Chapter 4
Mixed-Criticality Scheduling for TDMA Networks

Abstract To improve the schedulability of mixed-criticality industrial wireless networks, targeted algorithms should be developed. Therefore, in this chapter, we propose a mixed-criticality scheduling algorithm. The algorithm supports centralized optimization and adaptive adjustment. It can improve both the schedulability and flexibility. We conduct extensive simulations, and the results demonstrate that the proposed scheduling algorithm significantly outperforms existing ones.

4.1 Background

Since time division multiple access (TDMA) scheduling has the high predictability, it is widely used in industrial networks [1–3]. In mixed-criticality industrial wireless networks, when there are not enough resources for all data packets, the low-criticality data packets have to be discarded. Hence, almost all of mixed criticality systems must support discarding strategies [4–7]. Previous works on single-criticality industrial wireless networks apply centralized TDMA methods to guarantee the real time performance and reliability of industrial wireless networks, e.g. [8–13]. However, the centralized TDMA methods are inflexible and difficult to cope with discarding.

Intuitively, two types of methods can be used to schedule data flows in mixed-criticality wireless networks. The first type is to schedule flows based on criticality monotonic priorities. The criticality monotonic scheduling assigns the higher priority to the important flows and schedules them first. However, this method considers the criticality as the temporality. Actually, they are not equivalent. Thus, the criticality monotonic scheduling algorithm is not suitable for mixed criticality systems. This has also been demonstrated in [14]. The second type is to use the algorithms that have been proposed for previous mixed-criticality systems, such as uniprocessor/multiprocessor systems [15–17] and networks [18–20], to solve our problem. However, industrial wireless networks are different from the previous systems. To guarantee the strict requirements on the real-time performance and reliability, the main problem to be solved is how to avoid the collision and interference between parallel data flows. Mixed-criticality uniprocessor/mulitprocessor

X. Jin et al., *Mixed-criticality Industrial Wireless Networks*, Wireless Networks,
https://doi.org/10.1007/978-981-19-8922-3_4

systems only consider independent processors and do not have the interference between parallel tasks. Mixed criticality wired networks and IEEE 802.11-based wireless networks are based on CSMA (Carrier Sense Multiple Access) protocols, which are unacceptable by industrial wireless networks due to the unpredictability (We give more clarifications on the differences between our system and others in Sect. 4.2). Therefore, previous algorithms cannot be used without modification in mixed criticality industrial wireless networks. In this chapter, we present a holistic scheduling solution to guarantee the real time and reliability requirements of data flows in resource-constrained industrial wireless networks. Although some flexible and scalable MAC protocols [21, 22] are adopted to improve the real-time performance and reliability of networks, they are based on only local information and cannot optimize the whole network. Therefore, our scheduling method is implemented in the application layer. According to the generated schedules, each network node transmits or receives packets in the MAC (Medium Access Control) layer. The scheduling method of the application layer can manage all data flows based on global information. Thus, it can get the optimized solution.

This chapter includes the following:

First, we propose a scheduling algorithm for mixed-criticality networks. The scheduling algorithm not only implements the optimized global management for all flows, but also reserves network resources for dynamic adjustments to enhance the real time performance and reliability of important flows. It makes a trade-off between the scheduling performance and the flexibility. Performance evaluations demonstrate that the proposed scheduling algorithm outperforms existing ones.

Second, we present a schedulability analysis for the proposed scheduling algorithm. We analyze end-to-end delay for flows, and determine whether they are all schedulable. Simulation results show that our schedulability analysis is more effective than existing ones.

4.2 System Model

Industrial wireless networks must support the strict requirements on real time performance and reliability. Therefore, we consider an industrial wireless network as follows. It consists of a gateway and some devices. We use the node set $N = \{n_1, n_2, \ldots\}$ to denote these nodes. The physical layer of our industrial wireless networks is specified by the IEEE 802.15.4 protocol. It supports 16 non-overlapping channels. However, due to external interference, not all of them can be accessed all of the time. We denote the number of available channels as M ($1 \leq M \leq 16$). Our network serves the flow set $F = \{f_1, f_2, \ldots\}$. Each element f_i is characterized by $< T_i, \Pi_i, \chi_i >$. Each flow f_i periodically generates a packet at its period T_i, and then sends it to the destination via the routing path Π_i. The relative deadline of each packet is equal to the period T_i, i.e., a packet is released at the time t, and it must be delivered to its destination before the time $(t + T_i + 1)$. In industrial wireless protocols, e.g. [23, 24], periods conform to the expression

$$b \times 2^a, \tag{4.1}$$

where a is an integer value and b is the unit-period.

To keep consistent with related works on mixed criticality systems, our network also supports two criticality levels, L-crit (Low criticality) and H-crit (High criticality). The dual-criticality model can be easily extended to multi-criticality model. If the flow f_i is important, its criticality level χ_i is denoted as H. Otherwise, its criticality level χ_i is L. When the system is running in the normal mode without any exception, all flows are delivered to their destinations within deadlines. If important equipment has an exception, the corresponding data must be submitted frequently and via two paths to avoid faults on a single path. Thus, in our system model, the H-crit flows have two parameter sets: the L-crit parameters $< T_i(L), \Pi_i(L) >$ in the normal mode; the H-crit parameters $< T_i(H), \Pi_i(H) >$ in the exception mode, and $T_i(H) \leq T_i(L)$. $\Pi_i(L)$ is a path that is used by the H-crit flow in the normal mode. $\Pi_i(H)$ contains two paths that are used by the H-crit flow in the exception mode, and the two paths transmit the same packet to improve the reliability. In order to clearly distinguish these paths, they are denoted as $\Pi_i(L) = \{\pi_i^*\}$ and $\Pi_i(H) = \{\pi_i', \pi_i''\}$. The path π_i^* (and π_i', π_i'') is the set of links from the source to the destination. In this chapter, we do not consider how to select routing paths. We assume all paths have been given before generating schedules. The dynamism this chapter addresses refers to using different parameters in different modes. Transmitting a packet through the j-th link of the path π_i^* (or π_i', π_i'') is called as the transmission τ_{ij}^* (and τ_{ij}', τ_{ij}''). Each transmission has two attributes $< n_\alpha, n_\beta >$, which denote the transmission's source and destination respectively. As the constrained resources must provide enough services to H-crit flows, the L-crit flows cannot be transmitted when exceptions happen. Therefore, L-crit flows only have a parameter set $< T_i(L), \Pi_i(L) >$.

To improve the reliability of industrial networks, we adopt the TDMA scheme in the MAC layer. The network manager, which is connected to the gateway, assigns a time slot and a channel offset to each transmission. A transmission only is scheduled at the given time slot and on the given channel offset. Packets are generated periodically, and the schedules of corresponding transmissions have the same period. The schedules with the same period are organized within a *superframe* [24]. Transmitting a packet from the source to the destination has to be done in a superframe. Thus, superframes repeat themselves periodically, and then flows can be transmitted successfully. Figure 4.1a shows a simple network, which contains two flows f_1 and f_2. When the system is in normal mode, the flows use their L-crit parameters. Their periods are 8 time slots and 4 time slots, and their paths are $\{e_{52}, e_{21}\}$ and $\{e_{98}, e_{87}, e_{74}, e_{41}\}$, where e_{ij} denotes the link from the node n_i to the node n_j. Figure 4.1b shows their superframes with different periods. CH and TS denote Channel Offset and Time slot.

