Real-Time Scheduling for Event-Triggered and Time-Triggered Flows in Industrial Wireless Sensor-Actuator Networks

Xi Jin, Abusayeed Saifullah, Chenyang Lu, and Peng Zeng

***Abstract*—Wireless sensor-actuator networks enable an efficient and cost-effective approach for industrial sensing and control applications. To satisfy the real-time requirement of such applications, these networks adopt centralized scheduling algorithms to optimize the real-time performance based on global information. Existing centralized algorithms mostly focus on scheduling time-triggered flows. They cannot effectively schedule event-triggered flows due to the dynamics and unpredictability of events. In this paper, we propose three fundamental centralized algorithms that reserve as few resources as possible for event-triggered flows such that the real-time performance of time-triggered flows is not affected. We then analyze their advantages and disadvantages. Based on the analysis, we combine their advantages, including those in terms of their resource requirements, into a centralized algorithm. Finally, we conduct extensive simulations based on both real topologies and random topologies. The simulations indicate that for most test cases the schedulability of our combined algorithm is close to optimal solutions.**

I. Introduction

Wireless sensor-actuator networks (WSANs) enable an efficient and cost-effective approach for industrial sensing and control applications. A large class of industrial control applications impose real-time requirements between sensing and actuation. For example, in the cement production process, the temperature data of rotary kilns must be delivered to the control room before their deadlines. If a data packet with a high-temperature warning misses its deadline, timely control of temperature fails which may cause kiln explosion. Specifically, excessive or unbounded latency may lead to highly unstable systems, making real-time performance a critical requirement in industrial WSANs.

Since Time Division Multiple Access (TDMA)-based approaches can provide predictable latency, they are preferred over CSMA/CA-based ones for transmission scheduling in industrial WSANs [1]. TDMA-based approaches are usually adopted through a centralized algorithm that can exploit the global information and may optimize the real-time performance. Currently, most WSANs with stringent real-time requirements adopt centralized approaches [2].

Industrial WSANs may employ both time-triggered flows and event-triggered flows such as the flows involving periodic sensing data and sporadic emergency alarms. For a *time-triggered flow*, its packets are generated and sent periodically with a fixed period. Hence, from the release time of the first packet, all subsequent packets' release times can be known. Based on this, flow paths, and network topologies, centralized algorithms generate schedules that are disseminated to each node before the system starts. On the other hand, for *event-triggered flows*, the occurrence of events is unpredictable. Centralized algorithms have difficulty in generating schedules for these dynamic event-triggered packets. In real industrial systems, when an event occurs on a node, the node first sends a message to the network manager which then regenerates the schedule considering this new flow and redisseminates [3]. This approach introduces a long latency, affecting the real-time performance of event-triggered packets. State-of-the-art approaches do not assign dedicated time slots to event-triggered packets; instead they allow them to preempt the time slots assigned to time-triggered packets [4], [5]. While this method may satisfy the requirements of event-triggered packets, it may cause some important time-triggered packets miss their deadlines.

In this paper, we propose centralized algorithms to schedule all time- and event-triggered flows under real-time constraints. These algorithms reserve time slots for event-triggered flows such that the reserved time slots are sufficient to handle event-triggered flows, and the real-time performance of time-triggered packets is not affected. This approach thus avoids introducing long latency as well as the preemption of time-triggered packets. First, we propose three fundamental scheduling approaches: (1) a *virtual-period method (VP)*, which changes event-triggered flows to virtual time-triggered flows such that scheduling algorithms for time-triggered flows can be used; (2) a *slot-multiplexed method (SM)*, which is an optimal algorithm when the objective is to minimize the number of reserved time slots for event-triggered flows; (3) a *reverse-scheduling method (RS)*, which not only decreases the number of reserved time slots, but also reduces the workload of network nodes. Second, we analyze the advantages and disadvantages of the three methods, and combine their advantages to propose an algorithm that reserves as few resources as possible and makes the schedules easy to disseminate. Finally, we evaluate our algorithms based on the topologies of a physical WSAN testbed and random topologies. The simulations indicate that when the node workload is less than 90% of the maximum workload, regardless of what the network workload is, the difference of schedulable ratios between our combined algorithm and optimal solutions remains less than 30%.

In the rest of the paper, Section II reviews related work. Section III presents our system model and problem. Section IV introduces the three fundamental algorithms. Section V compares their performances, and Section VI proposes an

X. Jin and C. Lu are with the Department of Computer Science and Engineering, Washington University in St. Louis, USA. X. Jin and P. Zeng are with the Shenyang Institute of Automation, Chinese Academy of Sciences, China. A. Saifullah is with the Department of Computer Science, Wayne State University, USA. X. Jin and P. Zeng were supported by NSFC (61502474 and 61533015), Youth Innovation Promotion Association of the Chinese Academy of Sciences, and Liaoning Provincial Natural Science Foundation of China (20180520029). C. Lu was supported by the Fullgraf Foundation.

algorithm combining their advantages. Section VII presents our simulation results. Section VIII concludes the paper.

II. Related Work

Real-time scheduling for WSANs has been widely studied [6]. The work in [7] proves that the scheduling problem of WSANs is NP-hard, and then derives a strong necessary condition for schedulability. After that, to improve the real-time performance of WSANs, many centralized scheduling algorithms are proposed, such as assigning fixed priorities [8], [9], assigning segmented slots [10], eliminating bottleneck [11], and addressing spatial re-use [12]. However, these centralized algorithms assign communication resources only to time-triggered packets. If these algorithms are used to handle event-triggered packets that are released at every time slot, a large number of resource reservations must lead to extremely low schedulability.

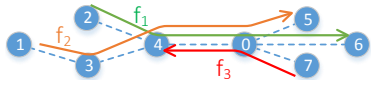
To handle event-triggered packets, some existing approaches allow to preempt the resource of time-triggered packets [4] or assign time slots that are shared by both types of packets [13]. In such approaches, time-triggered packets may be dropped or miss deadlines. In the approach proposed in [5], before event-triggered packets are transmitted, all nodes switch to an emergency state, and stop transmitting time-triggered packets. Scheduling and routing proposed in [14], [15] aims to reduce the preemption of time-triggered packets by event-triggered packets. In contrast, we propose real-time scheduling algorithms that do not allow event-triggered packets to preempt time-triggered packets and reserve as few time slots as possible for event-triggered packets such that the real-time requirements of all packets can be guaranteed.

III. Problem Statement

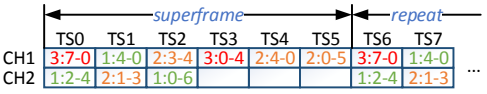
A WSAN is characterized by a two-tuple $< N, L >$. The node set N includes a gateway n_0 and sensor/actuator devices n_i. The gateway connects with an access point, and each sensor/actuator device is equipped with a single half-duplex transceiver. Therefore, a node cannot receive and send simultaneously. The network manager software, including scheduling, routing and maintenance algorithms, is implemented in the gateway. The link set L denotes the network topology. If nodes n_i and n_j can directly communicate with each other, then the link $l_{i,j}$ in set L is equal to 1; otherwise, $l_{i,j} = 0$.

The flow set is denoted by $F = \{f_1^e, f_2^e, ..., f_1^t, f_2^t, ...\}$. Each time-triggered flow $f_i^t \in F^t$, where $F^t \subseteq F$, generates a packet periodically with a period p_i^t, and its relative deadline is *implicit*, i.e., equal to its period p_i^t. The periods of time-triggered flows are harmonic, i.e., $p_i^t = p' \times 2^x$, where x is an integer, and p' is the unit period that contains a certain number of time slots. For a time- and event-triggered hybrid network, the period of the schedules is not less than the least common multiple of the periods of time-triggered flows. Harmonic periods are widely used in industrial WSANs [3] and real-time embedded systems [16] as the hyper-period (least common multiple) remains small that simplifies scheduling. We assume that at time slot 0 all of the time-triggered flows release their first packets. Then, time-triggered flow f_i^t releases its j-th packet at time slot $j \times p_i^t$, and its absolute deadline is $(j+1) \times p_i^t$. Event-triggered flow $f_i^e \in F^e$, where $F^e \subseteq F$, is aperiodic. An event-triggered packet can be released at any time t but must be delivered to its destination within its absolute deadline $t + d_i^e$. The time interval $[t, t + d_i^e]$ is called its *active interval*. We assume that when flow f_i^e releases a packet, it does not release another before its deadline. If an event-triggered flow has to release multiple packets, it can be regarded as multiple flows, and each of them releases one packet. The routing path π_i^* (* is a wild-card character, representing ^{e} and ^{t}) of flow f_i^* is from a sensor sn_i^* via the gateway n_0 to an actuator dn_i^*, and contains c_i^* hops. Since routing is already well-studied [17], we do not propose any routing algorithm and consider that the routes are already generated based on some existing algorithms [18].

