Research Article

Transmission Scheduling of Periodic Real-Time Traffic in IEEE 802.15.4e TSCH-Based Industrial Mesh Networks

Ke Shi,1 Lin Zhang,1 Zhiying Qi,1 Kang Tong,1 and Hongsheng Chen2

1School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
2College of Computer Science and Technology, Hubei University of Science and Technology, Xianning 437005, China

Correspondence should be addressed to Ke Shi; keshi@mail.hust.edu.cn

Received 29 March 2019; Revised 1 July 2019; Accepted 5 September 2019; Published 22 September 2019

Academic Editor: Olivier Berder

Copyright © 2019 Ke Shi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Time-slotted channel hopping (TSCH) is a part of an emerging IEEE 802.15.4e standard to enable deterministic low-power mesh networking, which offers high reliability and low latency for wireless industrial applications. Nonetheless, the standard only provides a framework, but it does not mandate a specific scheduling mechanism for time and frequency slot allocation. This paper focuses on a centralized scheme to schedule multiple concurrent periodic real-time flows in TSCH networks with mesh topology. In our scheme, each flow is assigned a dynamic priority based on its deadline and the hops remaining to reach the destination. A maximum matching algorithm is utilized to find conflict-free links, which provides more chances to transfer high-priority flows at each time slot. Frequency allocation is implemented by graph coloring to make finally selected links interference free. Simulation results show that our algorithm clearly outperforms the existing algorithms on the deadline satisfaction ratio with a similar radio duty cycle.

1. Introduction

Low-power wireless networks have been regarded as a key enabler for the Internet of things (IoT). Smart objects such as sensors, mobile devices, and actuators are deployed, interconnected, and used for sensing and controlling of physical processes. IoT has multitudes of applications in industry. For example, it can facilitate the production flow in a manufacturing plant, as IoT devices automatically monitor development cycles and manage warehouses as well as inventories.

However, most industrial applications have strict requirements, especially concerning the reliability and the latency [1]. A low-power wireless solution based on the IEEE 802.15.4 standard, whilst being cheaper, may not be able to deliver the desired low-latency and high reliability requirements due to its CSMA-CA-based access method and its vulnerability to multipath fading and interference [2, 3].

IEEE 802.15 Task Group 4e (TG4e) was created in 2008 to redesign the existing IEEE 802.15.4-2006 Medium Access Control (MAC) standard to obtain a low-power multihop MAC better suited to emerging needs of industrial applications. The IEEE 802.15.4e standard has been published in 2012 as an amendment of the IEEE 802.15.4-2011 MAC protocol [4]. The time slotted channel hopping (TSCH) mode is adopted in the standard to facilitate multihop operation and deals well with fading and interference. The core of the TSCH is a medium access technique which uses time synchronization to achieve ultralow-power operation and channel hopping to enable high reliability. This is very different from the “legacy” IEEE 802.15.4 MAC protocol.

In an IEEE 802.15.4e TSCH network, all nodes are synchronized. The time is divided into time slots, and the slot frame consists of a group of time slots. The actions, transmitting, receiving, or sleeping, that occur at each time slot are decided by a schedule. The node repeats the scheduled slot frame [5]. Therefore, scheduling algorithm assigning links to nodes for data transmission is a key element in the TSCH network, which needs to be built carefully and according to the specific requirements of the application.

Despite its importance, the standard defines the mechanisms to execute a communication schedule, but it...
does not define how the schedule is built, updated, and maintained [6]. Several scheduling algorithms have been proposed to solve this problem, which can be divided into two categories, centralized manner [7, 8] and distributed manner [9–12]. In the centralized manner, a central controller has a precise view of the traffic and radio characteristics and allocates a set of transmission opportunities to each radio link. In the distributed manner, nodes negotiate only with their neighbors (i.e., nodes at a maximum hop or Euclidean distance) to decide which channel/time slot to use. It tends to be more robust to changes without making any a priori assumption neither on the radio topology nor on the volume of traffic to transmit. However, the slots are allocated pseudorandomly, and a frame has to be buffered until a slot exists toward the next hop node. They both increase the end-to-end delay and the probability to have a buffer overflow.

For industrial applications, strict performance requirements often lead to relatively stable network topology and fixed data flows, which is more suitable to adopt the centralized scheduling mechanism. There have been several centralized scheduling approaches such as TASA [13, 14] and SA-PSO [15], which allocate the same time slot to duplex-conflict-less nodes and different channels when two interfering nodes use the same time slot. They transform the construction of a duplex-conflict-free and interference-free schedule into an optimization problem, solving it by using graphs or metaheuristic algorithms. Nonetheless, they assume the network has a multipoint-to-one-point traffic and tree-like topology. This assumption is validated in the data gathering scenarios. For many industrial applications involving feedback control, this assumption does not hold true. For example, in wireless sensor and actuator networks, the sensor devices periodically send data to the controllers, and the controllers process data and then deliver the generated results to the actuators to implement the closed control loops.