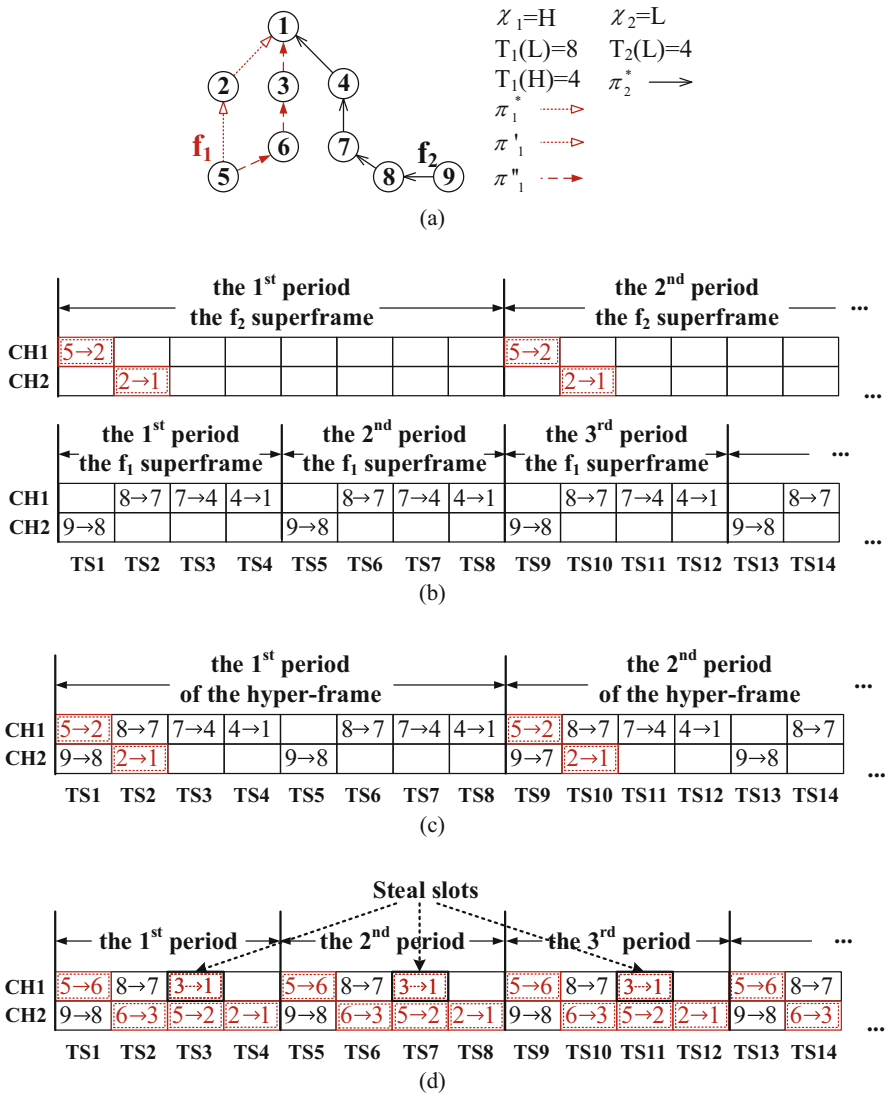

Fig. 4.1 Graph routing and superframe. (**a**) A network. (**b**) Superframes with different periods. (**c**) A hyper-frame. (**d**) The flow f_2 steal slots from the flow f_1

Two types of improper schedules will lead to transmission interference, which seriously affects the network reliability. The first type, called *node interference*, is that more than one transmissions uses the same node at the same time slot. Each node is only equipped with one transmitter. Therefore, one node cannot serve more

than one transmissions at the same time. The second type is called *scheduling interference* which means that more than one transmissions is scheduled at the same time slot and on the same channel. These overlapping transmissions cannot be separated. To avoid transmission interference between different superframes, we consider all superframes as a hyper-frame whose period is the lowest common multiple of all superframes. According to the period's Expression (4.1), the hyper-period $\mathcal{T} = LCM(T_1, T_2, \ldots) = \max_{\forall f_i \in F}\{T_i\}$. Figure 4.1c shows the hyper-frame of the simple example. We only consider how to schedule flows in the first hyper-period, since after that, all schedules are repeated periodically. The network manager generates all schedules under two situations: Situation 1: when the network is deployed; and Situation 2: when the deployment is changed. Due to the requirement of industrial applications being fixed, the deployment is not often changed. Thus, the schedules may be generated several times, but not frequently. According to this schedule information, it obtains the working modes of each node at every time slot, and then delivers them to the corresponding nodes. For the schedules in Fig. 4.1c, from $TS1$ to $TS4$, working modes of the node n_2 are {receive, send, idle, idle}.

When a node intends to send a transmission of L-crit flows, it waits for a constant time and then listens to whether its channel is used. If the channel is used by H-crit flows, the node discards its transmission. Otherwise, the node sends the transmission. Note that although the node uses the carrier sense technique to determine whether an L-crit transmission is discarded or not, it is different from the CSMA scheme. For L-crit flows, the node performs carrier sense within time slots of the TDMA frame. If the L-crit transmission is not discarded, it is also scheduled based on the TDMA scheme. When a node intends to send a transmission of H-crit flows, it immediately sends it at the beginning of the assigned time slot. The scheduling algorithm assigns the proper time slot and channel for each transmission and prevents H-crit transmissions from interfering with other H-crit transmissions. Therefore, H-crit transmissions are sent directly without checking the channel. In this way, the H-crit flow can steal slots from L-crit flows when it needs more resources to cope with exceptions [25]. Note that the H-crit flow using H-crit parameters is not permitted to steal slots that are used by any other H-crit flows even if these H-crit flows are using L-crit parameters. Figure 4.1d shows an example of mixed-criticality schedules. The period of the H-crit flow f_1 is changed from 8 to 4, and the new path $\{e_{56}, e_{63}, e_{31}\}$ begins to be used. In this case, there are not enough time slots. The H-cirt transmission $3 \rightarrow 1$ (the solid line in Fig. 4.1d) steals the resource of the L-cirt transmission $7 \rightarrow 4$. Based on the stealing strategy, the dynamic adjustment can be supported.

The *schedulable* flow set is defined as follows. When the system is in the normal mode, the flow set is schedulable if all flows characterized by L-crit parameters can hit their deadlines. When there are exceptions in the system, the flow set is schedulable if all H-crit flows can hit their deadlines no matter which parameters they are using.

4.3 Problem Statement

Based on the above system model, we describe the mixed criticality scheduling problem as follows. Given the network and the flow set F, our objective is to schedule transmissions in the time slot and channel dimensions such that the flow set is schedulable.

To explain the problem more clearly, we formulate the problem as a Satisfiability Modulo Theories (SMT) specification. The transmission τ_{ij}^* (and τ_{ij}', τ_{ij}'') is assigned the s_{ij}^*-th (and s_{ij}'-th, s_{ij}''-th) time slot and the r_{ij}^*-th (and r_{ij}'-th, r_{ij}''-th) channel offset. Note that a transmission is scheduled periodically. Therefore, the transmission uses all of the time slots $s_{ij} + g \cdot T_i$ ($\forall g \in [0, \frac{\mathcal{T}}{T_i})$) in a hyper-frame. These assignments must respect the following constraints.

(a) **Channel Offset Constraint**.

$$\forall f_i, \forall j \in [1, |\pi_i^*|], 1 \leq r_{ij}^* \leq M$$

For each transmission, its assigned channel offset must be in M available channels. This expression is for transmissions in the path π_i^*. Other transmissions τ_{ij}' and τ_{ij}'' in paths π_i' and π_i'' have the same constraint, and we omit them for simplicity.

(b) **Releasing Sequence Constraint**.

$$\forall f_i, \forall j \in [1, |\pi_i^*| - 1], s_{i,j}^* < s_{i,j+1}^*$$

In a routing path, the transmission $\tau_{i,j+1}$ is released after the transmission $\tau_{i,j}$ is scheduled. We still omit paths π_i' and π_i''.

(c) **Real Time Constraint**.

$$\forall f_i, 1 \leq s_{i,|\pi_i^*|}^* \leq T_i(L)$$

All transmissions cannot miss deadlines. Likewise, $s_{i,|\pi_i'|}'$ and $s_{i,|\pi_i''|}''$ have the same constraint.

(d) **Interference Constraint**. Assigning resources to transmissions must prevent the happening of node interference and scheduling interference. We use $\delta(\tau_a, \tau_b)$ to denote whether there exists interference between τ_a and τ_b,

$$\delta(\tau_a, \tau_b) = (\tau_a \cap \tau_b = \emptyset)?(\eta(s_a, s_b) \wedge (r_a = r_b)) : \eta(s_a, s_b),$$

where $\eta(s_a, s_b) = \bigvee_{\forall h \in [0, \frac{\mathcal{T}}{T_a}), \forall k \in [0, \frac{\mathcal{T}}{T_b})} (s_a + h \cdot T_a = s_b + k \cdot T_b)$ means whether the assigned time slots of τ_a and τ_b overlap each other. If the two transmissions do not use the same node, i.e., $\tau_a \cap \tau_b = \emptyset$, then they can be scheduled at different time slots or on the different channel offsets. Otherwise, there exists node interference and they cannot be scheduled at the same time slot. The

transmissions of the H-crit flow f_i are classified into three sets $\Gamma_i^* = \{\tau_{ij}^* | \forall j \in [1, |\pi_i^*|]\}$, $\Gamma_i' = \{\tau_{ij}' | \forall j \in [1, |\pi_i'|]\}$ and $\Gamma_i'' = \{\tau_{ij}'' | \forall j \in [1, |\pi_i''|]\}$. For the L-cirt flow f_i, $\Gamma_i' = \Gamma_i'' = \emptyset$, and then $\forall f_i \in F, \Gamma_i = \Gamma_i^* \cup \Gamma_i' \cup \Gamma_i''$. Thus, the interference constraints in the normal mode and exception mode are as follows.

