig. 4(a).

Our proposed DS-PaS was implemented by enhancing the data link layer of the OpenWSN stack. In the testbed, an external disturbance is emulated by the user pressing a button on a sender node device. This will trigger a random task originated at this device to enter the rhythmic state. The devices along the routing path of this task follow DS-PaS to generate the dynamic schedule for all rhythmic packets. A 16-channel Open bench Logic Sniffer is used to record device radio activities by GPIO pins, in order to accurately measure the timing information among different devices. As shown in Fig. 4(b), each node has two rows showing their radio activities on two different channels. We deploy three tasks in the testbed: $\tau_0 = \{\{V_0, V_1, V_3, V_5, V_6\}, 10\}$, $\tau_1 = \{\{V_1, V_3, V_5, V_6\}, 6\}$ and $\tau_3 = \{\{V_1, V_2, V_4\}, 6\}$. For each task, the first element denotes the routing path and the second element gives the nominal period (relative deadline) of the task. We further assume that τ_0 is the rhythmic task and $\overrightarrow{P_0}(\overrightarrow{D_0}) = [6, 7, 8, 9]$.

Fig. 4(b) shows the slot information and radio activities for two slotframes when the system switches from its nominal mode (1st slotframe) to the rhythmic mode (2nd slotframe). Here, a falling edge or a rising edge in the top most waveform marks the start of a new slotframe. The second waveform from the top represents time slot, and the fourteen waveforms below show the radio activities (either transmitting or receiving) of the respective seven nodes on two different IEEE 802.15.4e channels, as indicated in the left panel.

In this experiment, we let V_0 enter the rhythmic mode, and thus τ_0's period reduces from 10 to 6. In the rhythmic mode (2nd slotframe), τ_0 has to handle 4 packets while it only needs to handle 3 packets in the system nominal mode (1st slotframe) (see the 3nd and the 4th waveform from top). As shown in Fig. 4(b), at slot-offset 4 in the 1st slotframe, there are three transmissions ($V_0 \rightarrow V_1$, $V_2 \rightarrow V_4$, and $V_5 \rightarrow V_6$). Transmission $V_0 \rightarrow V_1$ and transmission $V_2 \rightarrow V_4$ happen on different channels, while transmission $V_0 \rightarrow V_1$ and transmission $V_5 \rightarrow V_6$ happen on the same channel since they are beyond interference range of each other and spatial reuse is allowed. The transmissions of τ_0's packets are marked by red dashed lines in Fig. 4(b). To accommodate the increased workload (*i.e.*, four packets of τ_0 instead three as in the nominal mode) in the system rhythmic mode, the first released packet of τ_1 in the 2nd slotframe is dropped. V_1 sends τ_0's packet instead of τ_1's packet. However, since V_2 and V_4 do not receive the disturbance information, they still follow the static schedule by enabling their radios and waiting for τ_1's packet at slot offset 3 and 4, respectively (marked by the purple circles in Fig. 4(b)). This behavior exactly matches the results from the simulation. We also measured the computation overhead

[9]Our Logic Sniffer only has 16 available channels, and 2 channels are needed to display the slot information. Thus, we only have 14 remaining channels which can support at most 7 nodes in a 2-channels RTWN.

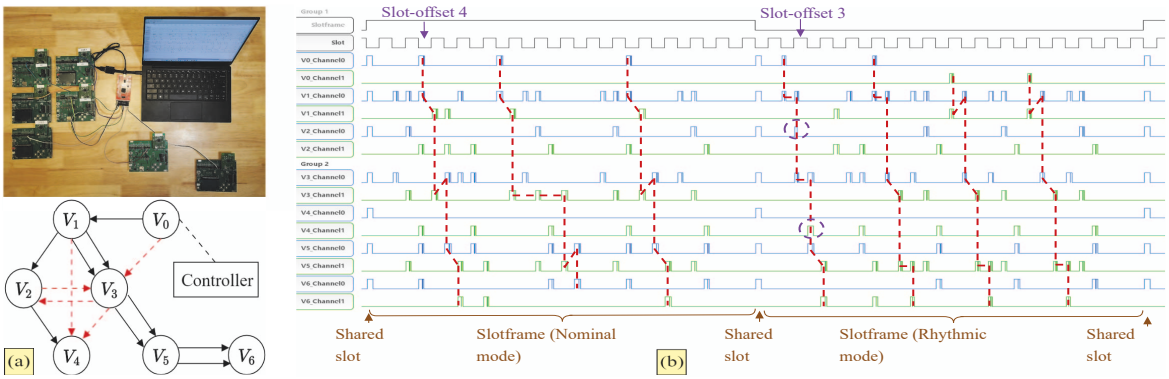


Fig. 4. (a) An overview of the testbed and employed network topology/task set used for functional validation of the DS-PaS framework. (b) Slot information and radio activities in the test case captured by Logic Sniffer. A blue (or green) rectangle represents a send/accept operation on Channel 0 (or Channel 1).

of the cell selection algorithm on the testbed. Executing the cell selection algorithm takes 5500 CPU cycles on CC2538 with the CPU frequency of 32MHz. Thus, executing the cell selection algorithm takes $5500/(32 \times 1024 \times 1024) = 163\mu s$, which is a rather small overhead compare to the ***TxOffset*** length (3180μs) and the slot length (15000μs).