Our paper focuses on transmitting multiple concurrent and periodic real-time data flows along the predefined routes in a mesh network to implement a control loop. MAC scheduling is our main concern, and routing optimization occurred in the network layer is not in our discussing scope. A central scheduling scheme, scheduling of periodic real-time flow (SPRF), is proposed to make traffic flows between multiple source-destination pairs in TSCH networks meet their deadline imposed by industrial applications.

The main contributions of our paper are summarized as the following three points. Firstly, a dynamic priority assignment mechanism is proposed to prioritize data flows based on their deadlines and the hops remaining to reach the destination, which allows urgent data flows been transferred first. Secondly, maximum matching and graph coloring algorithm are proposed to find duplex-conflict-free and interference-free links at each time slot and allocate these links to transfer high-priority flows preferentially. Thirdly, sufficient simulations have been performed, and the results are evaluated with different metrics, which proves that our approach achieves a significant improvement compared with other methods and the dynamic priority assignment mechanism works better than the fixed priority assignment mechanism.

The rest of the paper is organized as follows: Section 2 discusses related work in IEEE 802.15.4e TSCH scheduling. Section 3 introduces our system model and the problem formulation. Section 4 describes our dynamic priority and graph-based approach that can schedule multiple concurrent data flows to meet their real-time constraints. Section 5 presents the performance evaluation details. Section 6 concludes the paper and points out the direction of future work.

2. Related Work

TSCH utilizes time-division multiple access (TDMA) to avoid the conflicts on MAC layer. TDMA scheduling mechanism has been a hot research topic in the past years. However, the majority of TDMA scheduling algorithms consider single channel networks, which cannot directly be used in multichannel TSCH networks. In recent years, multichannel TDMA scheduling solutions have been proposed in the literature [16, 17, 18]. Nevertheless, most of the existing multichannel scheduling schemes are not suitable for 802.15.4e TSCH networks because they have not been designed for resource-constrained nodes and do not support per frame channel hoping. They are not efficient in terms of channel utilization.

The IEEE 802.15.4e standard defines the mechanism of how the TSCH MAC executes a schedule in the network, and it does not specify how to build an optimized schedule to meet certain performance requirement. To facilitate the use of IEEE 802.15.4e in the application with real-time constraints, several TSCH scheduling mechanisms have recently appeared in the literature.

The traffic aware scheduling algorithm (TASA) [13, 14] aims at finding a link schedule of minimal length, in order to minimize the number of slots needed to send all data. TASA uses an iterative procedure to build the TSCH schedule. During each iteration, TASA selects a certain number of links and accommodates their transmissions at the same time slot (using multiple channel offsets if needed). This process is repeated several times until all the transmissions required by each link of the network have been accommodated. In detail, it allocates a slot in both the matching process and the coloring process. First, links that still have frames to transmit are selected at the corresponding time slot through a matching process. Then, the channel offsets of the links are allocated so that interference does not occur through the matching process.

Efficient centralized scheduling algorithm (ECSA) [19] formulates the scheduling problem as a throughput maximization problem and delay minimization problem. A graph theoretical approach is proposed to solve the throughput maximization problem in a centralized way. The combinatorial properties of the scheduling problem are addressed by providing an equivalent maximum weighted bipartite matching (MWBM) problem to reduce the computational complexity and adopting the Hungarian algorithm with polynomial time complexity.

Meng et al. [20] used the matching rule to resolve conflicts and proposed a matching scheduling algorithm.
(MSA) to allocate time slots in tree topology aiming at shortening the time delay, reducing the energy consumption, and reducing the utilization rate of the channel. They further optimize MSA by only generating arranged links from the leaf node to the sink node. The node that transmits the data must be some leaf nodes; after the current leaf node completes schedule arrangement in accordance with matching rule, they will be deleted, and then the new leaf nodes are scheduled.

However, all the above algorithms follow the same assumption that the network has a multipoint-to-one-point traffic and tree-like topology. Our paper focuses on different scenarios where multiple concurrent and periodic real-time data flows are transmitted along the predefined routes in a mesh network to implement a control loop. Our goal is to ensure every flow meets its deadline constraint.