(d.1) Normal mode

$$\forall \tau_a, \tau_b \in \bigvee_{\forall f_i \in F} \Gamma_i^*, \delta(\tau_a, \tau_b) = 0$$

(d.2) Exception mode

$$\forall f_i, f_g \in F, \chi_i = \chi_g = H, \forall \tau_a \in \Gamma_i, \forall \tau_b \in \Gamma_g, \delta(\tau_a, \tau_b) = 0$$

The mixed criticality scheduling problem is NP-hard [26]. Our SMT specification can be solved by some solvers, such as Z3 [27] and Yices [28]. These solvers can find satisfying assignments for quite many problems, and their solutions have been an excellent standard to evaluate the effectiveness of other methods [29]. However, the running time may be unacceptable for complex networks and flow sets. Therefore, we propose a heuristic scheduling algorithm in Sect. 4.4 to solve the problem.

4.4 Scheduling Algorithm

In this section, we first introduce how to schedule transmissions, and then, based on these schedules, we determine working modes of each node at every time slot.

4.4.1 A Slot-Stealing Scheduling Algorithm

We propose a slot-stealing scheduling algorithm based on RM (StealRM). The proposed StealRM optimizes the solution according to the global information, and permits transmissions to share the same resource when the transmissions have different levels of criticality. Hence, the schedules can be adaptively adjusted based on the requirements of H-crit flows.

The proposed StealRM is shown in Algorithm 4.1. Each flow is assigned as the RM priority. If two flows have the same RM priority, the flow with the smaller ID has the higher priority. The transmission's priority is equal to its flow's priority. The set R contains all of schedulable transmissions (lines 1 and 19), and the set R' denotes released transmissions at the current time slot (line 3). At every time slot t, we first sort elements of R' according to the decreasing order of priorities, and τ_1

in the set R' has the highest priority (line 4). Then, for each transmission τ_a in the set R', we check whether it can be scheduled at the current time slot without any interference (lines 7–23). Let $\mathcal{F}(\tau_a)$ denote the flow that the transmission τ_a belongs to (line 6). The set Y_t^{HL} contains the transmissions that have been scheduled at the time slot t and belong to H-crit flows with L-crit parameters. The sets Y_t^H and Y_t^L correspond to those in H-crit flows with H-crit parameters and L-crit flows, respectively. The transmissions in the set Y' and the transmission τ_a cannot steal slots from each other. According to the criticality level of τ_a, the set Y' is assigned different transmissions (lines 7–13). If the transmission τ_a belongs to an H-crit flow with H-crit parameters, then it cannot steal slots from other H-crit transmissions (lines 7–8). Y^H and Y^{HL} may contain the transmissions belonging to the same flow with τ_a. These transmissions do not interfere the scheduling of τ_a. Thus, the set $\{\forall \tau_{ig}^*\}$ needs to be excluded from Y^H and Y^{HL} (line 8). Similarly, if the transmission τ_a belongs to an H-crit flow with L-crit parameters, then it cannot steal slots from any other transmissions (lines 9 and 10). If the transmission τ_a belongs to an L-crit flow, then its slots cannot be stolen by L-crit flows and H-crit flows with L-crit parameters (lines 11 and 13). When there is no node interference between τ_a and Y', and at least one channel is idle (line 14), the transmission τ_a can be scheduled at this current time slot. $\Theta(Y')$ denotes the channels that have been used by Y'. However, if the current time slot has exceeded its deadline, the flow set is unschedulable (lines 15 and 16). Otherwise, the time slot and channel offset of the transmission τ_a are assigned (line 18), and the schedulable transmission set R and the scheduled transmission set Y_t^H (Y_t^L and Y_t^{HL}) are updated (lines 19–26).

The number of iterations of the **for** loop in line 2 and the **for** loop in line 5 is $O(|\mathcal{T}|)$ and $O(|\Gamma|)$, respectively. The complexity of line 4, line 14 and line 21 is $O(|\Gamma| log |\Gamma|)$, $O(|\Gamma|)$ and $O(\frac{\mathcal{T}}{T_{min}})$, respectively. Therefore, the time complexity of Algorithm 4.1 is $O(|\mathcal{T}||\Gamma|^2 \frac{\mathcal{T}}{T_{min}})$.

4.4.2 Node Working Mode

Nodes have three working modes, including transmit mode ($\mathbb{S}$), receive mode ($\mathbb{R}$) and idle mode. We use $w_{\alpha,t}^H =< \mathbb{S}$ (or $\mathbb{R}$), $r_a >$ to denote that at the time slot t the node n_α transmits (or receives) H-crit flows on the channel r_a. Similarly, $w_{\alpha,t}^L$ denotes that the node n_α serves L-crit flows. Algorithm 4.2 determines the working mode for each node. For each transmission, we have assigned a time slot and a channel offset in Algorithm 4.1. According to the assignments, the working modes of the sender node and receiver node of the transmission can be obtained (lines between 4 and 10). The time complexity of Algorithm 4.2 is $O(|\Gamma| \frac{\mathcal{T}}{T_{min}})$.

Note that a node may serve two flows at the same time slot, but the two flows must have different criticality levels. Otherwise, node interference occurs. At the beginning of the time slot t, the node works in mode $w_{\alpha,t}^H$. Then, in a constant time if it needs to send an H-crit flow or has received a flow, it continues working in the

Algorithm 4.1 StealRM

Input: the flow set F
Output: the scheduling results $\forall s_a$ and $\forall r_a$;
the schedulable transmission set $R \leftarrow \{\tau_{i1}^*, \tau_{i1}', \tau_{i1}'' | \forall f_i \in F\}$;
for $\forall t \in [1, \mathcal{T}]$ **do**
  $R' \leftarrow R$;
  sort R' according to the decreasing order of priorities;
  for each a from 1 to $|R'|$ **do**
    $i \leftarrow \mathcal{F}(\tau_a)$;
    if $\chi_i == H$ and $\tau_a \in \Gamma_i' \cup \Gamma_i''$ **then**
      $Y' \leftarrow \bigcup_{\forall h \in [0, \frac{\mathcal{T}}{T_i(H)})} (Y^H_{t+T_i(H)\times h} \cup Y^{HL}_{t+T_i(H)\times h}) - \{\forall \tau_{ig}^*\}$;
    else if $\chi_i == H$ and $\tau_a \in \Gamma_i^*$ **then**
      $Y' \leftarrow (\bigcup_{\forall h \in [0, \frac{\mathcal{T}}{T_i(L)})} Y^L_{t+T_i(L)\times h}) \cup (\bigcup_{\forall h \in [0, \frac{\mathcal{T}}{T_i(H)})} (Y^H_{t+T_i(H)\times h} \cup Y^{HL}_{t+T_i(H)\times h})) - \{\forall \tau_{ig}', \tau_{ig}''\}$;
    else
      $Y' \leftarrow (\bigcup_{\forall h \in [0, \frac{\mathcal{T}}{T_i(L)})} Y^L_{t+T_i(L)\times h}) \cup (\bigcup_{\forall h \in [0, \frac{\mathcal{T}}{T_i(H)})} Y^{HL}_{t+T_i(H)\times h})$;
    end if
    if $\bigwedge_{\forall \tau_b \in Y'} (\tau_a \cap \tau_b \neq \emptyset)$ and $|\Theta(Y')| < M$ **then**
      if t exceeds the deadline of f_i **then**
        return unschedulable;
      end if
      $s_a \leftarrow t$; $r_a \leftarrow$ a random channel that is not in $\Theta(Y')$;
      $R \leftarrow R - \{\tau_a\}+$ the next transmission of τ_a;
      if $\chi_i == H$ and $\tau_a \in \Gamma_i' \cup \Gamma_i''$ **then**
        $\forall h \in [0, \frac{\mathcal{T}}{T_i(H)})$, $Y^H_{t+T_i(H)\times h} \leftarrow Y^H_{t+T_i(H)\times h} + \{\tau_a\}$;
      else if $\chi_i == H$ and $\tau_a \in \Gamma_i^*$ **then**
        $\forall h \in [0, \frac{\mathcal{T}}{T_i(L)})$, $Y^{HL}_{t+T_i(L)\times h} \leftarrow Y^{HL}_{t+T_i(L)\times h} + \{\tau_a\}$;
      else
        $\forall h \in [0, \frac{\mathcal{T}}{T_i(L)})$, $Y^L_{t+T_i(L)\times h} \leftarrow Y^L_{t+T_i(L)\times h} + \{\tau_a\}$;
      end if
    end if
  end for
end for
return $\forall s_a$ and $\forall r_a$;

same mode at this time slot. Otherwise, it works in mode $w^L_{\alpha,t}$. However, when its mode $w^L_{\alpha,t}$ is $\mathbb{S}$, it must determine whether the assigned channel is clear or not before it sends the flow. If the channel has been occupied by H-crit flows, the flow has to be discarded. The switch time between different modes is very short compared with a time slot. For example, the switch time of the transceiver CC2420 is just 200 μs while a time slot is 10 ms. Generally, at a time slot, most nodes only serve one flow or are idle, while only a few nodes serve two flows.