B. Simulation Study

In the simulation study, we compare DS-PaS with the state-of-the-art approach FD-PaS. Both DS-PaS and FD-PaS can handle unexpected external disturbances in RTWNs, but FD-PaS does not exploit multiple channels and spatial reuse. We compare their performance using two metrics. **Acceptance Ratio (AR)**: the fraction of feasible task sets over all the generated task sets. (Note that a task set is feasible only if it is schedulable in the system nominal mode under EDF. Our experiments are to evaluate the improved system schedulability benefited from multi-channel RTWNs supporting spatial reuse.) **Drop Rate (DR)**: the ratio between the number of dropped periodic packets and the total number of generated packets in the system rhythmic mode.

To ensure fairness, we randomly generate network topologies and task sets. We use an abstract 10×10 grid to construct the network and ensure that the generated topology is a connected graph. The total number of nodes, M, is an integer in the set of $\{10, 20, \ldots, 100\}$. The number of channels C is an integer in the set of $\{1, 2, \ldots, 8\}$.

After a feasible network topology is constructed, we generate random periodic task sets. Each task set is generated according to a target normalized utilization U (defined as the total utilization U^* of all the tasks divided by the number of channels) in the system nominal mode and by incrementally adding random periodic tasks to an initially empty set $\mathcal{T}$. Each task τ_i is generated according to the number of hops H_i, nominal period P_i randomly selected from $[20, 50]$, and nominal relative deadline $D_i = P_i$. After all tasks in $\mathcal{T}$ are generated, one task is randomly selected as the rhythmic task τ_0. We control the workload of the rhythmic task by tuning the number of elements in $\overrightarrow{P_0}$, denoted as R, which can be any integer in the set of $\{4, 6, \ldots, 16\}$.

In the first set of experiments, we compare the average AR of FD-PaS and DS-PaS by varying the number of nodes (M), normalized utilization (U), and number of channels (C). We first evaluate the schedulability improvement brought by spatial reuse by setting $C = 1$. As shown in Fig. 5(a), when $U \geq 1$, the AR of FD-PaS drops to 0 while DS-PaS still maintains a high AR. This is because spatial reuse is supported by DS-PaS where multiple transmissions can be scheduled within one time slot. Further, we can observe from Fig. 5(a) that AR increases along with the increase of M since more spatial reuse opportunities can be exploited as the network scales up. Next, we set $M = 60$ with a fixed network topology and vary the number of channels C. Fig. 5(b) shows that DS-PaS maintains a relatively high AR while the AR of FD-PaS drops to 0 when $C \geq 2$ even in the case of $U < 1$. The reason is that for a fixed number of nodes M, more transmission conflicts (caused by shared nodes) may occur within each time slot when the total utilization U^* increases with the increase of C. Note that U is normalized utilization and the total task utilization U^* is higher for larger C. Last, Fig. 5(c) shows the AR of both methods with varying M and C where the trends of AR are similar to those in the previous experiments.

In the second set of experiments, we compare the average DR of FD-PaS and DS-PaS by varying normalized utilization U in the nominal mode, number of nodes M, and number of rhythmic periods R of the rhythmic task. For each framework, we also evaluate the DR gap between storing the global interference table and the local interference table at individual nodes when making the packet dropping decisions. For fairness, we compare the DR when FD-PaS and DS-PaS are both schedulable in the nominal mode. We fix the number of channels as $C = 2$ and modify the packet dropping algorithm in FD-PaS to support multiple channels. The "$-L$" and "$-G$" in the legends indicate the local and global interference case, respectively. Fig. 6 summarizes the DR results. We can observe that DS-PaS achieves a significantly lower average DR compared to FD-PaS. The data also show that with the global interference information, DR can be further reduced. However, the difference between DS-PaS-L and DS-PaS-G is rather small. This is because when $U < 1$, DS-PaS can already achieve a very low DR. From Fig. 6(a) and Fig. 6(b), we can

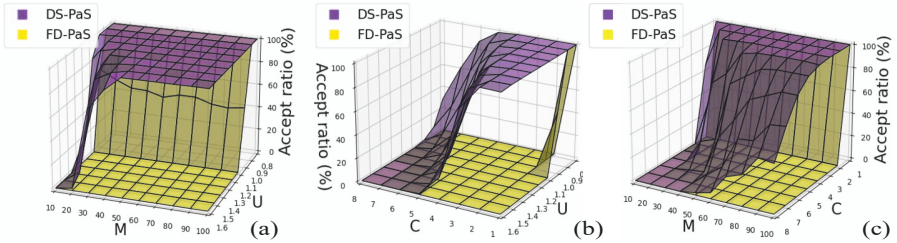


Fig. 5. Comparison of the avg. AR between DS-PaS and FD-PaS under different settings, (a) $C = 1$, (b) $M = 60$, and (c) $U = 0.9$.

