Received October 16, 2017, accepted November 6, 2017, date of publication November 10, 2017,
date of current version December 22, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2772328

Expected Completion Time Aware Message Scheduling for UM-BUS Interconnected System

JIQIN ZHOU[1,2], WEIGONG ZHANG[3,4], KENI QIU[1,3], RUIYING BAI[1], YING WANG[3], AND XIAOYAN ZHU[3]

[1]College of Information Engineering, Capital Normal University, Beijing 100048, China
[2]School of Mathematical Sciences, Capital Normal University, Beijing 100048, China
[3]Beijing Advanced Innovation Center for Imaging Technology, Capital Normal University, Beijing 100048, China
[4]Beijing Engineering Research Center of High Reliable Embedded System, Capital Normal University, Beijing 100048, China

Corresponding author: Weigong Zhang (zwg771@cnu.edu.cn).

This work was supported in part by the National Natural Science Foundation of China under Grant 61472260, Grant 61402302, Grant 61502321, Grant 61170009, in part by the Project of Construction of Innovative Teams and Teacher Career Development for Universities and Colleges Under Beijing Municipality under Grant IDHT20150507, and in part by the Project of Beijing Municipal Education Commission under Grant KM201710028016.

ABSTRACT Predictable message transmission is the primary requirement in networked safety critical embedded systems design. In these systems, delay jitter has been proven to be a critical factor that must be considered. For periodic messages, minimizing the delay jitter means messages should be transmitted at the expected time in every period. In this paper, we investigate the scheduling problem to reduce the delay jitter for periodic messages in networked safety critical embedded systems. Our approach is empirically assigning an expected completion time as a baseline for the periodic messages and minimizing the total deviation to them. It can be applied either in centralized control buses or in synchronized ones. This paper selects a novel bus protocol and UM-BUS, to evaluate the effectiveness of the proposed algorithm. UM-BUS is a multi-master bus with the capability of multi-lane concurrent transmission. Aiming at different operation mode of UM-BUS, we implemented two sets of experiments by configuring different parameters to change the bus utilization. The results show that the heuristic algorithm works effectively and can achieve a deviation within 0.35%, which is significantly smaller comparing with the existing scheduling algorithms.

INDEX TERMS UM-BUS, delay jitter, embedded system, safety critical, message scheduling algorithm, EDF.

I. INTRODUCTION

Networked embedded systems generally comprise multiple spatially distributed nodes which support processing information from multiple sensors or actuators. These distributed nodes typically exchange messages in a network. For these systems, control performance strongly depends on the network time delays (or latencies) [1], [2]. Especially in many safety critical embedded systems, such as aerospace, industrial control and automotive electronics, real-time responses of the messages are typically essential requirements [3]. Real-time means the transmissions of messages are expected to complete at predictable time. For periodic messages, the delay jitter, i.e., the deviation of transmission time from the periodicity plays a key role in real-time quality [4], [5]. It is necessary to minimize the delay jitter and transmit the messages at the expected time to enhance the control accuracy of embedded systems.

There are a variety of factors in an embedded system that can affect the delay jitter, among which the message scheduling strategy has an important influence on it. In the current real-time embedded systems, most of the message scheduling algorithms take into account the fulfillment of message deadlines. For example, it is well known that the Earliest Deadline First (EDF) and the Least Laxity First (LLF) strategy are two of the most widely used classic scheduling algorithms in real-time systems [6]–[8]. These scheduling algorithms can effectively satisfy the deadlines of a large-scale message set and have the advantages of high flexibility and bandwidth utilization. However, they can not guarantee the minimum delay jitter because of the concurrent transmission requests of multiple messages with different deadlines.

In order to solve this problem, there are some recent works trying to decrease the delay jitter by changing deadlines.