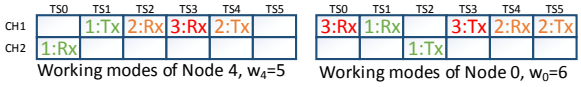
WSANs support 16 non-overlapping channels defined in the IEEE 802.15.4 standard some of which may remain unavailable due to external interference. We use m $(1 \leq m \leq 16)$ to denote the number of available channels. The scheduling algorithm is based on the time slotted channel hopping (TSCH) MAC protocol [19]. TSCH has two dimensions: time slots (TS) and channels (CH) as shown in Fig. 1(b). All nodes are time synchronized, and can access all channels. Scheduling algorithms assign a time slot and a channel to each transmission. A *transmission* $\tau_{i,j}^*$ denotes the j-th hop of a packet of flow f_i^*. Nodes change their working modes, Transmit (Tx) or Receive (Rx), based on assignment information. Two transmissions that involve a common node cannot be scheduled at the same time slot. This situation is called *node conflict*.



(a) Three flows



(b) Superframe



(c) Disseminated working-mode tables

Fig. 1. An example of a traditional time-triggered schedule

Assignment information is organized into *superframes*. If all flows are time-triggered, the length of a superframe is equal to the least common multiple of their periods. The network manager disseminates the working modes of one superframe to each node. Thus, nodes store their own working-mode information in local memory and change their modes accordingly, and then, the network runs continuously. Fig. 1 shows an example. When the three time-triggered flows have the same period 6, the superframe length is 6. At time slot $TS1$ and on channel $CH1$, the second hop of f_1 is sent from n_4 to n_0; hence, n_4 and n_0 are in Tx mode and Rx mode, respectively. In Fig. 1(c), the working-mode tables of n_4 and n_0 are shown. We use w_i to denote the number of working-mode entries of node n_i in one superframe, and $w_4 = 5$, $w_0 = 6$. Since the nodes are memory-constrained, the number of working-mode entries is restricted. The upper bound of w_i is

W. Since event-triggered flows are not periodic, the traditional superframe is not suitable for a time- and event-triggered hybrid network. In Sections IV and V, we will discuss how to reserve time slots for event-triggered flows.

Our objective is to schedule the transmissions so as to meet the following three constraints.

- *Real-time constraint*: All of the time-triggered packets and event-triggered packets have to be delivered to destinations before their deadlines.
- *Node-conflict constraint*: A node can serve (i.e., transmit or receive) at most one transmission at a time slot.
- *Resource constraint*: For each node, the number of working-mode entries is not greater than W.

A set of flows is called *schedulable*, if it has a feasible schedule that meets all the above constraints. A scheduling algorithm is *optimal*, if it can find a feasible schedule whenever there exists one. Note that the scheduling problem studied in [7] has $F^e = \emptyset$ and hence is a special case of our problem. Since the problem in [7] is NP-hard, our problem is also NP-hard. Therefore, we will design highly efficient heuristic algorithms. Our algorithms aim to guarantee the real-time performance and reliability of all flows. Although the reserved time slots for event-triggered flows are not used when no event-triggered flow occurs, the time-triggered flows can still be scheduled, and the reservations do not consume extra energy.

IV. Fundamental methods for scheduling

Compared to traditional time-triggered networks, the time-triggered and event-triggered hybrid networks have three new factors that affect the schedulability: periodicity, time slot reservation and node reservation for event-triggered flows. First, if schedules are not periodic, or the period is not short enough to satisfy the resource constraint, the network cannot be scheduled even though a feasible solution has been generated. Second, the more time slots and nodes that are reserved for event-triggered flows, the harder the scheduling of time-triggered flows becomes. In this section, we propose three methods and analyze them in terms of schedule length and resource requirements.

A. Virtual-Period Method

Although the real-time scheduling for event-triggered flows is still an open problem, there are many studies on time-triggered flows. An intuitive scheduling approach for event-triggered flows is to convert them to virtual time-triggered flows and then schedule them using the same algorithms with other time-triggered flows, e.g. RM [16] and C-LLF [7]. The only difference between event-triggered flows and time-triggered flows is that event-triggered flows are not periodic. Therefore, we assign virtual periods p_i^e to event-triggered flows. The calculation of virtual periods is shown in Theorem 1. When an event-triggered packet is released at time slot t, it waits to be scheduled until the subsequent virtual period starts. Theorem 1 proves that this virtual period will have an instance that is released and finishes in the interval $[t, t+d_i^e]$. Therefore, the event-triggered packet will not miss its deadline. Then, we can obtain the superframe length $H^{vp} = \max_{\forall f_i^* \in F}\{p_i^*\}$, as all periods are harmonic. At each time slot in the superframe, if an idle channel exists, the transmission that has the highest priority and does not involve a node that is common with the scheduled flows in this slot is scheduled on that channel.

Theorem 1. *If an event-triggered packet is released at time slot t, and its absolute deadline is $t + d_i^e$, then there must exist a harmonic virtual period $p_i^e = p' \times 2^{\lfloor log_2(\frac{d_i^e+1}{2p'})\rfloor}$ with an instance that is fully contained in interval $[t, t + d_i^e]$.*

Proof. We set the virtual period of flow f_i^e to $p' \times 2^x$ ($x \in \mathbb{Z}$). x should be as large as possible, as in each virtual period, c_i^e time slots have to be occupied. A packet is released at time slot t ($t \in \mathbb{N}$), and its deadline is $t + d_i^e$. Thus, our object is to find the maximum x such that

$$t \le j \times (p' \times 2^x) < (j+1) \times (p' \times 2^x) - 1 \le t + d_i^e,$$

where j can be any integer. We know $\frac{t}{p' \times 2^x} \le j$ and $j \le \frac{t+d_i^e+1}{p' \times 2^x} - 1$. So, in the range of $[\frac{t}{p' \times 2^x}, \frac{t+d_i^e+1}{p' \times 2^x} - 1]$, there must be at least one integer. We discuss two cases as follows.

- If $0 \le (\frac{t+d_i^e+1}{p' \times 2^x} - 1) - (\frac{t}{p' \times 2^x}) < 1$, then $\log_2(\frac{d_i^e+1}{2p'}) < x \le \log_2(\frac{d_i^e+1}{p'})$. The maximum x is $\lfloor \log_2(\frac{d_i^e+1}{p'}) \rfloor$. In this case, there are many values of t, d_i^e and p' that make no integer exist in the range, e.g., $t = 1$ and $d = p' = 4$. Therefore, this case does not hold.
- If $1 \le (\frac{t+d_i^e+1}{p' \times 2^x} - 1) - (\frac{t}{p' \times 2^x})$, there must be an integer in the range regardless of these values. Then, we obtain $x \le \log_2(\frac{d_i^e+1}{2p'})$. Therefore, $p_i^e = p' \times 2^{\lfloor log_2(\frac{d_i^e+1}{2p'})\rfloor}$.