Low-latency scheduling function (LLSF) [21] daisy-chains the time slots used in a multihop path to reduce the end-to-end latency. LLSF schedules the transmission slot in a 3-step process: (1) for each reception slot from the previous hop, determine the number of slots (the “gap”) between that and the previous reception slot from the same neighbor; (2) pick the slot which has the largest gap to its left; and (3) pick the new transmission slot to the next hop is the closest unused slot to the right of the selected reception slot. These simple heuristic selecting rules may not work well when there are multiple flows in the network.

Adaptive MUltihop Scheduling (AMUS) algorithm [7] is proposed to provide low-latency guarantee for time-critical applications. The proposed method enables multihop scheduling by reserving communication resources along the route for each set of end-to-end communication links. In the time slot allocation, a tentative cell allocation method is utilized to provide additional resources to vulnerable links such that possible MAC retransmissions can be accommodated within the same slot frame, hence significantly reducing the delay caused by interference or collisions. Tentative cell allocation is a kind of over provision, which may reduce the delay of single data flow significantly. However, when there are many concurrent data flows, over provision may waste too much communication resources and make it hard for all the data flows meeting their deadlines.

Optimum scheduling of multiple channels and time slots in IEEE 802.15.4e TSCH mesh-like networks is an NP-complete problem. Kim and Lee [15] applied metaheuristic approaches, simulated annealing (SA), and particle swarm optimization (PSO) to solve the scheduling problem. The objective function is to minimize the end-to-end delay. Generally, metaheuristic approaches need more iterations (which means longer computation time) to get better results. The computation complexity is high; for example, the computation of PSO-based scheduling which outperforms SA-based scheduling increases exponentially as the number of nodes increases. The generated schedule is fixed, and any transmission failure will cause the whole schedule to be invalidated. Besides, minimizing the total delay is not our goal. Our focus is ensuring all the data flows meet their deadlines.

To sum up, much of the existing works such as TASA, ECSA, and MSA, has been focused on the data collection use cases with a multipoint-to-point traffic pattern. Our SPRF addresses the more general case of traffic flows between multiple source-destination pairs. There are some existing methods such as LLSF and AMUS that can deal with mesh topology and general traffic pattern. However, they allocate time/frequency based on simple heuristic schemes. Comparing with them, SPRF adopts maximum matching and graph coloring to find more links that can transfer data flows concurrently and utilize dynamic priority assigning to allocate these links to the urgent data flows first. SPRF is also computationally efficient and flexible comparing with nature-inspired algorithms such as PSO and SA.

3. System Model and Problem Definitions

We consider an IEEE 802.15.4e TSCH network comprising of several synchronized nodes. A central scheduling entity is in charge of supervisory control of network traffic flows and management, which computes the optimized time slot and channel assignment for the TSCH MAC. This scheduling entity can be implemented at a powerful node such as gateway node (coordinator) or at the backbone.

The network can be modelled as a graph \( G = (V, E) \) where \( V = \{n_0, n_1, n_2, \ldots, n_N\} \) is the set of nodes. \( |V| = N \) is the total number of nodes in the network. \( E \) is the set of links. A link \( l_{i,j} \in E \subset V \times V \) means node \( i \) can transmit data to node \( j \).

In such a network, a set of periodic real-time data flows denoted by \( DF = \{DF_0, DF_1, \ldots, DF_{m-1}\}, \ i \in [0, m-1], \) needs to be transmitted. The slot frame contains \( T \) time slots. At the beginning of the slot frame, each data flow \( DF_i \) periodically generates a frame that originates at source and has to be delivered to destination along predefined route within deadline \( D_i \). The transmission of a frame from one node to its neighboring node can be implemented in one time slot. Deadline is expressed as the number of time slots relative to the beginning of the slot frame. Deadlines of all frames fall within the same slot frame they were generated. Therefore, \( D_i \leq T \). Figure 1 shows an example of transmitting three periodic real-time data flows in a TSCH network consisting of six nodes.

Each data flow \( DF_i \) can be represented as a \( 2 \times m \) matrix, where \( m \) is the total number of predefined route hops. Each column of this matrix represents the source node id and destination node id of this hop. For example, the flows in Figure 1 can be represented as

\[
DF_0 = \begin{bmatrix} n_4 & n_1 \\ n_1 & n_0 \end{bmatrix}, \\
DF_1 = \begin{bmatrix} n_2 \\ n_0 \end{bmatrix}, \\
DF_2 = \begin{bmatrix} n_0 & n_3 \\ n_3 & n_5 \end{bmatrix}
\]

For further analysis, we use \( 3 \times m \) matrix \( DF_k(k) \) to represent the data flows needed to be scheduled at a time slot
Figure 1: Example of transmitting three periodic real-time data flows in a TSCH network consisting of six nodes.