In [9]–[11], a straight and effective way of reducing delay jitter is presented that assigning tighter deadlines rather than looking them as the same as their periods. Although shorter deadline is helpful to make a specific message obtain higher priority, it impairs the schedulability of the entire message set and may make the message set non-schedulable if the bandwidth utilization is high. Besides, this method cannot sufficiently explore the design space where deadlines of certain messages may be increased (within the upper limit) to reduce the overall delay jitter. In [12], the delay jitter reduction problem is formulated as an optimization problem to identify more appropriate deadline assignments. It proposed a heuristic to adjust the deadlines dynamically as the workload changes. For EDF scheduling, the deadline assignment algorithms are proposed in [13] and [14] to improve the schedulability. Meanwhile, the influence of release jitter in EDF was studied, and it confirmed that jitter avoidance techniques can significantly increase the scheduling capabilities of EDF scheduling. A common theme of all these methods is to focus on adjusting deadlines through different strategies and then scheduling by EDF. Assigning different deadlines to change the priority of messages under EDF scheduling can change the transmission order of concurrent messages. The overall delay jitter can be reduced accordingly. However, since schedulability is closely related to the deadlines of messages, changing them will directly impact the schedulability and increase the complexity of the schedulability analysis methods. In addition, these methods cannot obtain the minimum delay jitter. There are still plenty of rooms for optimization. For example, delay jitter can also be decreased by changing the transmission interval of messages.

Another set of common used methods to smooth out delay jitter is to deploy play-back buffers in the physical system, e.g., [15]–[17]. The essence of play-back buffers is that control messages are saved by the actuator for a certain time before being executed, which is called play-back delay. These methods can effectively smooth out the delay jitter and improve the real-time performance and stability of embedded system. However, to fulfill these strategies, the chosen value of the play-back delay is very important and how to reasonably choose it is complex. Besides, play-back buffers will increase the hardware costs. Since this paper aims to establish a simple and efficient solution to decrease delay jitter, this policy is left outside the scope.

As we all know, delay jitter is closely related to the bus bandwidth utilization in networked embedded systems. Based on the requirements of performance and cost in safety critical embedded systems, the scheduling strategies are expected to maximize the bandwidth utilization and meanwhile to minimize the delay jitter. There are plenty of works focusing on the message scheduling problem to improve the bandwidth utilization or the delay jitter for various buses like CAN, FlexRay, EtherCAT, etc. In [18]–[20], message scheduling strategies for CAN and FlexRay are proposed to maximize bandwidth utilization. In [21], real-time performance of EtherCAT has been studied. In [22], scheduling algorithms are provided to optimize both bandwidth utilization and the jitter of periodic messages. As most of the buses in embedded systems adopt TDMA scheme, these two objectives are contradicted. The space to optimize the delay jitter is becoming smaller along with the growing of the bandwidth utilization. However these current works did not consider the relationship between delay jitter and bandwidth utilization.

In this paper, we investigate the scheduling issues in networked safety critical embedded systems. The research goal is to decrease the message delay jitter by empirically assigning specific expected completion time as the baseline and minimizing the total deviation to them. We first formulate the scheduling problem as a constrained optimization problem with an objective of minimizing the total deviation to expected completion time. On the basis of ensuring schedulability, the optimal transmission order and interval of messages can be identified to attain minimum delay jitter. As this problem is NP-hard, then we propose an efficient heuristic algorithm to achieve near optimal solution. With this heuristic, we first create an initial schedule under EDF scheduling and then get the optimal schedule by improving it iteratively according to the baseline of messages. Our approach has no need of changing deadlines but adjusting the start time of message transmission in order to minimize the gap to baseline. Compared with the above mentioned methods of adjusting deadlines to reduce delay jitter, our method can explore more sufficient design space to decrease delay jitter without affecting the schedulability. In order to evaluate the effectiveness of the proposed algorithm, we implement the experiments in an embedded system design based on a novel high-speed serial bus that our previous work presented named UM-BUS [23]. But not limited to UM-BUS, the proposed algorithm in this paper can be applied in the other centralized control buses or synchronized ones.