Algorithm 4.2 Working mode

Input: the scheduling results $\forall s_a$ and $\forall r_a$
Output: all $w^L_{*,*}$ and $w^H_{*,*}$

all $w^L_{*,*}$ and $w^H_{*,*}$ are initiated as idle mode;
for $\forall \tau_a \in \Gamma$ **do**
  $i \leftarrow \mathcal{F}(\tau_a)$; $< n_\alpha, n_\beta >$ are the sender and receiver of τ_a;
  if $\chi_i == H$ **then**
    $\forall h \in [0, \frac{\mathcal{T}}{T_i(H)})$, $w^H_{\alpha, s_a + T_i(H) \times h} \leftarrow < \mathbb{S}, r_a >$;
    $\forall h \in [0, \frac{\mathcal{T}}{T_i(H)})$, $w^H_{\beta, s_a + T_i(H) \times h} \leftarrow < \mathbb{R}, r_a >$;
  else
    $\forall h \in [0, \frac{\mathcal{T}}{T_i(L)})$, $w^L_{\alpha, s_a + T_i(L) \times h} \leftarrow < \mathbb{S}, r_a >$;
    $\forall h \in [0, \frac{\mathcal{T}}{T_i(L)})$, $w^L_{\beta, s_a + T_i(L) \times h} \leftarrow < \mathbb{R}, r_a >$;
  end if
end for
return all $w^L_{*,*}$ and $w^H_{*,*}$;

4.5 Scheduling Analysis

In this section, we analyze the worst case end-to-end delay for each flow and use the delay to test the schedulability of the flow set. If the worst case delay of all flows does not exceed deadlines, the flow set is schedulable. For the sake of simplicity, we first explain how to compute the worst case delay in single-criticality networks (in Sect. 4.5.1) and then extend it to mixed-criticality networks (in Sect. 4.5.2).

4.5.1 Analyzing Method for Single-Criticality Networks

Besides transmitting time, the end-to-end delay is introduced by the interference from higher priority flows. Therefore, in Sect. 4.5.1.1, we present the analyzing method of the total interference. In Sect. 4.5.1.2 we distinguish the different types of interference and compute the worst case delay.

4.5.1.1 Total Interference

During the time interval between the release and completion of the flow f_k, all the active transmissions that belong to the higher priority flows may have node interference or scheduling interference to the flow f_k. Therefore, in the worst case, the total interference is equal to the number of those higher-priority transmissions. The method of computing the workload in a period has been proposed in multi-processor systems [30]. The mapping between the multiprocessor system model and the network model has been explained in the work [31], in which a channel corresponds to a processor and a flow is scheduled as a task. Therefore, we propose

our analyzing method based on the work [30], which is the start-of-the-art analysis for multiprocessor systems. To make this chapter self-contained, we first simply introduce the method of multiprocessor systems, and then present our method.

For the simplicity of expression, the multiprocessor system uses the same notations as our network model. For multiprocessor systems, the calculation of the worst case delay of the task f_k is based on the *level-k busy period* (as shown in Definition 4.1).

Definition 4.1 (Level-k Busy Period for Multiprocessor Systems) The level-k busy period is the time interval $[t_0, t_k)$, in which t_k is the finish time of the task f_k, and t_0 satisfies the following conditions:

1. $t_0 < t_r$ where t_r is the release time of the task f_k.
2. $\forall t \in [t_0, t_r]$, at the time t, all processors are occupied by higher-priority tasks.
3. $\forall t < t_0, \exists t' \in [t, t_0]$, at the time t', at least one processor is occupied by lower-priority tasks.

If there is no t_0 that satisfies all conditions, then $t_0 = t_r$.

The level-k busy period is determined by the workload of all higher-priority tasks. The set $\bar{P}(f_k)$ contains the tasks with higher priority than the task f_k. If the task f_i ($f_i \in \bar{P}(f_k)$) has a job that is released earlier than the level-k busy period and its deadline is in the busy period, then the task f_i has the carry-in workload in the level-k busy period. Otherwise, the task has no carry-in workload. The two types of workload are presented as follows, and the length of the level-k busy period is x.

1. In the level-k busy period, if the task f_i has no carry-in workload, the upper bound of its workload is

$$W_k^{NC}(f_i, x) = \left\lfloor \frac{x}{T_i} \right\rfloor \cdot c_i + \min\{x \mod T_i, c_i\},$$

   where c_i is the execution time of the task f_i.
2. If the task f_i has the carry-in workload, the upper bound of its workload is

$$W_k^{CI}(f_i, x) = \left\lfloor \frac{\max\{x - c_i, 0\}}{T_i} \right\rfloor \cdot c_i + c_i + \alpha,$$

   where $\alpha = \min\{\max\{\max\{x - c_i, 0\} - (T_i - D_i), 0\}, c_i - 1\}$ and D_i is the worst case delay of the task f_i.

Based on the upper bounds of workload, two types of interference of the task f_i to the task f_k are as follows:

$$I_k^{NC}(f_i, x) = \min\{\max\{W_k^{NC}(f_i, x), 0\}, x - c_k + 1\},$$

$$I_k^{CI}(f_i, x) = \min\{\max\{W_k^{CI}(f_i, x), 0\}, x - c_k + 1\}.$$

Therefore, the total interference suffered by the task f_k is

$$\Omega_k(x, \bar{P}^{NC}(f_k), \bar{P}^{CI}(f_k)) = \sum_{\forall f_i \in \bar{P}^{NC}(f_k)} I_k^{NC}(f_i, x) + \sum_{\forall f_i \in \bar{P}^{CI}(f_k)} I_k^{CI}(f_i, x),$$

where $\bar{P}^{NC}(f_k)$ and $\bar{P}^{CI}(f_k)$ denotes the set of tasks without carry-in workload and the set of tasks with carry-in workload, respectively. In a busy period, at most $M-1$ higher-priority tasks have carry-in workload. Therefore, the set $\bar{P}^{CI}$ contains $M-1$ tasks that have maximal values of $I_k^{CI}(f_i, x) - I_k^{NC}(f_i, x)$. Other tasks are in the set $\bar{P}^{NC}$.

In the following, we propose our analyzing method. Industrial wireless networks apply strict periodic schedules based on superframes, which can reduce system complexity and run time overhead. While in multiprocessor systems and previous works about wireless networks, schedules are variable, i.e., the assigned time slots to a task (or a flow) are non-periodic, so our workload bounds are not the same as previous ones. Our workload bounds are computed with Theorem 4.1. Definition 4.2 defines the level-k busy period in the network.

Definition 4.2 (Level-k Busy Period for Networks) The level-k busy period is the time interval $[t_0, t_k)$, in which t_k is the finish time of the flow f_k and t_0 satisfies the following conditions:

1. $t_0 < t_r$ where t_r is the release time of the flow f_k.
2. $\forall t \in [t_0, t_r]$, at the time t, all channels are occupied by higher-priority flows or there exists node interference between the scheduled flows and the flow f_k.
3. $\forall t < t_0, \exists t' \in [t, t_0]$, at the time t', there is no node interference and at least one channel is occupied by lower-priority flows or idle.

If there is no t_0 that satisfies all conditions, then $t_0 = t_r$.

Theorem 4.1 *The workload bounds can be computed with*

$$W_k^{NC}(f_i, x) = W_k^{CI}(f_i, x) = \left\lfloor \frac{x}{T_i} \right\rfloor \cdot c_i + \min\{x \mod T_i, c_i\}, \tag{4.2}$$

where c_i is the number of hops in the path π_i, i.e. $c_i = |\pi_i|$.

Proof of Theorem 4.1 The computation of the non-carry-in workload $W_k^{NC}(f_i, x)$ is shown in Fig. 4.2a. There are $\left\lfloor \frac{x}{T_i} \right\rfloor$ complete periods and a scheduling window $(x \mod T_i)$. In the scheduling window, at most c_i workloads exist. Therefore, the expression of the non-carry-in workload is shown as Eq. (4.2).