The third row represents the number of frames needed to be transmitted at time slot \( k \). For example, if the data flows in Figure 1 only transmit one frame from source to destination and intermediate nodes do not generate frame, at the initial time slot \( (k = 0) \), the data flows can be represented as

\[
\begin{align*}
\text{DF}_0 &= \begin{bmatrix} n_4 & n_1 \\ n_1 & n_0 \end{bmatrix}, \\
\text{DF}_1 &= \begin{bmatrix} n_2 \\ n_0 \end{bmatrix}, \\
\text{DF}_2 &= \begin{bmatrix} n_3 & n_5 \\ n_3 & n_5 \end{bmatrix}.
\end{align*}
\]

The goal is to establish a network schedule to allocate time slots and assign channels to multiple data flows to fulfill the real-time requirements. Figure 2 shows a feasible schedule for the scenario illustrated in Figure 1. In this schedule, node \( n_4 \) transmits the frame of \( \text{DF}_0 \) to node \( n_1 \) at the time slot 0. At the same time slot, link \( n_0 \rightarrow n_3 \) is scheduled to transmit the frame of \( \text{DF}_2 \) using a different channel. At the time slot 1, \( \text{DF}_0 \) and \( \text{DF}_2 \) are transmitted along \( n_1 \rightarrow n_0 \) and \( n_3 \rightarrow n_5 \) with different channels, respectively. Finally, \( \text{DF}_1 \) is transmitted at time slot 2.

To establish such a schedule, we must avoid the possible conflict and interference that may happen in the wireless environment. In 802.15.4e networks, all the links are half duplex links, which means a node cannot transmit and receive at the same time and it cannot receive from multiple nodes at the same time either. Two or more links that have a common node cannot be scheduled at the same slot. We call such links conflict links; for example, \( n_2 \rightarrow n_0 \) and \( n_2 \rightarrow n_0 \) in Figure 3 are interference links. Only noninterfering and nonconflicting links can be scheduled in parallel to use the same time slot and same channel.

After the discussion on network topology, traffic pattern, and possible conflict and interference, we define the scheduling problem as the follows.

Given an 802.15.4e TSCH network \( G \) with a periodical real-time data flow set \( \text{DF} \), finding a time slot allocating and channel assigning scheme satisfying the following requirements:

(a) No conflict links scheduled at the same time slot
(b) Interfering links should use different channels if assigned with the same time slot or use different time slots
(c) The frames generated by the data flows should be transmitted to their destinations within their deadline

4. Scheduling for Periodic Real-Time Flows

In this section, we present scheduling for periodic real-time flow (SPRF) to solve the problem defined in the above section.

4.1. Overview. The main steps of SPRF are summarized in Figure 4. SPRF has complete topology information, \( G = (V, E) \); the traffic load generated by the data flows, \( \text{DF} \); the length of slot frame, \( T \); the deadline vector \( D \); and the number of available channels, \( n_{ch} \). Starting from such information, SPRF builds the schedule through an iterative process from time slot 0 to time slot \( T-1 \). The basic idea is to arrange the transmission of high-priority flows as soon as possible and make the number of parallel transmissions at each time slot as large as possible.
In each iteration, \textit{PriCalculating} function calculates the link priority \( l_{DF} (k) \) for each data flow \( DF \) at current slot \( k \) based on its residual hops, deadline, and the number of possibly transmitted frames.

Then \textit{PriMaxMatching} function finds maximum non-conflicting links set \( NCL (k) \). Note that \( NCL (k) \) may contain interference links that have to be scheduled on a different channel offset. In order to avoid interference at this time slot, \textit{FindInterferenceGraph} builds an interference graph \( I (k) = \{V_I (k), E_I (k)\} \), where \( V_I (k) \) is the set of links of \( NCL (k) \) and \( E_I (k) \) is the set of edges representing the interference between the links. If there is no edge between these two links in the interference graph, then we can use the same channel offset for both of them, thus reducing the number of channels needed. On the contrary, if there is a graph edge between two links, then we should not assign the same channel offset to them since interference will happen. Applying \textit{Coloring} to \( I (k) \), we can finally find a set of conflict and interference-free links with dedicated channel offsets that can transmit data flows concurrently in this slot.