□

For the generated schedules, the resource constraint must be satisfied. The number of working-mode entries that flow f_i^* introduces to n_j is

$$w_{i,j}^*(H^{vp}) = \begin{cases} {}^{H^{vp}}/_{p_i^*} & \text{if } n_j = sn_i^* \text{ or } n_j = dn_i^* \\ 2 \times {}^{H^{vp}}/_{p_i^*} & \text{if } n_j \in \pi_i^* \text{ and } n_j \ne sn_i^* \text{ and } n_j \ne dn_i^* \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

If n_j is the source or destination, it is only used once in one period. Otherwise, the node receives a packet and then sends it; thus, the node is used twice. Then, the number of working-mode entries of node n_j is $w_j^{vp} = \sum_{\forall f_i^* \in F} w_{i,j}^*(H^{vp})$. If $\forall n_j, w_j^{vp} \le W$, the generated schedules are feasible.

Then, for this virtual-period method (VP), we calculate the reservation of time slots and nodes. We consider a certain time interval $[0, y)$, and the packets with deadlines before time slot y must be delivered. Thus, the number of time slots reserved for flow f_i^e is $\lfloor \frac{y}{p_i^e} \rfloor \times c_i^e$. In fact, for event-triggered flow f_i^e, the smallest release interval between two consecutive packets is $d_i^e + 1$. Therefore, in the worst case, only $\lfloor \frac{y}{d_i^e+1} \rfloor \times c_i^e$ time slots are used. Comparing the time slots that are actually used and the reserved time slots, we can obtain that

$$\frac{\lfloor \frac{y}{d_i^e+1} \rfloor \times c_i^e}{\lfloor \frac{y}{p_i^e} \rfloor \times c_i^e} = \frac{\lfloor \frac{y}{d_i^e+1} \rfloor \times c_i^e}{\lfloor \frac{y}{p' \times 2^{\lfloor log_2(\frac{d_i^e+1}{2p'})\rfloor}} \rfloor \times c_i^e} \le \frac{\lfloor \frac{y}{d_i^e+1} \rfloor}{\lfloor \frac{y}{\frac{d_i^e+1}{2}} \rfloor} \le \frac{1}{2}.$$

Thus, the utilization of reserved time slots is not greater than 50%, and at least half of the time slot reservations are wasted. The calculation of node reservations is similar to that of time slot reservations. In each reserved slot, two nodes are reserved. Therefore, the number of node reservations for flow f_i^e is $\lfloor \frac{y}{p_i^e} \rfloor \times c_i^e \times 2$, and less than 50% of reservations are used.

B. Slot-Multiplexed Method

The VP method wastes some reserved time slots. To minimize such waste, we propose a slot-multiplexed method (SM). For flow f_i^e, its time slot reservation is repeated after every d_i^e+1 time slots. We define the d_i^e+1 time slots as a *reservation interval*. In different reservation intervals, the reserved time slots must be at the same places. For example, Fig. 2 shows simple SM scheduling, where the deadline of the event-triggered flow is 4 and the period of the two time-triggered flows is 10. Their routing paths are the same as those in Fig. 1(a). The gray blocks are reserved for the event-triggered flow. They are placed at the first and second time slots of every reservation interval. In each reserved time slot, all nodes in its routing path are reserved for this flow. If a node is passed through by an event-triggered flow, at one reserved time slot, it is assigned to work in two modes Rx and Tx, e.g. node n_0. Its working method is as follows: if the node has not received the packet, it will listen to the channel; if the node has received the packet at a time slot, it will send it at the next reserved time slot. For example, node n_0 receives the packet at time slot 1, and then will send it at time slot 5. In this way, whenever an event-triggered flow f_i^e releases a packet, in its active interval, there must be c_i^e time slots that are reserved for transmitting the packet. We prove this in Theorem 2. Then, we can know that when the objective is to minimize the number of reserved time slots, the feasible solution found by SM is optimal. This is because if the reserved time slots in a reservation interval are less than c_i^e, the packet released at the beginning of the interval is unschedulable.

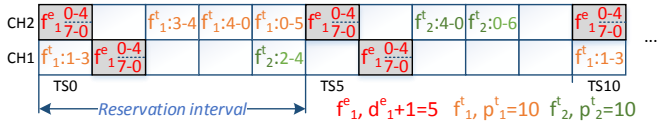


Fig. 2. An illustration of the slot-multiplexed method

Theorem 2. *For the SM method, whenever an event-triggered flow f_i^e releases a packet, in its active interval there must be c_i^e time slots that are reserved to transmit the packet.*

Proof. Suppose that flow f_i^e releases a packet at time slot t. Its active interval is $[t, t+d_i^e]$. When the active interval is a reservation interval, there must be c_i^e time slots. Otherwise, we set that the reservation interval includes t_1 to t, and the ending is t_2. $R(t_1, t-1)$ denotes the number of reserved time slots for f_i^e between t_1 and $t-1$. We know $R(t_1, t-1)+R(t, t_2) = c_i^e$. Since the reserved time slots are at the same places in different intervals, $R(t_1, t-1) = R(t_2+1, t+d_i^e)$. Hence, $R(t_1, t-1)+R(t, t_2) = R(t_2+1, t+d_i^e)+R(t, t_2) = R(t, t+d_i^e) = c_i^e$. □

In SM, the superframe length is $H^{sm} = LCM(d_1^e+1, d_2^e+1, ..., p_1^t, p_2^t, ...)$. If d_i^e+1 is relatively prime with p_j^t, then the superframe length may be very long, making it difficult to satisfy the resource constraint W. Hence, we first check the constraint. For node n_j, the number of working-mode entries introduced by time-triggered flow f_i^t can be calculated based on Eq. (1). If n_j is in the path of event-triggered flow f_i^e, it will be used at each reserved slot. Thus, the number of working-mode entries introduced by f_i^e is calculated as follows.

$$\bar{w}_{i,j}^e(H^{sm}) = \begin{cases} c_i^e \times {}^{H^{sm}}/_{(d_i^e+1)} & \text{if } n_j \in \pi_i^e \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, the number of working-mode entries of n_j is $w_j^{sm} = \sum_{\forall f_i^t \in F^t} w_{i,j}^t(H^{sm}) + \sum_{\forall f_t^e \in F^e} \bar{w}_{i,j}^e(H^{sm})$. If $\forall n_j, w_j^{sm} \le W$, we invoke Algorithm 1 to schedule flows.

Algorithm 1 Time slot and channel assignment in SM

Input: V, F
Output: Schedule S, if feasible

1: $\forall v_k \in V$, $c'_k = 0$;
2: **for** $t = 0$ to $(H^{sm}-1)$ **do**
3: $V' = V$;
4: **while** $idleCH_t \neq \emptyset$ and $V' \neq \emptyset$ **do**
5: find the packet v_k with the earliest absolute deadline in V';
6: $V' = V' - \{v_k\}$;
7: **if** $t >$ the absolute deadline of v_k **then**
8: **return** Unschedulable;
9: **if** v_k belongs to an event-triggered flow f_i^e **then**
10: $A = \{t + q \times (d_i^e+1) | \forall q \in [0, \frac{H^{sm}}{d_i^e+1})\}$;
11: **if** $\forall a \in A, idleCH_a \neq \emptyset$ and $\forall n_j \in \pi_i^e, n_j \notin usedNode_a$ **then**
12: $S = S + \{< f_i^e, a, b > | \forall a \in A, b \in idleCH_a\}$;
13: update $idleCH_a$ and $usedNode_a$;
14: **if** $(++c'_k) == c_i^e$ **then** $V = V - \{v_k\}$;
15: **if** v_k belongs to a time-triggered flow f_i^t **then**
16: **if** $\forall n_j \in \tau^t_{k,c'^t_i}, n_j \notin usedNode_t$ **then**
17: $S = S + \{< \tau^t_{k,c'^t_i}, t, b > | b \in idleCH_t\}$;
18: update $idleCH_t$ and $usedNode_t$;
19: **if** $(++c'_k) == c_i^t$ **then** $V = V - \{v_k\}$;
20: **if** $V == \emptyset$ **then return** S;

Algorithm 1 assigns time slots and channels to the packets in set V. In the beginning, V includes all time-triggered packets that are released in the superframe and the event-triggered packets that are released at slot 0. The other event-triggered packets are excluded because the subsequent assignments are repeated. V' is the set of packets that have not been attempted to be scheduled at the current slot. For each packet $v_k \in V$, c'_k denotes the number of time slots that have been assigned to v_k. $idleCH_a$ and $usedNode_a$ are the set of idle channels and the set of used nodes at time slot a, respectively. Each element $< f_i^e$ (or $\tau^t_{k,g}), TS, CH > \in S$ denotes event-triggered flow f_i^e or the g-th hop of the time-triggered packet v_k that occupies time slot TS and channel CH. This algorithm is based on the EDF (Earliest Deadline First) policy [16], which assigns a higher priority to the packet with an earlier absolute deadline.