In the following, we compute W_k^{CI} as shown in Fig. 4.2b. The notations A and B denote the two incomplete periods, respectively. We know that $A < T_i$, $B < T_i$ and $A + B = (x \mod T_i)$ or $(x \mod T_i + T_i)$. We discuss the two cases as follows.

Case 1: $A + B = x \mod T_i$ We draw out the windows A and B in Fig. 4.3a. We consider four different value ranges of the windows A and B as shown in Table 4.1,

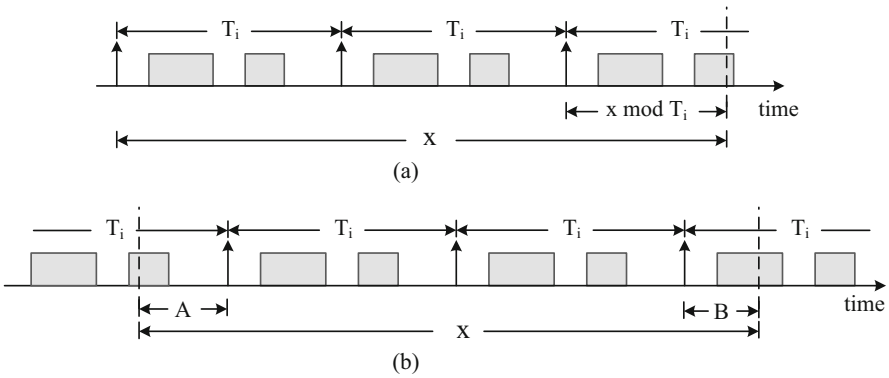

Fig. 4.2 Illustration of Theorem 4.1. (**a**) $W_k^{NC}(f_i, x)$. (**b**) $W_k^{CI}(f_i, x)$

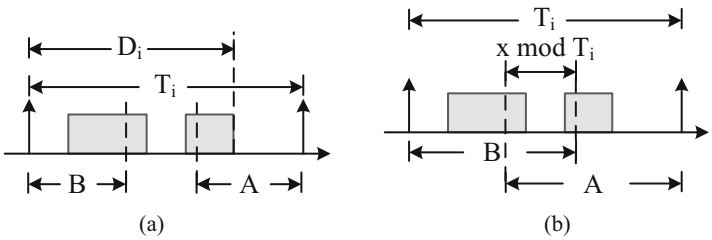

Fig. 4.3 Computation of W_k^{CI}. (**a**) $A + B = x \mod T_i$. (**b**) $A + B = x \mod T_i + T_i$

in which if $A \geq T_i - D_i$ and $B \geq D_i$, it is Case 2. If $A < T_i - D_i$, then there is no workload in A. If $B \geq D_i$, then all execution time c_i must be the available workload. In this case, the workload can also be expressed as $\min\{B, c_i\}$. Therefore, only if $A < T_i - D_i$, the workload is $\min\{B, c_i\}$. If $A \geq T_i - D_i$ and $B < D_i$, the time interval $T_i - D_i$ does not contain any workload. Therefore, the available window $A + B$ is equal to $(x \mod T_i) - (T_i - D_i)$.

In Case 1, we can get that the total workload is

$$\left\lfloor \frac{x}{T_i} \right\rfloor \cdot c_i + C1. \tag{4.3}$$

The notation $C1$ denotes the workload in the incomplete period as shown in Table 4.1. It is equal to $\min\{B, c_i\}$ or $\min\{x \mod T_i - (T_i - D_i), c_i\}$.

Table 4.1 The workload in the incomplete period under different value ranges of A and B

Workload	$A < T_i - D_i$	$A \geq T_i - D_i$
$B \geq D_i$	c_i	Case 2
$B < D_i$	$\min\{B, c_i\}$	$\min\{x \mod T_i - (T_i - D_i), c_i\}$
C1	$\min\{B, c_i\}$	$\min\{x \mod T_i - (T_i - D_i), c_i\}$

Case 2: $A+B = x \mod T_i+T_i$**, which is shown in Fig. 4.3b** In this case, there are $\left\lfloor \frac{x-T_i}{T_i} \right\rfloor$ complete periods. In the windows A and B, at most $c_i+\min\{x \mod T_i, c_i\}$ workloads exist. Therefore, the workload of Case 2 is

$$\left\lfloor \frac{x-T_i}{T_i} \right\rfloor \cdot c_i + c_i + \min\{x \mod T_i, c_i\}$$

$$\Rightarrow \left\lfloor \frac{x}{T_i} \right\rfloor \cdot c_i + \min\{x \mod T_i, c_i\}. \tag{4.4}$$

Comparing with Eqs. (4.3) and (4.4), the upper bound of workload is Eq. (4.4). Since $(x \mod T_i)$ is not less than B and $(x \mod T_i) - (T_i - D_i)$, equation (4.4) is the same as Eq. (4.2). The theorem holds. □

Due to the two types of workload having the same computing formula, we do not distinguish them in the following and use $W_k(f_i, x)$ to denote them. Based on the workload bound, the interference of the flow f_i to the flow f_k is

$$I_k(f_i, x) = \min\{\max\{W_k(f_i, x), 0\}, x - c_k + 1\}.$$

Thus, the total interference suffered by the flow f_k is

$$\Omega_k^{total}(x, \bar{P}(f_k)) = \sum_{\forall f_i \in \bar{P}(f_k)} I_k(f_i, x).$$

4.5.1.2 Worst Case Delay in Single-Criticality Networks

$\Omega_k^n(x, \bar{P}(f_k))$ and $\Omega_k^s(x, \bar{P}(f_k))$ denote node interference and scheduling interference suffered by the flow f_k in the level-k busy period. If there exists a node interference at a time slot, the flow f_k cannot be transmitted at this time slot, no matter how many channels are idle, i.e., the flow f_k is delayed one time slot due to the node interference. However, only when M transmissions are scheduled at a time slot, does the flow f_k suffer scheduling interference and is delayed for one time slot. In the worst case, all the node interference and scheduling interference will introduce a delay to the flow f_k. Therefore, the worst case delay is

$$\Omega_k^n(x, \bar{P}(f_k)) + \left\lfloor \frac{\Omega_k^s(x, \bar{P}(f_k))}{M} \right\rfloor + c_k. \tag{4.5}$$

From Eq. (4.5), we know that node interference introduces more delay. Since the sum of node interference and scheduling interference is $\Omega_k^{total}(x, \bar{P}(f_k))$, so when as much as possible node interference occurs, the end-to-end delay is the worst case.

The upper bound of node interference introduced by h consecutive hops of the flow f_i to the flow f_k is computed as

$$R_{k,i}(h) = \max_{\forall a \in [1, c_i - h]} \{|\{\tau_{iy} | \forall \tau_{iy}, y \in [a, a+h], \exists \tau_{kz} \text{ such that } \tau_{iy} \cap \tau_{kz} \neq \emptyset\}|\}.$$

Thus, the workload introduced by transmissions that have node interference is

$$W_k^n(f_i, x) = \left\lfloor \frac{x}{T_i} \right\rfloor \cdot R_{k,i}(c_i) + R_{k,i}(\min\{x \mod T_i, c_i\}).$$

Then,

$$I_k^n(f_i, x) = \min\{\max\{W_k^n(f_i, x), 0\}, x - c_k + 1\},$$

and

$$\Omega_k^n(x, \bar{P}(f_k)) = \sum_{\forall f_i \in \bar{P}(f_k)} I_k^n(f_i, x).$$

Then, we can get that the worst case delay of the flow f_k in the single-criticality network is

$$D_k = \Omega_k^n(x, \bar{P}(f_k)) + \left\lfloor \frac{\Omega_k^{total}(x, \bar{P}(f_k)) - \Omega_k^n(x, \bar{P}(f_k))}{M} \right\rfloor + c_k.$$

From the definition of the level-k busy period, we know that the length x is the upper bound of the delay D_k (shown in Theorem 4.2).

Theorem 4.2 *For the flow f_k and the level-k busy period, the following holds:*

$$x \geq D_k.$$

Proof of Theorem 4.2 We assume by contradiction that $x < D_k$. From the definition of the level-k busy period (Definition 4.2), we know that the finish times of the busy period and the flow f_k are the same, and t_0 must be less than (the first condition) or equal to t_r (when t_0 does not satisfy at least one condition). If $x < D_k$, then $t_r < t_0$ as shown in Fig. 4.4. It is not consistent with the definition. The above assumption does not hold. □

According to Theorem 4.2, the solution of Eq. (4.6) is the upper bound of end-to-end delay D_k.

$$x = \Omega_k^n(x, \bar{P}(f_k)) + \left\lfloor \frac{\Omega_k^{total}(x, \bar{P}(f_k)) - \Omega_k^n(x, \bar{P}(f_k))}{M} \right\rfloor + c_k \tag{4.6}$$

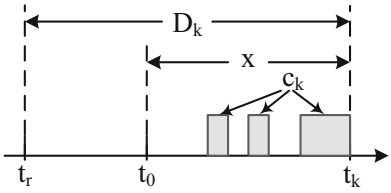

Fig. 4.4 Illustration of Theorem 4.2

Equation (4.6) can be solved by the iterative fixed point search [32]. The iterative calculation of x starts with $x = c_k$; until the value of x does not change.