We may not schedule all the transmissions at current time slot, the remaining ones will be reconsidered in the next step of the procedure. At the end of each step, based on the schedule built for the considered time slot \( k \), traffic information is updated. The update assures that the links to be scheduled at the next time slot \( k + 1 \) will be chosen according

\begin{itemize}
  \item No
  \begin{itemize}
    \item Determine link priorities:
      \( l_{DF} (k) = \text{PriCalculating} (G, DF, k) \)
    \item Find maximum nonconflicting links set from possible data flows transmitting links:
      \( NCL (k) = \text{PriMaxMatching} (G, DF, k) \)
    \item Find interference links from \( NCL (k) \) by constructing interference graph:
      \( I (k) = \text{FindInterferenceGraph} (NCL (k)) \)
    \item Assign channel offset to interference links by coloring:
      \( \text{Selected} = \text{Coloring} (I (k), DF, N_{ch}) \)
    \item Update traffic information:
      \( f_{set} = f_{set} + [(k, \text{Selected})] \)
      \( k++; \)
  \end{itemize}

  \item Yes
  \begin{itemize}
    \item \textit{Find feasible schedule}
      \textit{Return} \( f_{set} \: \text{fset} \)
    \item \textit{Cannot find feasible schedule}
      \textit{Return} \( f_{set} \: \text{fset} \)
  \end{itemize}
\end{itemize}

\textbf{Figure 4: Flowchart of the main procedures of the SPRF algorithm.}
4.2. Dynamic Priority Assignment. At a given time slot, each unfinished data flow is assigned a priority \( \rho_i \) as
\[
\rho_i = \frac{D_i}{D_i - \text{HODF}_i},
\]
where \( D_i \) is the deadline of flow \( i \) and \( \text{HODF}_i \) is the number of the residual hops to the flow destination. \( D_i \) is expressed in the number of time slots. Since we assume the number of time slot to transmit a frame is 1, \( \text{HODF}_i \) represents the least number of time slots needed to transfer the frame to the destination. The data flow with the most remaining hops and with the earliest deadline has the highest priority. At the beginning, all data flows are unfinished and labelled by priorities. With the scheduling process advancing, some data flows finishing their transmissions are no longer considered.

Next, we assign the priorities to the links that have frames belonging to the unfinished data flows to transmit. The unrelated links are excluded to reduce the scheduling complexity. The link priority is represented by a tuple:
\[
I_{DF} (k) = \left( \max_{i \in \text{DF}_k} \rho_i, n_k \right),
\]
where the first element is the highest priority of data flow whose frames can be transmitted by this link and \( n_k \) is the number of total frames now waiting this link to transmit. All these possibly transmitting links are put into a queue \( L (k) \) in the descending order by \( \max_{i \in \text{DF}_k} \rho_i \). If two or more links have the same \( \max_{i \in \text{DF}_k} \rho_i \), they are further sorted by \( n_k \) in the descending order. From the head of the queue to the end of the queue, the priority of the corresponding link descends.

4.3. Priority-Based Maximum Matching. In graph theory, maximum graph matching often occurs and is a well-studied problem. We use the blossom algorithm [22] because it can find maximum matching on general graphs containing odd-length cycles that the most used bipartite matching cannot deal with. In the blossom algorithm, an odd-length cycle in the graph (blossom) is contracted to a single vertex, with the search continuing iteratively in the contracted graph. If there are no odd cycles in graph, blossoms will never be found and the algorithm reduces to the standard algorithm for matching in bipartite graphs. The process of PriorMaxMatching consists of the following steps:

1. Constructing a reduced graph \( G (k) \) from \( G \) only containing the links from \( L (k) \) and corresponding nodes.
2. Constructing initial matching as follows:

\( (i) \) Set initial matching empty, and put the head of \( L (k) \) (the link with the highest priority) into the initial matching
\( (ii) \) Get the next link of \( L (k) \), and set it as the current link
\( (iii) \) If the current link does not have any common nodes with the links in the initial matching, put it into the initial matching
\( (iv) \) If all the links of \( L (k) \) are traversed, the subprocess ends, else go to Step (ii)

3. If the initial matching contains all the links of \( L (k) \), or the initial matching is a maximum matching, the process ends up with returning the links belonging to the initial matching as NCL \( (k) \). Otherwise, go to Step 4.

4. Utilizing the blossom algorithm to construct a maximum matching by iteratively improving the initial matching along augmenting paths in the graph. The links belonging to this maximum matching constitute NCL \( (k) \).

The links with the highest priorities are inserted into the initial matching first. When using the blossom algorithm to find augmented paths, the unmatched nodes are selected in the descending order of priorities of their connected links. This is the reason why we call our maximum matching finding process priority based. It can allocate time slot for the transmissions of urgent data flows as early as possible.

If NCL \( (k) \) does not contain all the links of \( L (k) \), the transmissions corresponding to the links of \( L (k) \)–NCL \( (k) \) will be scheduled at the next time slot.