At each time slot t, if there exist idle channels in $idleCH_t$ and packets in V' (line 4), then the packet with the highest priority is attempted to be scheduled (line 5). When the selected packet v_k is event-triggered (line 9), we must check whether the t-th time slot in all reservation intervals can be assigned to the packet or not (lines 10–11). If there is no node conflict between the current event-triggered flow and the occupied nodes, then the event-triggered flow can be scheduled at the t-th time slot and on a channel in $idleCH_a$ (line 12). The occupied channel is removed from $idleCH_a$, and the used nodes are added into $usedNode_a$ (line 13). If c_i^e time slots are assigned to f_i^e, its schedule finishes (line 14). For time-triggered packet v_k (lines 15–19), its schedule is similar to event-triggered packets. The difference is that we only check whether the current transmission τ^t_{k,c'^t_i} can use time slot t or not, and do not consider the t-th time slot in the next period. If a packet misses its deadline (line 7), the network is unscheduled (line 8). Otherwise, the result S is returned when all schedules are completed (line 20).

The time complexity of this algorithm is $O(|V| \cdot H^{sm})$. In a certain interval $[0, y)$, the number of time slots reserved for flow f_t^e is $\lfloor \frac{y}{d_i^e+1} \rfloor \times c_i^e$, and the number of node reservations is $\lfloor \frac{y}{d_i^e+1} \rfloor \times c_i^e \times (c_i^e + 1)$ because in each time slot all nodes in its routing path are reserved.

C. Reverse-Scheduling Method

The SM method minimizes the number of reserved time slots, but all nodes in routing paths have to be occupied in these reserved time slots. We propose a reverse-scheduling method (RS) to reduce both time slot and node reservations. In this method, a reserved time slot is allowed to be used by only one transmission unless no other time slot is available for another packet of the same flow. The time slots reserved for transmitting one packet are called an *arrangement* of time slots. We assume that if a packet is released at a time slot, it is ready to be transmitted in that slot. An arrangement of time slots can provide service for the packets released before or at the first time slot of the arrangement, and their deadlines are not later than the last slot of the arrangement. Fig. 3 shows the schedules generated by RS for the same flow set considered in Fig. 2. The packets released between $TS0$ and $TS3$ can use the first arrangement. However, the packet released at $TS4$ misses the first hop at $TS3$. It has to wait for the next arrangement to be scheduled, and the last time slot of the next arrangement cannot be after its deadline. In this way, although event-triggered packets can be released at any time slot, only part of them need to be scheduled, e.g., the packets released at $TS0$ and $TS4$, and we call them *critical* packets. The others can reuse the reserved time slots. However, these reserved slots are still dedicated since two consecutive packets of the same flow cannot exist simultaneously.

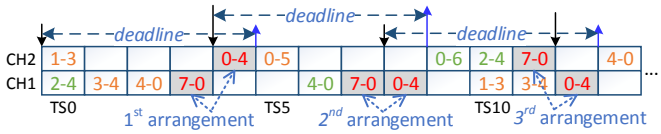


Fig. 3. An illustration of the RS method

For each event-triggered flow f_i^e, the release time of the first critical packet is $r_{i,1} = 0$, and its deadline is $d_{i,1} = d_i^e$. We define the end-to-end latency $l_{i,j}$ of the j-th critical packet as the time between its first reserved time slot and its deadline. Thus, the release time of the second critical packet is $r_{i,2} = d_{i,1} - l_{i,1} + 1$, and the deadline is $d_{i,2} = r_{i,2} + d_i^e$. Therefore, the release time and deadline of the k-th critical packet are

$$r_{i,k} = (k-1) \times d_i^e + (k-1) - \sum_{\forall j \in [1,k)} l_{i,j} \tag{2}$$

$$d_{i,k} = k \times d_i^e + (k-1) - \sum_{\forall j \in [1,k)} l_{i,j}. \tag{3}$$

In a certain interval $[0, y)$, we need to reserve k arrangements of time slots for flow f_i^e, i.e., $d_{i,k} \leq y \leq d_{i,(k+1)}$. From Eq. (2) and (3), we know that if we want to decrease the number of reserved arrangements, the latency $\sum_{\forall j \in [1,k)} l_{i,j}$ should be reduced. Therefore, RS reserves time slots for critical event-triggered packets from their deadline to their release time. The basic idea of RS, called *BasicRS*, is as follows:

Step (1): The time-triggered packets released at time slot 0 and the first critical packets of all event-triggered flows are included in packet set V.

Step (2): The packet with the highest EDF priority in set V is selected, breaking ties by prioritizing the flow with the smaller ID. Then, if the packet is time-triggered, the method finds c_i^t available time slots from its release time to its deadline; if the packet is event-triggered, the method finds c_i^e available time slots from its deadline to its release time.

Step (3): After a packet is scheduled, the next packet of the same flow is added to V. Then, the execution jumps to Step(2).

The method should run until one superframe H^{rs} is finished. However, we cannot calculate the superframe length H^{rs} using the same approaches as H^{vp} and H^{sm}. Critical packets are aperiodic. The release time of a critical packet is determined by the reservations of its previous critical packet, and the reservations are unknown before the previous critical packet is scheduled successfully. Therefore, for RS, only when the schedules in one superframe are generated, can we know the superframe length. In VP and SM, the schedules are executed repeatedly between the first and last time slots of a superframe. Similarly, in RS, there also exist two special time slots α and β $(\alpha < \beta)$. They meet the following conditions: (1) the schedules at time slots $\alpha + g$ and $\beta + g$ $(0 < g)$ are the same; (2) the schedules at time slots $\alpha - g$ and $\beta - g$ $(0 \leq g)$ are different. Thus, the network first executes the schedules from $TS0$ to $TS\beta$, then jumps to $TS\alpha$ and repeats the schedules between $TS\alpha$ and $TS\beta$. Therefore, β corresponds to the superframe length. In Theorem 3, we analyze its upper bound.

Theorem 3. *When the BasicRS method is used to schedule a network, the superframe length is not greater than* $\Pi_{\forall f_i^e} \Pi_{\forall \eta_i \in [1, d_i^e+1]} \min\{\eta_i + 1, c_i^e + 1\} \times H^{'rs}$, *where* $H^{'rs} = \max_{\forall f_i^t \in F^t} \{p_i^t\}$.

Proof. We use $\xi_{i,t}^t =< \epsilon, \eta >$ and $\xi_{i,t}^e =< \epsilon_1, \eta_1, \epsilon_2, \eta_2, ... >$ to describe the status of flow f_i^t and flow f_i^e at time slot t, respectively. For a time-triggered flow, $\xi_{i,t}^t$ means that at the beginning of time slot t, its packet still has ϵ hops to be transmitted in the following η time slots. For an event-triggered flow f_i^e at one time slot, there are probably multiple critical packets to be scheduled. The j-th packet of these corresponds to (ϵ_j, η_j). We use $\xi_t =< \xi_{1,t}^t, \xi_{2,t}^t, ..., \xi_{1,t}^e, \xi_{2,t}^e, ... >$ to denote the network status at time slot t. At two time slots α and β, if $\xi_\alpha = \xi_\beta$, then the schedules after time slot β are the same as those after time slot α. Additionally, η reflects the deadline of the first packet and the starting offset of the following packets, and ϵ and c_i^* represent the execution time of the first packet and following packets, respectively. When the deadlines and execution times of all packets are determinate, based on the EDF policy the schedules must be unique. Hence, the upper bound of the superframe length is at most $\beta - 1$.