4.5.2 Mixed-Criticality Scheduling Analysis

In dual-criticality networks, there are three types of worst case delay.

1. D_k^L: the worst case end-to-end delay of the L-crit flow.
2. D_k^{HL}: the worst case end-to-end delay of the H-crit flow with the L-crit parameter.
3. D_k^H: the worst case end-to-end delay of the H-crit flow with the H-crit parameter

We use $D(x, Q, c)$ to denote $\Omega_k^n(x, Q) + \left\lfloor \frac{\Omega_k^{total}(x,Q) - \Omega_k^n(x,Q)}{M} \right\rfloor + c$. The methods of computing these types of delays are similar. The only difference is that higher-priority flows they suffered are different, i.e., their interference sets Q are different. H-crit flows have multiple paths. These paths suffer different interference and cause different delays. Therefore, we use sub-flows f_k^*, f_k' and f_k'' to distinguish them.

If there are H-crit flows with H-crit parameters in networks, L-crit flows can be discarded. Therefore, when we compute the delay D_k^L, all flows have L-crit parameters. Thus, $D_k^L = D(x, Q_k^L, c_k^*)$, where $Q_k^L = \{f_i^* | \forall f_i^*, T_i(L) < T_k(L)\}$ and $c_k^* = |\pi_k^*|$.

Similarly, for H-crit flows with L-crit parameters, the interference set is $Q_k^{HL} = \{f_i', f_i'' | \forall f_i', \forall f_i'', \chi_i = H, T_i(H) < T_k(L)\} \cup \{f_i^* | \forall f_i^*, T_i(L) < T_k(L)\}$. Thus, $D_k^{HL} = D(x, Q_k^{HL}, c_k^*)$, where $c_k^* = |\pi_k^*|$.

An H-crit flow with its H-crit parameter suffers the interference from H-crit flows with H-crit parameters. The H-crit flow has two sub-flows f_k' and f_k''. For these sub-flows, their interference set is $Q_k^H = \{f_i', f_i'' | \forall f_i', \forall f_i'', \chi_i = H, T_i(H) < T_k(H)\} \cup \{f_i^* | \forall f_i^*, \chi_i = H, T_i(L) < T_k(H)\}$ and $c_k' = |\pi_k'|$, $c_k'' = |\pi_k''|$. Thus, $D_k'^H = D(x, Q_k^H, c_k')$ and $D_k''^H = D(x, Q_k^H, c_k'')$, and then $D_k^H = \max\{D_k'^H, D_k''^H\}$.

According to the above delays, the schedulability test is as follows. For the L-crit flow f_k, if $D_k^L \leq T_k(L)$, it is schedulable; otherwise, unschedulable. For the H-crit flow f_k, if $D_k^{HL} \leq T_k(L)$ and $D_k^H \leq T_k(H)$, it is schedulable; otherwise, unschedulable. If all flows in a flow set are schedulable, the set is schedulable.

4.6 Performance Evaluations

In this section, we conduct experiments to evaluate the performance of our proposed methods.

4.6.1 Scheduling Algorithm

We consider three comparison methods: (1) **SMT** uses the Z3 solver [27], which is a high-performance solver being developed at Microsoft Research and whose solution has been regarded as an excellent standard, to solve our SMT specification (Sect. 4.3); (2) **noStealRM** applies the RM priority and does not allow slots to be stolen; (3) **StealCM** allows slots to be stolen and applies the criticality monotonic priority. Our method is **StealRM**. The performance metric we used is the *Schedulable ratio*, which is defined as the percentage of flow sets for which a scheduling algorithm can find a schedulable solution.

We randomly generate a number of test cases to evaluate these methods. For each test case, the number of channels M and the number of nodes $|N|$ are given. According to the suggestion in the work [33], these nodes are placed randomly in the square area A, and $A = \frac{|N|d^2\sqrt{27}}{2\pi}$, where the transmitting range d is 40 meters. Except for the gateway, each node has a data flow from itself to the gateway or vice versa. There are two necessary schedulability conditions for flow sets: (1) the network utilization U is not larger than 1; (2) the utilization of each node is not larger than 1. If a flow set does not satisfy the two conditions, it cannot be scheduled. Thus, in order to make flow sets available, we specify the network utilization $U (U < 1)$, and use the method UUniFast [34] to assign the utilization u_i for each flow, where $U = \sum_{\forall f_i \in F} u_i$. Then, if the flow set can satisfy condition (2), it is an available flow set. Otherwise, discard it, and repeat the process until an available set is found. The period of each flow can be obtained according to $T_i = \frac{c_i}{u_i}$. The high-crit probability of the flows is controlled by the parameter ρ. Routing paths are selected randomly.

In order to make test cases solvable by the Z3 solver, the parameters are set as $\rho = 0.3$, $M = 2$ and $U = 0.8$. For each configuration, 100 test cases are checked using the four algorithms. Figure 4.5 shows their schedulable ratios. Our algorithm StealRM is close to the result of Z3. In these simple test cases, the method StealCM has similar results with our algorithm StealRM. Figure 4.6 shows the average execution time of solvable test cases in Fig. 4.5. When the number of nodes is 25, the execution time of the method SMT is about 16.5 minutes. We also use the method SMT to solve the network with 30 nodes, but cannot get the result within 3 hours. Except for the method SMT, the execution time of other methods is not more than 10 milliseconds. Therefore, from the perspective of execution time, heuristic algorithms are significantly more efficient than the method SMT.

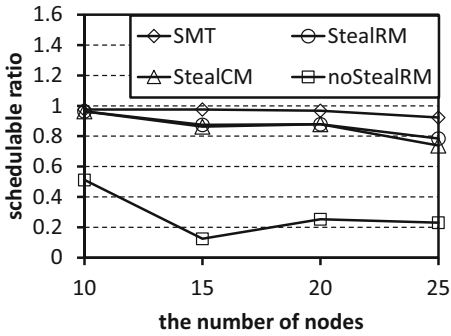

Fig. 4.5 Schedulability comparison among all methods

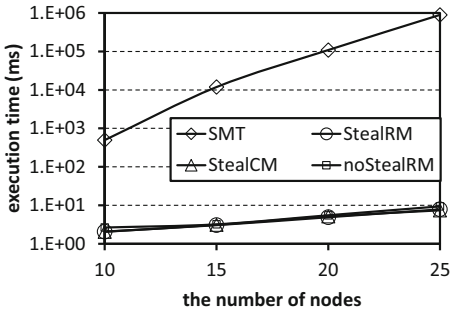

Fig. 4.6 Execution time comparison among all methods

Since the execution time of the method SMT is too long, the following experiments do not contain it. Figure 4.7 shows the schedulable ratios of the three scheduling algorithms. For each point in the figure, 500 test cases are randomly generated. From the figure, we can know that our algorithm StealRM has the highest schedulable ratio no matter with which parameters, while the algorithm noStealRM has the worst result. Therefore, the stealing mechanism can significantly improve the algorithm's performance. Our algorithm StealRM has better performance than the algorithm StealCM, especially when the node numbers are higher. This demonstrates that: (1) the priority should correspond to the urgency, but not the importance, while the stealing mechanism reflects the importance; (2) the urgency and the importance have to be distinguished, except in very small networks. Comparing these subfigures, we observe that schedulable ratios decrease with the increases of ρ, $|N|$, U and M. The reasons are as follows. An H-crit flow can be regarded as two L-crit flows. Thus, a larger value of the parameter ρ leads to more flows, which are hard to schedule. A test case contains $|N| - 1$ flows. Likewise, the larger $|N|$ makes scheduling hard. The network utilization U corresponds to the network workload. Heavy workloads lead to scheduling failures. Note that compared with Fig. 4.7a, d has three additional channels, but its schedulable ratios decrease. Because the two subfigures generate test cases according to the respective numbers of channels. Their test cases are different. Although the number of channels increases, the utilization is not changed. When the utilization U is constant, with the increase of the number of channels M, the packets that need to be transmitted increase. The increased packets