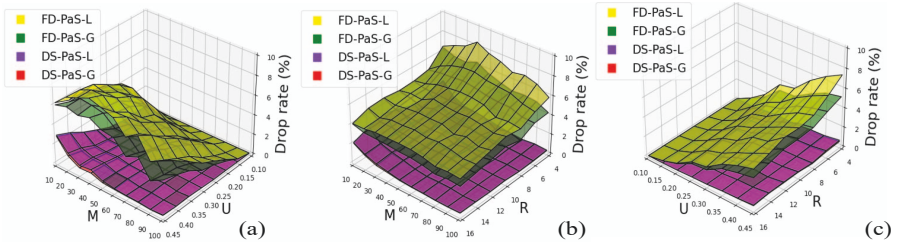


Fig. 6. Comparison of the avg. DR between DS-PaS and FD-PaS under different settings, (a) $C = 2, R = 4$, (b) $C = 2, U = 0.45$, and (c) $N = 60, C = 2$. The DS-PaS-G surface mostly co-insides with the DS-PaS-L one.

observe that when M increases, DR drops due to more spatial reuse opportunities. Another observation from Fig. 6(a) and Fig. 6(c) is that when U increases, DR increases due to more workloads in the network. An interesting observation is that when R (the number of rhythmic periods, hence the number of rhythmic packets) increases, the average DR drops first then increases (see Fig. 6(b) and Fig. 6(c)). This is because when R is small, though we need to drop more packets due to the increase in the workload of the rhythmic task, the number of dropped packets grows more slowly than the number of packets in the rhythmic mode.

VII. Conclusion and Future Work

In this paper, we propose DS-PaS, a distributed successive packet scheduling framework, to handle external disturbances in multi-channel RTWNs with spatial reuse while guaranteeing to meet the deadlines of all critical tasks. DS-PaS is built on a novel successive scheduling mechanism and an efficient packet dropping heuristic method. Our extensive experiments validate the correctness and effectiveness of DS-PaS. As future work, we will explore how to handle concurrent disturbances in a distributed fashion.

VIII. Acknowledgement

This work is supported in part by the U.S. NSF under Grant No. CCF-2028875, CCF-2028879 and CNS-2008463.

References

[1] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," *J. Ind. Inf. Integr.*, vol. 6, pp. 1–10, 2017.

[2] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power wireless bus," in *SenSys*, 2012, pp. 1–14.

[3] W. Shen, T. Zhang, M. Gidlund, and F. Dobslaw, "Sas-tdma: a source aware scheduling algorithm for real-time communication in industrial wireless sensor networks," *Wirel. Netw.*, vol. 19, no. 6, pp. 1155–1170, 2013.

[4] X. Fu and Y. Yang, "Modeling and analysis of cascading node-link failures in multi-sink wireless sensor networks," *Reliab. Eng. Syst. Saf.*, vol. 197, p. 106815, 2020.

[5] 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 Trans Mob Comput*, vol. 18, no. 11, pp. 2502–2517, 2018.

[6] 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 *ECRTS*, 2019.

[7] S. Hong, X. S. Hu, T. Gong, and S. Han, "On-line data link layer scheduling in wireless networked control systems," in *ECRTS*, 2015, pp. 57–66.

[8] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled tsch," in *Sensys*, 2015, pp. 337–350.

[9] 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 Trans Mob Comput*, vol. 20, no. 2, pp. 502–518, 2019.

[10] V. P. Modekurthy and A. Saifullah, "Online period selection for wireless control systems," in *ICII*, 2019, pp. 170–179.

[11] J. Shi, M. Sha, and Z. Yang, "Digs: Distributed graph routing and scheduling for industrial wireless sensor-actuator networks," in *ICDCS*, 2018, pp. 354–364.

[12] V. P. Modekurthy, A. Saifullah, and S. Madria, "Distributedhart: A distributed real-time scheduling system for wirelesshart networks," in *RTAS*, 2019, pp. 216–227.

[13] G. Zhou, T. He, J. A. Stankovic, and T. Abdelzaher, "Rid: Radio interference detection in wireless sensor networks," in *INFOCOM*, vol. 2, 2005, pp. 891–901.

[14] M. A. Moridi, Y. Kawamura, M. Sharifzadeh, E. K. Chanda, M. Wagner, H. Jang, and H. Okawa, "Development of underground mine monitoring and communication system integrated zigbee and gis," *Int. J. Min. Sci*, vol. 25, no. 5, pp. 811–818, 2015.

[15] J. Kim, K. Lakshmanan, and R. Rajkumar, "Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems," in *ICCPS*, 2012, pp. 55–64.