We know that $\forall f_i^t, \xi_{i,0}^t = \xi_{i,H'^{rs}}^t = \xi_{i,2H'^{rs}}^t = ... =< c_i^t, p_i^t >$. We assume that there are at most x different statuses for event-triggered flows. Hence, if the network runs for $x \times H^{'rs}$ time slots, then there must be two time slots that have the same status. Therefore, $\beta \leq x \times H^{'rs} + 1$, and the upper bound of the superframe length is not greater than $x \times H^{'rs}$. Then,

we calculate x. We know that for flow f_i^e, $\eta_j \in [1, d_i^e + 1]$. When η_j takes a value in this range, ϵ_j probably takes $\varepsilon(\eta_j)$ values, where $\varepsilon(\eta_j) = \min\{\eta_j + 1, c_i^e + 1\}$. That is because when $\eta_j \in [1, c_i^e]$ and the network is schedulable, ϵ_j cannot be greater than η_j. If $\epsilon_j > \eta_j$, then there are not enough time slots to transmit the remaining ϵ_j hops. Hence, ϵ_j can be any value in $\{0, 1, ..., \eta_j\}$, i.e., $\eta_j + 1$ values. When η_j is greater than c_i^e, ϵ_j can be a value in $\{0, 1, ..., c_i^e\}$, i.e., $c_i^e + 1$ values. In the extreme case, at one time slot, $d_i^e + 1$ critical packets that belong to the same flow need to be scheduled, and their η_j differ from each other by one time slot. Thus, $\xi_{i,t}^e =< \epsilon_1, 1, \epsilon_2, 2, ..., \epsilon_{d_i^e+1}, d_i^e + 1 >$. Therefore, there are at most $\Pi_{\forall \eta_i \in [1, d_i^e+1]} \min\{\eta_i + 1, c_i^e + 1\}$ statuses at one time slot for an event-triggered flow f_i^e. All event-triggered flows are independent of each other. Thus, when we consider all of them, there are at most $\Pi_{\forall f_i^e} \Pi_{\forall \eta_i \in [1, d_i^e+1]} \min\{\eta_i + 1, c_i^e + 1\}$ statuses. Therefore, the upper bound of the superframe length is $\Pi_{\forall f_i^e} \Pi_{\forall \eta_i \in [1, d_i^e+1]} \min\{\eta_i + 1, c_i^e + 1\} \times H^{'rs}$. □

Algorithm 2 The reverse-scheduling method

Input: F
Output: S, α, β

1: **for** $(\beta = (H^{'rs} - 1);\ ; \beta += H^{'rs})$ **do**
2: $S = S + BasicRS(\beta - (H^{'rs} - 1), \beta)$;
3: **if** $\exists n_j, w_j^{rs} > W$ **then**
4: **return** Unschedulable;
5: **for** $(\alpha = 0; \alpha < \beta; \alpha += H^{'rs})$ **do**
6: $V' = Check(S, \alpha, \beta)$;
7: **if** $Schedulable(S, \alpha, V')$ **then**
8: **return** S, α and β;

Method $Check(S, \alpha, \beta)$

Input: S, α, β
Output: Unschedulable packet set V'

1: $V' =$ the event-triggered packets that have not finished at time slot β;
2: **for** $\forall v_k \in V'$ (v_k belongs to f_i^e) **do**
3: **if** $\exists (\epsilon_a, \eta_a) \in \xi_{i,\alpha}^e, \epsilon_k \le \epsilon_a, \eta_a \le \eta_k$ **then**
4: $V' = V' - \{v_k\}$;
5: **return** V';

In the worst case, nodes need a huge local memory to store schedule information of such a long superframe. Therefore, we propose Algorithm 2 to construct a short superframe such that the resource constraint can be satisfied. In Algorithm 2, we first use Function *BasicRS(starting TS, ending TS)* to generate schedules in each $H^{'rs}$ (line 2). When a one-$H^{'rs}$ schedule is generated, we check if all nodes can satisfy the resource constraint (line 3). The number of working-mode entries w_j^{rs} can be counted based on the generated schedule S. If any node does not satisfy the resource constraint, then the network is unschedulable (line 4). Otherwise, we try to find the starting time slot α of periodic schedules (line 5). Function *Check()* checks if there are unschedulable event-triggered packets when the schedules are repeated between α and β (line 6). In *Check()*, set V' includes the event-triggered packets that have not finished by $TS\beta$ (line 1 of *Check()*). For each packet in V', if there are enough reserved time slots after $TS\alpha$ (lines 2–3 of *Check()*), then the packet can be scheduled. Otherwise, it is re-scheduled from $TS\alpha$ (line 7), and the occupied time slots by the packets in V' can be re-used. If all packets in V' are scheduled successfully, the schedules between the current α and β can be executed repeatedly (line 8), and β is the superframe length H^{rs}.

The time complexity of *Check()* is $O(|V|^2)$. For Algorithm 2, the number of iteration of **for** loop in line 1 is $O(\frac{H^{rs}}{H^{'rs}})$, where H^{rs} is limited by the resource constraint W. The time complexities of lines 2, 5 and 7 are $O(H^{'rs}|V|)$, $O(\frac{H^{rs}}{H^{'rs}})$ and $O(H^{'rs}|V|)$, respectively. Therefore, the time complexity of RS is $O((\frac{H^{rs}}{H^{'rs}})^2 H^{rs}|V|)$.

In a certain interval $[0, y)$, the number of time slots reserved for flow f_i^e is not less than $\lfloor \frac{y}{(d_i^e+1)-(c_i^e-1)} \rfloor \times c_i^e$ because in the best case, the latency is c_i^e, and then, the difference between the release times of two sequence critical packets is $(d_i^e + 1) - (c_i^e - 1)$. Similarly, the number of node reservations is not less than $\lfloor \frac{y}{(d_i^e+1)-(c_i^e-1)} \rfloor \times c_i^e \times 2$.

V. Comparison

A relative comparison of the three methods in terms of schedule length and resource requirements is summarized in Table I. As the table shows, the simplest method VP has the shortest superframe, while more than half of reserved communication resources are wasted. Therefore, if communication resources are ample, VP is the best choice. The SM method reserves the minimum number of time slots. However, its node reservation requirement is very high and the superframe length may be very long when the parameters in $LCM()$ are relatively prime. Hence, if only a small number of nodes are occupied and $d_i^e + 1$ is not relatively prime with the others, then SM becomes an effective approach. For the RS method, we cannot calculate its superframe length. However, its node reservation may be the least, and, unlike VP, it does not waste resources. Therefore, RS becomes an effective approach when its schedules can be constructed. Comparing the three algorithms, we know that VP is the simplest; SM has the best time-sensitive performance; RS is the most flexible. Thus the three methods have different advantages. Therefore, event-triggered flows should be scheduled by different algorithms based on their features. In the next section, we propose an algorithm that combines the advantages of the three methods to solve the scheduling problem.

VI. A Combined Algorithm

Considering $|F^e|$ event-triggered flows and 3 methods, there are $3^{|F^e|}$ possible combinations for scheduling. When the number of flows is very small, the execution time to explore all combinations may be acceptable. However, for most cases, we need a fast algorithm to find a feasible solution. Our proposed combined algorithm (CA) is shown in Algorithm 3. In the beginning of CA, all event-triggered flows are scheduled by VP because if there are enough communication resources, VP is the fastest and simplest. Then, the combination is checked to verify the schedulability. If it is not schedulable, we adjust flows to be scheduled by SM or RS until a feasible solution is found.