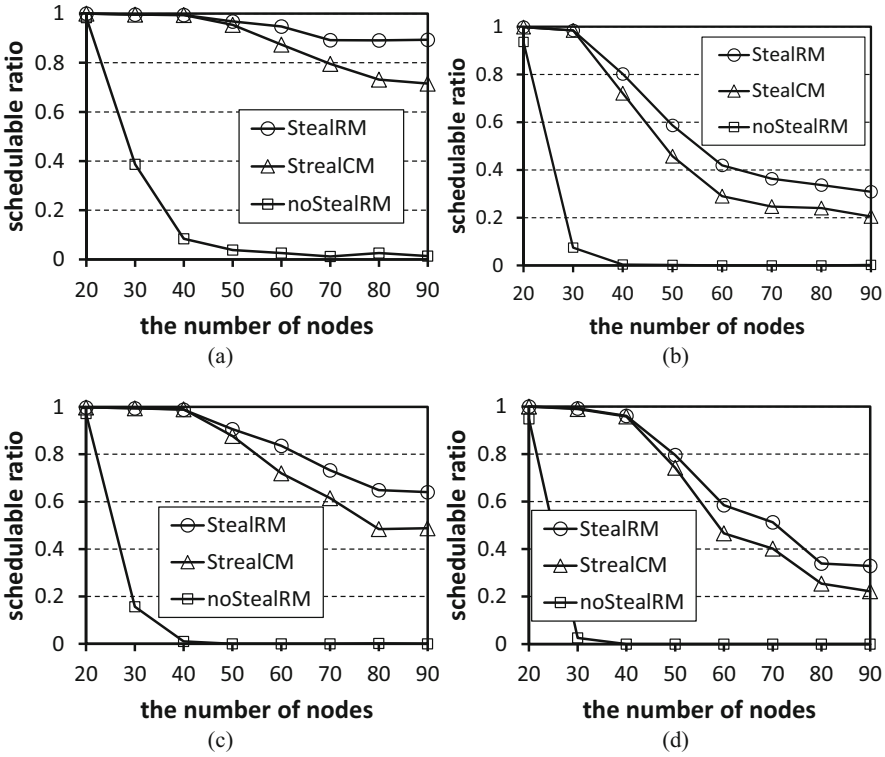

Fig. 4.7 Schedulability comparison among StealRM, StealCM and noStealRM. (**a**) $M = 6, U = 0.5, \rho = 0.3$. (**b**) $M = 6, U = 0.5, \rho = 0.4$. (**c**) $M = 6, U = 0.6, \rho = 0.3$. (**d**) $M = 9, U = 0.5, \rho = 0.3$

will introduce more interference, which has a negative impact on the scheduling performance. Therefore, Fig. 4.7d has a lower schedulable ratio than Fig. 4.7a.

Figure 4.8 shows the average execution time of Fig. 4.7. As the results are similar, we only show two subfigures for Fig. 4.7a, d. Compared with our algorithm StealRM, the algorithms StealCM and noStealRM need more time to find feasible solutions. Therefore, their execution time slightly increases. From the figure, we know that our algorithm StealRM does not introduce extra time costs. For the three algorithms, the execution time increases with the increase of the number of nodes, since more data flows need to be scheduled.

4.6.2 Analyzing Method

The comparison method is **SingleAna**, in which flow sets are tested using the single-criticality analysis. Our mixed-criticality analysis method is **MixedAna**. The performance metrics are the *analyzable ratio* (the percentage of flow sets which

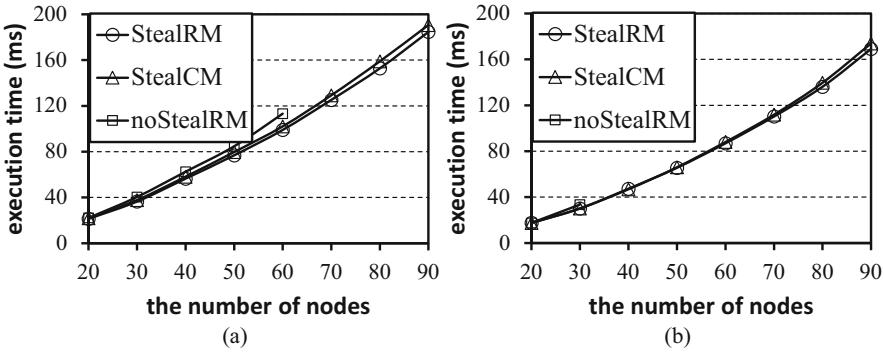

Fig. 4.8 Average execution time. (**a**) $M = 6$, $U = 0.5$, $\rho = 0.3$. (**b**) $M = 9$, $U = 0.5$, $\rho = 0.3$

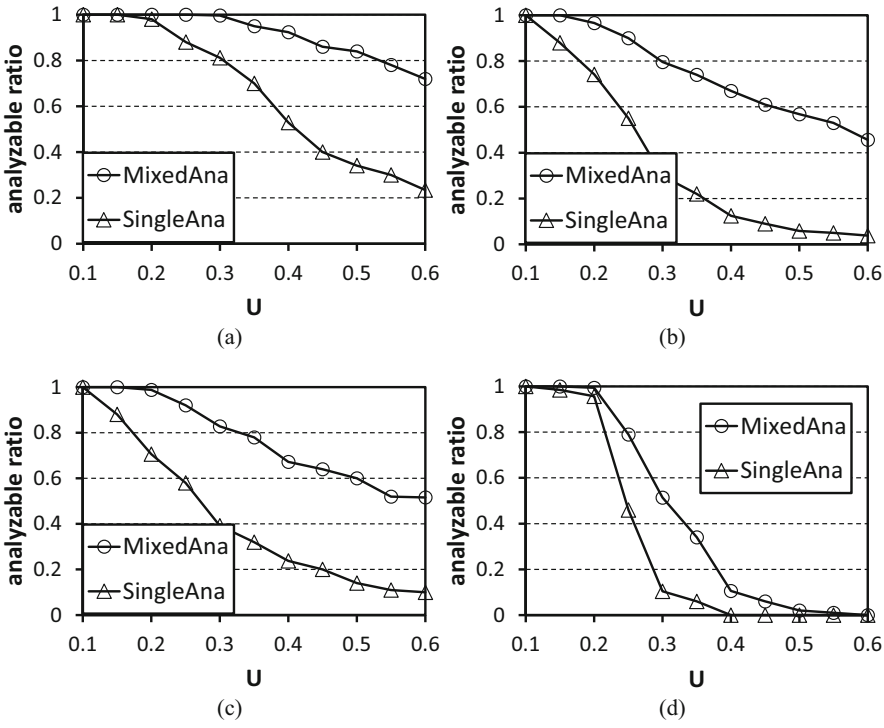

Fig. 4.9 Schedulability comparison among analyzing algorithms. (**a**) $|N| = 20$, $M = 6$, $\rho = 0.1$. (**b**) $|N| = 20$, $M = 6$, $\rho = 0.3$. (**c**) $|N| = 20$, $M = 9$, $\rho = 0.1$. (**d**) $|N| = 60$, $M = 6$, $\rho = 0.1$

are tested as schedulable by an analyzing method) and the *pessimism ratio* (the proportion of analyzed delay to the delay observed in StealRM). Figure 4.9 shows the comparison of analyzable ratios. For each point, 500 test cases are analyzed. Our method MixedAna outperforms SingleAna. The analyzable ratios decrease with

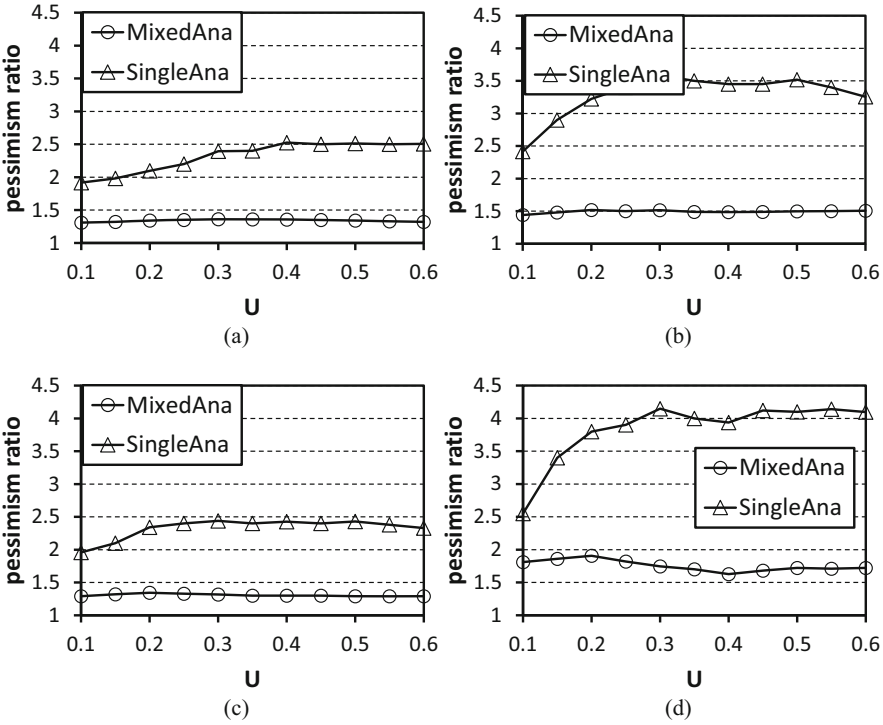

Fig. 4.10 Delay comparison with StealRM being used as the baseline. (**a**) $|N| = 20$, $M = 6$, $\rho = 0.1$. (**b**) $|N| = 20$, $M = 6$, $\rho = 0.3$. (**c**) $|N| = 20$, $M = 9$, $\rho = 0.1$. (**d**) $|N| = 60$, $M = 6$, $\rho = 0.1$