4.4. Coloring. The links of NCL \( (k) \) are conflict free but not interference free. The coloring process allocates different channels to the interference links to make them transmit parallel at a same time slot. First, an interference graph \( I (k) = \{V_i (k), E_i (k)\} \) is constructed by FindInterference-Graph to describe the possible interference among the links of NCL \( (k) \). \( V_i (k) \) is the set of links of NCL \( (k) \). If two links are close enough to interfere with each other, an edge is added to connect the vertices representing these two links. \( E_i (k) \) is the set of such edges that represent the existing interference.

If there is no edge between the vertices of \( I (k) \), the links represented by these vertices can use the same channel at a given time slot. Otherwise, they must use different channels. Therefore, the channel allocation can be defined as a graph coloring problem. Given an interference graph \( I (k) \) and a number \( m \), determine if the graph can be colored with at most \( m \) colors such that no two adjacent vertices of the graph are colored with same color. Here, coloring of a graph means assignment of colors to all vertices, and \( m \) equals the available number of channel offsets \( N_{ch} \). If \( N_{ch} \) is larger than the minimum number of needed colors, all the links can be scheduled at this time slot. Otherwise, only a part of links belonging to \( I (k) \) can be scheduled.

Minimum vertex coloring is a well-known NP-complete problem. Complex algorithms may get a better solution at
the cost of computing time. However, our scenario requires getting a solution fast. Therefore, we use a heuristic algorithm to solve this coloring problem, which consists of the following steps:

1. Sorting the vertices in the descending order according to the priorities of the corresponding links, putting them to the uncolored set, and initiate the colored set as empty.
2. Finding the vertex $l_j$ corresponding to the highest priority link from the uncolored set. If the number of available colors is larger than 0, assigning a color to it, removing it from the uncolored set, adding it into the colored set, and decreasing the number of available colors by 1. Otherwise, the process ends up with established schedule.
3. Dividing the vertices in the uncolored set into two groups. One is called neighbor list containing the vertices interfered with the vertices of colored set, and the other one is called nonneighbor list containing the vertices not interfered with the vertices of the colored set.
4. Finding a vertex from the nonneighbor list, assigning it the same color with $l_j$, and adding it to the colored set.
5. Repeating Steps 3 and 4 for all the vertices in the nonneighbor list. The links in the colored set form a conflict- and interference-free set CIF ($k$), which can be assigned a certain channel offset value and written to the schedule table.
6. If the uncolored set is empty, the process ends up with the established schedule. Otherwise, reinitiating the colored set as empty, repeating Steps 2, 3, 4, and 5 to find another CIF ($k$), and adding them to the schedule table.

This splitting and searching process repeatedly divides the vertices set into colored set and uncolored set and assigns the colors to the vertices belonging to uncolored set until all vertices are colored or all colors are used. In the worst case, this process needs $m$ iterations. In the average case, this process needs $\log m$ iterations.

If all the vertices are colored, the established schedule contains all the links having the frames to transmit at this time slot. Otherwise, only the links corresponding to colored vertices are assigned channel offset and transmit frames at this time slot. The links corresponding to uncolored vertices are left to be scheduled at the next time slot. Sorting the vertices in the descending priority order in coloring process ensures that high-priority links are scheduled first.

4.5. Time Complexity Analysis. Given a network $G$ mentioned above, $n$ is the number of nodes, $m$ is the number of transmitting links, at each time slot the maximum number of links needed to be scheduled is less than or equal to $m$, and the maximum number of connected nodes is also less than or equal to $n$.

The time complexity of fast sorting is $O(m \log m)$. The blossom-based matching commonly has a total of run time of $O(m^2)$. An improved algorithm [23] used here can reduce the time complexity to $O(m^{1.5})$. The complexity of constructing an interference graph is $O(m)$. Our heuristic coloring algorithm has the average time complexity of $O(m \log m)$ and the worst case complexity ($m^2$). Since our algorithm repeats the sorting, matching, and coloring operations for each time slot until all transmissions finish or every time slot in the slot frame has been traversed, the above steps are repeated up to $T$ times. The average time complexity of the whole process is $O(Tmn^{1.5}) + O(Tm \log m)$, and the worst case time complexity is $O(Tmn^{1.5}) + O(Tm^2)$.

Therefore, our scheduling algorithm is a polynomial time algorithm.

4.6. Dealing with Transmission Failure. Although time division and frequency hopping used by TSCH network make its links have higher reliability than traditional 802.15.4 links, transmission failure may still occur due to the versatility of wireless communication. One solution is to regenerate the scheduling table for the remaining time slots of this slot frame, which may cause too much computing and communicating overhead since all the nodes should be notified, stop its current operation, and get the new scheduling table from the central scheduling entity.