The assignment process is as follows. Set F^{vp} (F^{sm} or F^{rs}) includes the flows that are scheduled by VP (SM or RS). Then, $F^{vp} \cap F^{sm} \cap F^{rs} = \emptyset$ and $F^{vp} \cup F^{sm} \cup F^{rs} = F^e$. First, all event-triggered flows are in F^{vp} (line 1). Then, we remove a flow from F^{vp} to one of the other two sets. To reduce the resource waste, the removed flow f_i^e has the largest resource

TABLE I
COMPARISON AMONG THREE METHODS

Method	Periodicity (superframe length)	Node reservation	Time slot reservation
VP Method	$= \max_{\forall f^* \in F}\{p_i^*\}$	$= \sum_{\forall f_i^e} \lfloor \frac{y}{p_i^e} \rfloor \times c_i^e \times 2$	$= \sum_{\forall f_i^e} \lfloor \frac{y}{p_i^e} \rfloor \times c_i^e$
SM Method	$= LCM(d_1^e+1, d_2^e+1, ..., p_1^t, p_2^t, ...)$	$= \sum_{\forall f_i^e} \lfloor \frac{y}{d_i^e+1} \rfloor \times c_i^e \times (c_i^e+1)$	$= \sum_{\forall f_i^e} \lfloor \frac{y}{d_i^e+1} \rfloor \times c_i^e$
RS Method	$\leq \Pi_{\forall f_i^e} \Pi_{\forall \eta_i \in [1, d_i^e+1]} \min\{\eta_i+1, c_i^e+1\} \times H^{'rs}$	$\geq \sum_{\forall f_i^e} \lfloor \frac{y}{d_i^e+2-c_i^e} \rfloor \times c_i^e \times 2$	$\geq \sum_{\forall f_i^e} \lfloor \frac{y}{d_i^e+2-c_i^e} \rfloor \times c_i^e$

utilization $\frac{c_i^e}{d_i^e+1}$ (line 8). If the period of f_i^e is not relatively prime with those of other time-triggered flows and the removal does not add node reservation (line 9), it is added to F^{sm} (line 10). Otherwise, it is added to F^{rs} (line 12). Function *AddRes()* checks the addition of node reservation. If $c_i^e \times (c_i^e+1) > \lfloor \frac{d_i^e+1}{p_i^e} \rfloor \times c_i^e \times 2$, then *AddRes()* returns true. If all flows in F^{vp} have been moved out and the network is still unschedulable, then the flow having the maximum node reservations in F^{sm} is removed to F^{rs} until $F^{sm} = \emptyset$ (lines 13–15). Finally, all event-triggered flows are in F^{rs}. Considering both time slot reservations and node reservations, RS may have the fewest reservations. It is opposite to VP. Therefore, the adjustment is to remove flows from F^{vp} to F^{rs}.

Algorithm 3 The combined algorithm

Input: F
Output: Schedule S, if feasible
1: $F^{vp} = F^e$; $F^{sm} = F^{rs} = \emptyset$;
2: **while** true **do**
3: $F^t = F^t + VP(F^{vp})$;
4: **if** all necessary conditions are satisfied **then**
5: **if** $RS(F, F^{sm})$ returns S **then**
6: **return** S;
7: **if** $F^{vp} \neq \emptyset$ **then**
8: find a flow f_i^e with the largest $\frac{c_i^e}{d_i^e+1}$ in F^{vp};
9: **if** $(\forall f_j^t, (d_i^e+1) \not| p_j^t$ or $(d_i^e+1) = p_j^t)$ and $!AddRes(f_i^e)$ **then**
10: $F^{sm} = F^{sm} + \{f_i^e\}$; $F^{vp} = F^{vp} - \{f_i^e\}$;
11: **else**
12: $F^{rs} = F^{rs} + \{f_i^e\}$; $F^{vp} = F^{vp} - \{f_i^e\}$;
13: **else if** $F^{sm} \neq \emptyset$ **then**
14: find a flow f_i^e with the most node reservations in F^{sm};
15: $F^{rs} = F^{rs} + \{f_i^e\}$; $F^{sm} = F^{sm} - \{f_i^e\}$;
16: **else** break;
17: **return** Unschedulable;

For each combination of F^{vp}, F^{sm} and F^{rs}, we use the following algorithm to schedule them. First, the flows in F^{vp} are changed to virtual-period flows (line 3). Then, we propose three necessary conditions as follows. If set F does not satisfy these conditions, it must be unschedulable. Thus, the execution time can be reduced effectively.

Condition 1: the utilization of each node is not greater than 1, i.e., $\forall n_j \in N$,

$$\sum_{\forall f_i^t \in F^t} \frac{\delta_{i,j}}{p_i^t} + \sum_{\forall f_i^e \in F^{sm}} \frac{\varepsilon_{i,j} \times c_i^e}{d_i^e+1} + \sum_{\forall f_i^e \in F^{rs}} \frac{\delta_{i,j}}{d_i^e+2-c_i^e} \leq 1,$$

where $\delta_{i,j} = 1$ if n_j is the source or destination of flow f_i^t and $\delta_{i,j} = 2$, if f_i^t passes through n_j; otherwise, $\delta_{i,j} = 0$. If f_i^e uses n_j, $\varepsilon_{i,j} = 1$; otherwise, $\varepsilon_{i,j} = 0$.

Condition 2: the network utilization is not greater than the number of channels m, i.e.,

$$\sum_{\forall f_i^t \in F^t} \frac{c_i^t}{p_i^t} + \sum_{\forall f_i^e \in F^{sm}} \frac{c_i^e}{d_i^e+1} + \sum_{\forall f_i^e \in F^{rs}} \frac{c_i^e}{d_i^e+2-c_i^e} \leq m.$$

Condition 3: the lower bound of the number of working-mode entries is not greater than W. For the RS method, the shortest superframe is $H^{'rs}$. F^{vp} is added into F^t. F^{sm} does not increase the superframe length. Thus, the lower bound of the superframe length is $H^{'rs}$. Therefore,

$$\sum_{\forall f_i^t \in F^t} w_{i,j}^t(H^{'rs}) + \sum_{\forall f_i^e \in F^{sm}} \bar{w}_{i,j}^e(H^{'rs}) + \sum_{\forall f_i^e \in F^{rs}} \hat{w}_{i,j}^e(H^{'rs}) \leq W,$$

where the lower bound of RS is as follow.

$$\hat{w}_{i,j}^e(H^{'rs}) = \begin{cases} {H^{'rs}}/{d_i^e+2-c_i^e} & \text{if } n_j = sn_i^e \text{ or } n_j = dn_i^e \\ 2 \times {H^{'rs}}/{d_i^e+2-c_i^e} & \text{if } n_j \in \pi_i^e \text{ and } n_j \neq sn_i^e \text{ and } n_j \neq dn_i^e \\ 0 & \text{otherwise.} \end{cases}$$

In Function *RS()*, the flows in F^{sm} need to reserve all nodes on their paths. This is the only difference from Algorithm 2. Thus, the time complexity of *RS()* is $O(|N|(\frac{H^{rs}}{H'^{rs}})^2 H^{rs}|V|)$. The number of iterations of **while** loop in line 2 is $O(|F|)$. Therefore, the time complexity of Algorithm 3 is $O(|F||N|(\frac{H^{rs}}{H'^{rs}})^2 H^{rs}|V|)$.

VII. EVALUATION

In this section, we will evaluate our algorithm based on the topologies of a physical WSAN testbed and random topologies. Two metrics are used for performance evaluation: (1) *schedulable ratio* is the percentage of test cases for which an algorithm is able to find a feasible schedule, and (2) *execution time* is the time required to generate a feasible schedule. We compare our algorithm CA and the three fundamental methods VP, SM and RS with three methods SS [13], EDF and UP. UP shows the percentage of cases that satisfy the three necessary conditions, thereby showing a conservative upper bound of schedulable ratio that an optimal algorithm can achieve. The original SS method allows time-triggered packets to be discarded when event-triggered packets are using shared time slots. To make it suitable for our problem, SS does not assign shared time slots. The pure EDF policy is an effective scheduling algorithm for time-triggered flows. In our evaluation, the EDF method schedules not only time-triggered packets but also all possible event-triggered packets. If two event-triggered packets belong to the same flow, EDF can assign the same time slots to them.