the increase of these parameters. The reasons are similar to those in Fig. 4.7. The increases of U and M lead to more packets, and the increases of $|N|$ and ρ lead to more flows. These will cause more interference. Thus, the analysis introduces more pessimism, and the analyzable ratios decrease. Figure 4.10 shows the pessimism ratios of experiments in Fig. 4.9. The pessimism ratios of MixedAna are less than 2, while the pessimism ratios of SingleAna are all larger than 2. This is because the interference that does not exist between H-crit and L-crit flows is eliminated in MixedAna.

4.7 Summary

Mixed-criticality data flows coexist in advanced industrial applications. They share the network resource, but their requirements for the real time performance and reliability are different. In this chapter, we propose a scheduling algorithm to guarantee their different requirements, and then analyze the schedulability of this

scheduling algorithm. Simulation results show that our scheduling algorithm and analysis have more performance than existing ones.

References

1. Soldati P, Zhang H, Johansson M (2009) Deadline-constrained transmission scheduling and data evacuation in WirelessHART networks. In: The European control conference, pp 4320–4325
2. Zhang H, Osterlind F, Soldati P, Voigt T, Johansson M (2015) Time-optimal convergecast with separated packet copying: scheduling policies and performance. Trans Veh Technol 64:793–803
3. Saifullah A, Xu Y, Lu C, Chen Y (2011) Priority assignment for real-time flows in WirelessHART networks. In: The Euromicro conference on real-time systems (ECRTS), pp 35–44
4. Burns A, Davis RI (2022) Mixed criticality systems - a review. https://www-users.cs.york.ac.uk/burns/review.pdf
5. Baruah S, Vestal S (2008) Schedulability analysis of sporadic tasks with multiple criticality specifications. In: The Euromicro conference on real-time systems (ECRTS), pp 147–155
6. Burns A, Harbin J, Indrusiak LS (2014) A wormhole NoC protocol for mixed criticality systems. In: The real-time systems symposium (RTSS). IEEE, Piscataway, pp 184–195
7. Tobuschat S, Axer P, Ernst R, Diemer J (2013) IDAMC: a NoC for mixed criticality systems. In: The international conference on embedded and real-time computing systems and applications (RTCSA). IEEE, Piscataway, pp 149–156
8. Saifullah A, Xu Y, Lu C, Chen Y (2010) Real-time scheduling for WirelessHART networks. In: Real-time systems symposium (RTSS). IEEE, Piscataway, pp 150–159
9. Zhang HB, Soldati P, Johansson M (2009) Optimal link scheduling and channel assignment for convergecast in linear WirelessHART networks. In: The international symposium on modeling and optimization in mobile, ad hoc, and wireless networks, pp 1–8
10. Chipare O, Lu CY, Roman GC (2013) Real-time query scheduling for wireless sensor networks. Trans Comput 62(9):1850–1865
11. Carvajal G, Fischmeister S (2013) An open platform for mixed-criticality real-time ethernet. In: The design, automation and test in europe conference and exhibition. IEEE, Piscataway, pp 153–156
12. Koubâa A, Alves M, Tovar E, Cunha A (2008) An implicit GTS allocation mechanism in IEEE 802.15.4 for time-sensitive wireless sensor networks: theory and practice. Real-Time Syst 39:169–204
13. Zhan Y, Xia Y, Anwar M (2016) GTS size adaptation algorithm for IEEE 802.15.4 wireless networks. Ad Hoc Netw 37:486–498
14. Huang HM, Gill C, Lu CY (2014) Implementation and evaluation of mixed-criticality scheduling approaches for sporadic tasks. Trans Embed Comput Syst 13(4):1–25
15. Vestal S (2007) Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In: Real-time systems symposium (RTSS). IEEE, Piscataway, pp 239–243
16. Burns A, Fleming T, Baruah S (2015) Cyclic executives, multi-core platforms and mixed criticality applications. In: Euromicro conference on real-time systems (ECRTS). IEEE, Piscataway, pp 3–12
17. Lee J, Phan KM, Gu XZ, Lee JY, Easwaran A, Shin I, Lee I (2014) Mc-fluid: fluid model-based mixed-criticality scheduling on multiprocessors. In: Real-time systems symposium (RTSS). IEEE, Piscataway, pp 41–52
18. Burns A, Davis RI (2013) Mixed criticality on controller area network. In: The Euromicro conference on real-time systems (ECRTS). IEEE, Piscataway, pp 125–134

19. Cros O, Fauberteau F, George L, Li X (2014) Mixed-criticality over switched ethernet networks. In: The workshop on mixed criticality for industrial systems, pp 138–143
20. Addisu A, George L, Sciandra V, Agueh M (2013) Mixed criticality scheduling applied to JPEG2000 video streaming over wireless multimedia sensor networks. In: The 1st workshop on mixed criticality systems. IEEE, Piscataway, pp 1–6
21. Shen W, Zhang T, Barac F, Gidlund M (2014) PriorityMAC: a priority-enhanced MAC protocol for critical traffic in industrial wireless sensor and actuator networks. Trans Ind Inf 10:824–835
22. Hussain SW, Khan T, Zaidi SH (2006) Latency and energy efficient MAC (LEEMAC) protocol for event critical applications in WSNs. In: The international symposium on collaborative technologies and systems, pp 370–378
23. Liang W, Zhang XL, Xiao Y, Wang F, Zeng P, Yu HB (2011) Survey and experiments of WIA-PA specification of industrial wireless network. Wirel Commun Mob Comput 11(8):1197–1212
24. IEC (2016) Industrial networks - wireless communication network and communication profiles - WirelessHARTTM. International Electrotechnical Commission
25. Li B, Nie LS, Wu CJ, Gonzalez H, Lu CY (2015) Incorporating emergency alarms in reliable wireless process control. In: ACM/IEEE international conference on cyber-physical systems. ACM/IEEE, Piscataway, pp 218–227
26. Baruah S, Bonifaci W, DAngelo G, Li HH, Marchetti-Spaccamela A, Megow N, Stougie L (2012) Scheduling real-time mixed-criticality jobs. Trans Comput 61(8):1140–1152
27. De Moura L, Bjorner N (2008) Z3: an efficient SMT solver. In: Tools and algorithms for the construction and analysis of systems. Springer, Berlin, pp 337–340
28. Dutertre B, De Moura L (2006) Tools and algorithms for the construction and analysis of systems. In: Satisfiability modulo theories competition, pp 1–3
29. Pellizzoni R, Paryab N, Yoon MK, Bak S, Mohan S, Bobba RB (2015) A generalized model for preventing information leakage in hard real-time systems. In: Real-time and embedded technology and applications symposium (RTAS). IEEE, Piscataway, pp 271–282
30. Guan N, Stigge M, Wang Y, Yu G (2009) New response time bounds for fixed priority multiprocessor scheduling. In: Real-time systems symposium. IEEE, Piscataway, pp 387–397
31. Saifullah A, Xu Y, Lu CY, Chen YX (2011) End-to-end delay analysis for fixed priority scheduling in WirelessHART networks. In: Real-time and embedded technology and applications symposium. pp 13–22
32. Joseph M, Pandya P (1986) Finding response times in a real-time system. Comput J 29(5):390–395
33. Camilo T, Silva JS, Rodrigues A, Boavida F (2007) Gensen: a topology generator for real wireless sensor networks deployment. In: The international conference on software technologies for embedded and ubiquitous systems. Springer, Berlin, pp 436–445
34. Bini E, Buttazzo CC (2005) Measuring the performance of schedulability tests. Real-Time Syst 30(1):129–154