Therefore, we use a local repairing mechanism to retransmit frames. Since we adopt a central scheduling scheme, each node can have the whole scheduling table. When a node fails to receive acknowledgment of its transmitted frame, it will scan the scheduling table to find a nearest spare cell to retransmit the frame to the next hop. When a node does not receive the frame successfully at the allocated cell, it will stay awake until the slot frame ends. The node receiving this retransmitted frame repeats this operation until it arrives at the destination or the slot frame ends. It should be noticed that some nodes may not transmit according to the established schedule since they do not receive the frames from their previous hop. Their frame queue may be empty. In that case, these nodes simply stop transmission and wait for incoming frames.

This repairing process can be described as delay-and-insertion. It may lead to longer end-to-end delay and cause the frame to miss its deadline. It also ignores the possibility of reusing the previously occupied now available time slots. But it is easy to implement and has low overhead. We will improve our algorithm to deal with transmission failure more efficiently in the future work.

5. Performance Evaluation

We have carried out simulation-based study to evaluate the performance of the proposed algorithm.

5.1. Simulation Settings. To perform the experiments, the 6TiSCH simulator [24] is used: an open-source, event-driven Python simulator developed by the 6TiSCH Working Group. We have simulated a TSCH network with a mesh topology, where nodes are randomly distributed in an area of $200 \times 200$ m$^2$. Each node has a coverage of 50 meters. The
number of nodes varies from 10 to 60. Slot frame length is set to 50 slots with the duration of 10 ms for each time slot. We also assume a number of available channels \( n_{ch} \in [2, 10] \). The successful transmission ratio varies from 95% to 100%.

To generate data flows representing traffic conditions, a fraction of nodes is selected as sources and destinations. The sets of sources and destinations are disjoint. These data flows pass through randomly chosen available paths. The number of data flows varies from 1 to 25. The hops of routing path vary from 2 to 5. The average number of frames generated by a source node within a single slot frame can vary in the range [2, 6]. The deadline of each flow is set equal to the length of single slot frame.

The goal of our algorithm is to schedule as many data flows meeting their deadline as possible instead of minimizing their delay. So we use deadline satisfaction ratio (DSR) to characterize the real-time performance of our algorithm, which is defined as the ratio of the number of frames that are transferred to the destinations before their deadline. We also use radio duty cycle to evaluate the resource consumption of our algorithm. The duty cycle over DSR is used to show the comprehensive performance.

A single simulation run includes two stages: deploying a network with certain parameters and simulating the behaviour of the network for a period of time. For each run, all nodes’ locations are randomly chosen, which generates the dedicated mesh topology. For each set of parameters, a large number of runs are performed. The simulation results presented in Sections 5.2–5.4 are depicted as average values with a 95% confidence interval. Each data point represents 100 simulation runs.

5.2. Comparing with Existing Algorithms. We compare our SPRF algorithm with two existing scheduling algorithms, LLSF and AMUS. To demonstrate the effectiveness of the dynamic priority mechanism, we implement a fixed-priority SPRF, FSPRF, which assigns a fixed priority \( \rho_{fi} \) to each data flow according to its deadline. \( \rho_{fi} \) is computed as

\[
\rho_{fi} = \frac{1}{D_i},
\]

where \( D_i \) is the deadline of flow \( i \). Figure 5 shows the DSR, and Figure 6 shows the average duty cycle for varying number of data flows.

As shown in Figure 5, LLSF always has a lowest DSR since it adopts a very simple heuristic rule to allocate the slot along the path, which does not consider deadline and interference. When the number of data flows is small, AMUS performs slightly better than FSPRF and SPRF. AMUS uses tentative slot allocation to reserve backup slots that can be used for retransmitting when transmission failure occurs.

When the number of data flows increases, the number of redundant time slots needed by AMUS increases, which squeezes the available time slots for the remaining data flows. Our delay-and-insertion mechanism does not allocate extra time slots for successful transmissions. This is the reason why DSR drops and AMUS performs worse than FSPRF and SPRF. Furthermore, AMUS only uses a simple rule to avoid possible interference. When a slot is allocated, the link is assigned a channel offset that is not used at this slot. The channel allocation is not optimized. Our algorithm has more efficient use of each time slot by exploring spatial diversity and accommodating multiple concurrent communications in the same slot, as long as they are not interfering with one another. Specifically, when the number of data flows exceeds 20, the average number of frames needed to be transmitted in a single time slot may exceed 5, which means more 5 links should be active at the same time. It requires a very efficient scheduling algorithm to avoid conflicts and interferences given 4 available channels. Compared with LLSF and AMUS whose SDRs drop very quickly, SPRF still can achieve 85% SDR when the number of data flows is 20 and 70% SDR when the number of data flows is 25.
SPRF always performs better than FSPRF in terms of DSR since SPRF utilizes dynamic priority assignment to schedule the links transmitting the frames with earliest deadline first. When the number of data flows is small, there are enough available time slots and channels, the difference between FSPRF and SPRF is not significant. With the number of data flows increasing, the difference becomes more significant.