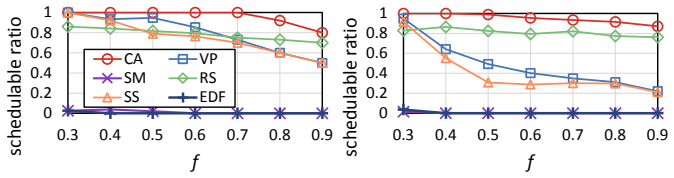
All algorithms are written in C and run on a Windows machine with 3.4GHz CPU and 16GB memory. The parameters used in this section are summarized in Table II. The number of flows is $\lceil \frac{1}{2} \times n \times f \rceil$, and there are $\lceil \frac{1}{2} \times n \times f \times e \rceil$ event-triggered flows. Source nodes and destination nodes are randomly selected, and are not repeatedly used. Random topologies are generated based on the node density ρ. The harmonic periods are randomly selected in $\{10 \times 2^i | i \in [1, 10]\}$, and the deadlines of event-triggered flows are in $\{10 \times i | i \in [2, 2^{10}]\}$. We use Conditions 1 and 2 to calculate the utilization of the access point and network. The access point is the hotspot, and its utilization is larger than network utilization and is a key factor that affects the schedulability. The schedule information includes source, destination, Rx/Tx, TS and CH. Rx/Tx and CH can be stored in an unsigned char. Hence, a working-mode entry needs 5 bytes. We set the memory size to 50kB and 100kB. Their corresponding W are 10240 and 20480, respectively.

TABLE II
PARAMETERS

n	Number of nodes
m	Number of channels
ρ	Node density of a network
f	Fraction of sources and destinations
e	Fraction of event-triggered flows
u	Utilization of the access point
W	Upper bound of the number of working-mode entries

A. Real Topologies

We use the topologies of an indoor testbed [20] deployed in Jolley Hall of Washington University in St. Louis. The testbed consists of 70 TelosB motes, each equipped with CC2420 radios and compliant with the IEEE 802.15.4e standard. All nodes take turns broadcasting packets. If the packet reception ratio between two nodes is higher than 80%, a reliable link is considered between them. Thus, we obtain two real topologies at transmission power levels $0dBm$ and $-5dBm$.

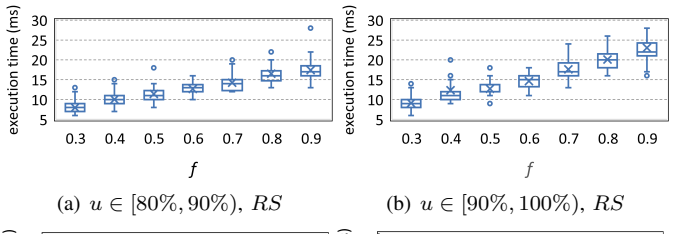


(a) $u \in [60\%, 70\%)$ (b) $u \in [70\%, 80\%)$

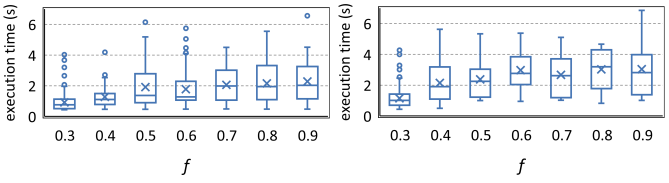
(c) $u \in [80\%, 90\%)$ (d) $u \in [90\%, 100\%)$

Fig. 4. Schedulable ratio with a topology of 0dBm

Fig. 4 shows the comparison of schedulable ratios when TX power is $0dBm$. The parameters are $f \in [0.2, 0.9], n = 70, m = 6, e = 0.2$ and $W = 10240$. For each f, 10000 test cases are randomly generated. According to the utilization u, these test cases are presented in different sub-figures. The results are normalized with UP as the baseline. As u and f increase, the schedulable ratios decrease because it is difficult to find a feasible solution when the solution space becomes more and more complex. Among all methods, CA has the highest schedulable ratio. When $u > 90\%$, CA can still solve more than half of test cases. VP and SS are similar. In the SS method, event-triggered packets are considered more important than time-triggered packets. After scheduling all event-triggered packets, time-triggered packets start to be scheduled. SS pays more attention to the criticality, but not the temporality. VP considers only temporality. Therefore, it has better schedulability than SS. For the SM method, its superframe length is very long, and each reserved time slot needs multiple working-mode entries to describe. The local memory of a TelosB mote is not sufficient to store all these entries. Only when f is very small can a small part of test cases be scheduled. Although SM reserves the minimum number of time slots, using SM alone may not be practical. EDF has to handle all possible event-triggered packets. High workload leads to low schedulable ratio. In the following, the simulation results will not include SM and EDF.

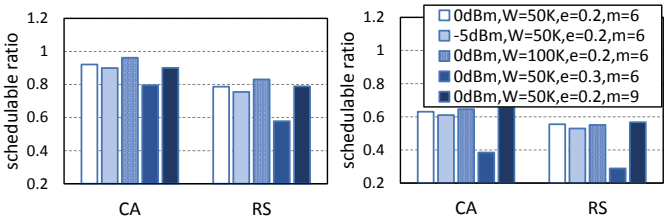


(a) $u \in [80\%, 90\%)$, RS (b) $u \in [90\%, 100\%)$, RS



(c) $u \in [80\%, 90\%)$, CA (d) $u \in [90\%, 100\%)$, CA

Fig. 5. Execution time



(a) $u \in [80\%, 90\%)$ (b) $u \in [90\%, 100\%)$

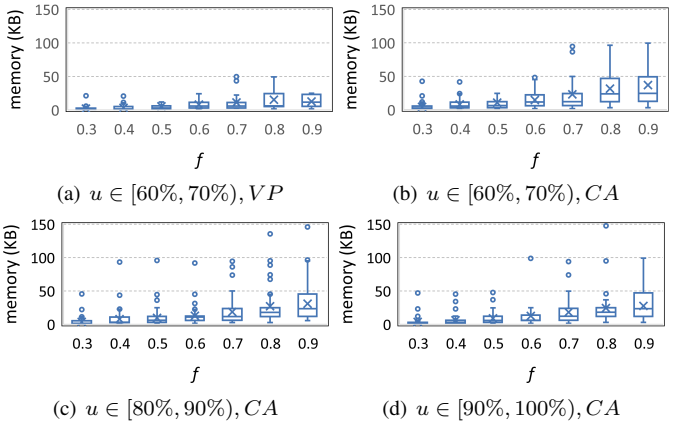
Fig. 6. Schedulable ratio under varying parameters

Fig. 5 shows the execution times of RS and CA. The corresponding test cases are the same as Fig. 4. VP and SS can find a feasible solution within 2 ms. As the number of flows and the access point utilization increase, the execution time of RS increases because there are more transmissions to be dealt with. CA repeatedly executes to find a feasible solution. Therefore, the impact of u and f on CA is not obvious. For all test cases, the execution time of CA is less than 7s. This is acceptable to handle dynamic re-deployments of industrial applications.