As shown in Figure 6, AMUS always has the highest duty cycle since the introduction of tentative time slots makes nodes have to wake up more frequently in a slot frame in order to listen or transmit in these time slots. When the number of data flows is small, LLSF, FSPRF, and SPRF can arrange time slots and channels to transfer all the data flows within a slot frame, the needed number of transmissions is almost the same. So the duty cycle is almost the same. When the number of flows increases, LLSF and FSPRF fail to transfer all the data flows within a slot frame, which leads to lower duty cycle than SPRF.

To clearly demonstrate the advantage of SPRF, Figure 7 shows the results on the metric duty cycle over DSR. SPRF always has the lowest value, which means SPRF can achieve a similar DSR with the least energy consumption. It also means SPRF can achieve the highest DSR with the same energy consumption. With the number of flows increasing, the advantage becomes more significant, which proves SPRF can generate efficient schedules especially when the traffic load is high.

5.3. Impacts of Number of Available Channels. Figures 8 and 9 show impacts of the number of available channels on the performance. Generally, more available channels mean more links can be scheduled to transmit at the same time slot, which explains the SDRs of all the algorithms increase with the number of available channels increasing as illustrated in Figure 8. We also find SPRF achieves the highest SDR with the least number of available channels (when $n_{ch}$ is 4, the SDR of SPRF is 0.7, the SDR of FSPRF is 0.63, and the SDRs of LLSF and AMUS are lower than 0.4) since it adopts dynamic priority assignment, matching, and coloring to make most efficient use of space and frequency diversity and schedules the most urgent data flows first.

In terms of average duty cycle, AMUS has the highest value due to tentative slot allocation. The duty cycle of LLSF is lower than that of SPRF/FSPRF since the DSR of LLSF is lower than that of SPRF/FSPRF. When the number of available channels is high, the difference is more significant since SPRF/FSPRF achieves much higher DSR than LLSF.

Considering duty cycle and DSR together, DSR of SPRF is higher than those of all other methods when they all have
the same duty cycle as shown in Figure 10. When the number of nodes is small, the advantage is more obvious.

5.4. Impacts of Network Size. With the number of nodes increasing, the chances that find more node pairs that can transmit frames concurrently increase, which allows more data flows to be scheduled simultaneously. Therefore, the DSR increases for all the algorithms as shown in Figure 11. SPRF always achieves the highest DSR due to optimizing the link and channel allocation. The difference between FSRPF and AMUS becomes very small because they all adopt flow by flow allocation mechanism and the effects of channel allocation decrease with the increasing number of nodes.

As shown in Figure 12, the average duty cycle also decreases since there are more nodes in the network. AMUS always achieves the highest duty cycle due to abundant time slot reserving. SPRF has a higher duty cycle than LLSF since its DSR is higher than LLSF.

As shown in Figure 13, SPRF always has the lowest duty cycle over DSR for all different number of nodes. When the number of nodes is small, SPRF has a great advantage.

6. Conclusions and Future Directions

This paper presents a centralized scheme to schedule multiple concurrent periodic real-time flows in TSCH networks with mesh topology. Each flow is assigned a dynamic priority based on its deadline and the hops remaining to reach the destination. A maximum matching algorithm is utilized to
find conflict-free links, which provides more chances to transfer high-priority flows at each time slot. Frequency allocation is implemented by graph coloring to make finally selected links interference free. Simulation results show that our algorithm clearly outperforms the competition on the deadline satisfaction ratio with a similar radio duty cycle.

Our algorithm takes the routing path as an input deciding which nodes participate in the schedule. In a mesh network, there may exist several feasible paths from the source to the destination. Considering alterable paths may lead to better scheduling. We will study cross-layer methods to jointly optimize the routing paths and MAC schedules to improve the performance. Wireless transmitting is vulnerable, and we also will study more efficient schedule repairing approaches to deal with transmission failure more efficiently.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partially supported by the CERNET Innovation Project under Grant no. NGII20170323 and Natural Science Foundation of Hubei Province under Grant no. 2016CFC721.

References