Fig. 6 shows the comparison of schedulable ratios under varying parameters. A bar corresponds to the average of a curve from $f = 0.3$ to 0.9. From these figures, we find that:

- The decrease of power level leads to a slightly decrease of schedulable ratios because when the transmission power reduces, the number of hops increases.
- When the local memory doubles, the schedulable ratio increases by 4% and 1% in the two sub-figures. Fig. 7 shows the memory size required by CA and VP under varying u. When $u > 70\%$, VP can hardly find feasible solutions. Hence, there is no corresponding subfigure. Memory needed by CA is about four times that by VP. CA trades memory for schedulability. 50kB memory is sufficient for most test cases.
- If the number of channels increases, then more resources can be used, and the performance should be improved. However, the increase of m has almost no impact on the schedulable ratio. The reason is that regardless of how many channels are used, the utilization u is unchanged.
- When the fraction of event-triggered flows increases from 0.2 to 0.3, the schedulable ratio obviously decreases. Each combination of these parameters corresponds to two sub-figures that are similar to Fig. 4(c) and (d). We do not show

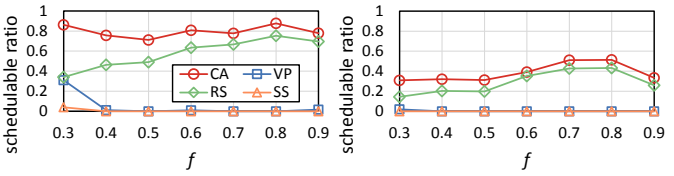
those similar results. Only the parameter combination with $e = 0.3$ has a difference that is shown in Fig. 8. When $f = 0.3$ and $u \in [80\%, 90\%)$, the schedulable ratio of RS is very low. When u is fixed, the fewer flows, the more utilization each flow has. Hence, their periods and deadlines are short. These lead to more conflicts and long latencies. In this case, RS is difficult to construct the periodicity. CA changes some event-triggered flows to time-triggered flows. Although the change makes the flows consume more resources, the consumption is acceptable when $u \in [80\%, 90\%)$. In Fig. 8(b), the schedule ratio of CA sharply reduces because when $u > 90\%$, there is almost no redundant resources.



(a) $u \in [60\%, 70\%), VP$ (b) $u \in [60\%, 70\%), CA$

(c) $u \in [80\%, 90\%), CA$ (d) $u \in [90\%, 100\%), CA$

Fig. 7. Comparison of memory size



(a) $u \in [80\%, 90\%)$ (b) $u \in [90\%, 100\%)$

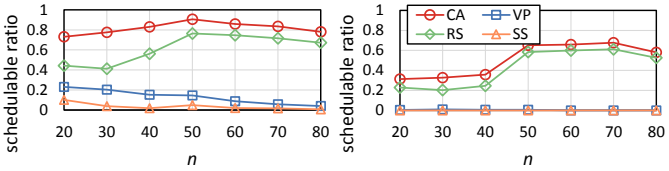
Fig. 8. Schedulable ratio when $e = 0.3$

B. Random Topologies

To evaluate the impact of n and ρ on schedule ratios, we randomly generate topologies based on the equation $A = \frac{nd^2\sqrt{27}}{2\pi\rho}$ [21], where A is a square area, and d denotes the transmitting range of 40 m. A gateway is deployed at the center of the square A, and $n - 1$ nodes are randomly deployed in A. If the distance of two nodes is less than d, there is a reliable link between them. Fig. 9 shows the schedulable ratio under varying n. The other parameters are $m = 6$, $\rho = 1$, $f = 0.8$, $e = 0.2$ and $W = 10240$. Our combined algorithm CA outperforms the others. When u is fixed, the higher the number of nodes, the lesser the node conflicts. Thus, as n increases, the schedulable ratio of CA and RS increases. Fig. 10 shows the schedulable ratio under varying ρ. When $u \in [80\%, 90\%)$, as the increase of ρ, the ratios slightly increase because the number of available relay nodes increases, and the number of hops decreases. When $u \in [90\%, 100\%)$, the test cases are hard to be scheduled regardless of the value of ρ.

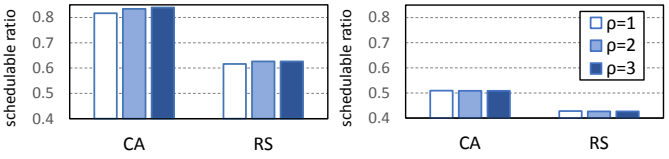
VIII. Conclusion

In this paper, we focus on the real-time scheduling problem for time-triggered and event-triggered hybrid networks. In existing approaches, time-triggered packets are dropped when event-triggered packets are transmitted. Our algorithms do not



(a) $u \in [80\%, 90\%)$ (b) $u \in [90\%, 100\%)$

Fig. 9. Schedulable ratio under varying n



(a) $u \in [80\%, 90\%)$ (b) $u \in [90\%, 100\%)$

Fig. 10. Schedulable ratio under varying ρ

drop time-triggered packets and schedule all time- and event-triggered packets under real-time constraints. Our proposed combined algorithm reserves as few time slots as possible and constructs feasible superframes. The simulations indicate that our combined algorithm significantly outperforms the others. In the future, we will study distributed scheduling algorithms where each node dynamically generates schedules based on the behaviors of event-triggered flows. We shall also consider preemptive scheduling where event-triggered flows are allowed to preempt low-critical time-triggered flows under system stability constraints.

References

[1] D. Chen, M. Nixon, and M. Aloysius, *WirelessHART^{TM}–Real-time mesh network for industrial automation*. Springer, 2010.

[2] P. Park, S. C. Ergen, C. Fischione, C. Lu, and K. H. Johansson, "Wireless network design for control systems: A survey," *IEEE Commun. Surv. & Tutor.*, 2017.

[3] IEC, "Iec 62591: Industrial communication networks–wireless communication network and communication profiles–wirelesshart," 2009.

[4] X. Zhu, X. Tao, T. Gu, and J. Lu, "Target-aware, transmission power-adaptive, and collision-free data dissemination in wireless sensor networks," *IEEE Trans. Wirel. Commun.*, 2015.

[5] D. Yang, J. Ma, Y. Xu, and M. Gidlund, "Safe-wirelesshart: A novel framework enabling safety-critical applications over industrial wsns," *IEEE Trans. Ind. Inform.*, 2018.

[6] R. T. Hermeto, A. Gallais, and F. Theoleyre, "Scheduling for ieee802.15.4-tsch and slow channel hopping mac in low power industrial wireless networks: A survey," *Comput. Commun.*, 2017.

[7] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-time scheduling for wirelesshart networks," in *RTSS*. IEEE, 2010.

[8] ——, "Priority assignment for real-time flows in wirelesshart networks," in *ECRTS*. IEEE, 2011.

[9] ——, "End-to-end delay analysis for fixed priority scheduling in wirelesshart networks," in *RTAS*. IEEE, 2011.

[10] 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 Trans. Ind. Electron.*, 2015.

[11] X. Jin, F. Kong, L. Kong, W. Liu, and P. Zeng, "Reliability and temporality optimization for multiple coexisting wirelesshart networks in industrial environments," *IEEE Trans. Ind. Electron.*, 2017.

[12] W.-B. Pöttner, H. Seidel, J. Brown, U. Roedig, and L. Wolf, "Constructing schedules for time-critical data delivery in wireless sensor networks," *ACM Trans. Sen. Netw.*, 2014.

[13] B. Li, L. Nie, C. Wu, H. Gonzalez, and C. Lu, "Incorporating emergency alarms in reliable wireless process control," in *ICCPS*. ACM, 2015.

[14] X. Jin, F. Kong, L. Kong, H. Wang, C. Xia, P. Zeng, and Q. Deng, "A hierarchical data transmission framework for industrial wireless sensor and actuator networks," *IEEE Trans. Ind. Inform.*, 2017.

[15] C. Xia, X. Jin, L. Kong, and P. Zeng, "Scheduling for emergency tasks in industrial wireless sensor networks," *Sensors*, 2017.

[16] J. W. Liu, *Real-time systems*. Prentice hall, 2000.

[17] W. Rehan, S. Fischer, M. Rehan, and M. H. Rehmani, "A comprehensive survey on multichannel routing in wireless sensor networks," *J. Netw. Comput. Appl.*, 2017.

[18] M. Nobre, I. Silva, and L. A. Guedes, "Routing and scheduling algorithms for wirelesshartnetworks: a survey," *Sensors*, 2015.

[19] D. De Guglielmo, S. Brienza, and G. Anastasi, "Ieee 802.15. 4e: A survey," *Comput. Commun.*, 2016.

[20] (2018) Wustl wireless sensor network testbed. [Online]. Available: http://cps.cse.wustl.edu/index.php/Testbed

[21] T. Camilo, J. S. Silva, A. Rodrigues, and F. Boavida, "Gensen: A topology generator for real wireless sensor networks deployment," in *SEUS*. Springer, 2007.