UM-BUS is a MLVDS based multi-master bus designed for safety critical embedded system. It can configure multiple lanes to transmit data concurrently to achieve a high bandwidth of up to 6.4Gbps. These lanes are also redundant backup of each other to tolerate faults. Based on this feature, we can change the bandwidth utilization by configuring redundant lanes for UM-BUS, that makes it convenient to evaluate the optimization effects of delay jitter under different bandwidth utilization. So in this paper, we implement multiple sets of experiments with different UM-BUS lanes to validate the proposed algorithm and measure the relationship between delay jitter and bandwidth utilization. On the other hand, considering that different message sets can also change the bandwidth utilization, the other sets of experiments are implemented to test the delay jitter under different message sets. The experimental results show that the proposed heuristic algorithm is efficient and can limit the delay jitter within 0.35%.

The organization of this paper is as follows. First we introduce the UM-BUS protocol in Section II. Then in Section III, we present the system model and formulate the scheduling problem. In Section IV, the heuristic algorithm is formulated

TABLE 1. **The protocol packet format of UM-BUS.**

Command Header (16B)								Data (1025B)	
1B	1B	1B	6B	4B	1B	1B	1B	1024B	1B
Destination Node	Source Node	Command Frame	Address Offset	Short-packet Data	Acknowledge Command	Acknowledge Status	Command CRC	Long-packet Data	Data CRC

and described. In Section V, we design two groups of experiments with different parameters to test the effectiveness of the proposed heuristic algorithm under different bus bandwidth utilization. Finally, conclusions are given in Section VI.

II. THE UM-BUS PROTOCOL

A. GENERAL OVERVIEW

UM-BUS [23] is a kind of master-slave serial bus for embedded system. It adopts bus topology where the distributed nodes of embedded system can be interconnected directly as shown in Figure 1. All nodes of UM-BUS are divided into two sets including master nodes and slave nodes, among which only the master nodes can initiate a communication process. The slave nodes response to the command of the master and perform the corresponding operation.

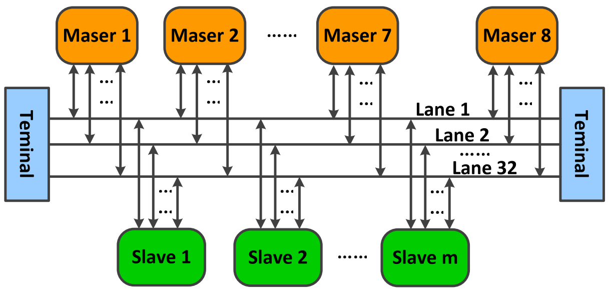


FIGURE 1. **The topology of UM-BUS.**

UM-BUS is based on MLVDS (Multi-point Low Voltage Differential Signaling) and uses multiple lanes (ranging from 2 to 32) to transmit data concurrently. Meanwhile these lanes are also redundant backup of each other. Normally, the transmission data are allocated to all lanes by the bus controller. However, if one or a few lanes failed, the bus controller can detect the fault and allocate the data to the other healthy lanes. Thus, multiple failures in lanes or circuits of nodes can be dynamically tolerated. By this mechanism, the total transmission rate of UM-BUS can easily be changed by configuring different number of lanes. This makes it convenient to implement corresponding experiments under different bandwidth utilization.

UM-BUS uses master-slave command-response communication protocol to transmit data. In the process of bus communication, all bus accesses are performed in the form of exchanging packets between the master and the slave. As shown in Figure 2, the whole process of transmission consists of the following three parts:

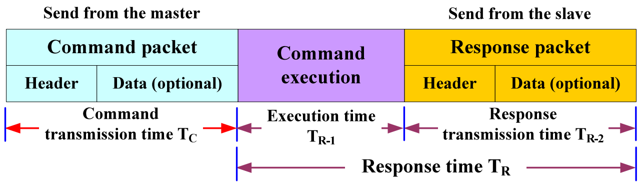


FIGURE 2. **The transmission process of UM-BUS.**

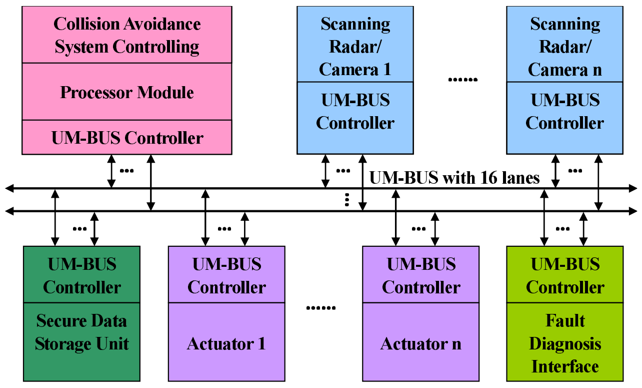


FIGURE 3. **A design of automotive collision avoidance system based on UM-BUS.**

- First, the master node sends a command packet to the slave;
- Then, the slave node receives the packet and completes the relevant read or write operations according to the command;
- Finally, the slave sends a response packet including communication status and data to the master.

According to the UM-BUS protocol, both the command packets and the response packets are divided into two classes including short packets and long packets, as shown in Table 1. A short packet is used to transmit control messages with a 16-byte format. A long packet is used to transmit bulk data with a 1041-byte format for information such as image acquisition data.

In comparison to existing protocols such as CAN or FlexRay, UM-BUS not only has the advantage of higher bandwidth and reliability, but also can simplify the processing model of embedded system by minimizing the Electronic Control Units (ECUs) or distributed processors in the system [24], [25]. A design of automotive collision avoidance system based on UM-BUS is presented in Figure 3. In this system, UM-BUS can achieve a transmission rate of up to 3.2Gbps with 16 lanes. In addition, most of the distributed nodes have no processors or ECUs, such that they are all controlled by the Processor Module (the pink block in Figure 3).

Figure 3 describes an architecture of single-master UM-BUS system. When there is only one master in the UM-BUS system, all the slave nodes are controlled by this master that constructs a physically distributed but logically centralized embedded system architecture. In this case, the message scheduling strategy is the same as that for the centralized embedded system. That means there are much more flexibility to adjust the duration time of messages to decrease the delay jitter.

B. UM-BUS ARBITRATION MECHANISM

According to the protocol of UM-BUS, up to 8 masters can be contained in an embedded system. When there are more than one master in the system, these multiple masters employ the flexible TDMA (FTDMA) approach to allocate the right of bus usage. The mechanism of UM-BUS multi-master arbitration is shown in Figure 4.

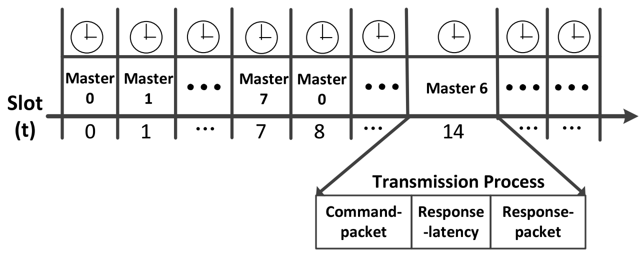


FIGURE 4. The multi-master arbitration mechanism of UM-BUS.

The operation of multi-master UM-BUS is based on a repeatedly executed cycle which comprises of multiple slots, each corresponding to a master. The smallest time unit of the cycle is the minislot (MS) with a duration determined by the maximal transmission time of bus signal through various master nodes and also the asynchronous degrees of them. For example, the maximum transmission distance of UM-BUS is 40 meters and the error of the crystal oscillator adopted in UM-BUS controller is 40ppm. Moreover, the synchronization among the master nodes is done every three milliseconds. Therefore, the minislot can be set as 500ns. The duration of minislot is calculated by the following equation.

$$T_{MS} = (\frac{D_M \cdot 2}{0.3 \cdot 10^9} + E \cdot 2 \cdot S_T)ns \tag{1}$$

where D_M is the maximum transmission distance among the master nodes, E is the error of the crystal oscillator of UM-BUS controller and S_T is the synchronization frequency of the master nodes.

If there are no transmission requests, then every master only take up a period of one minislot. However, when a master need to transmit a packet, the corresponding minislot will extend to a dynamic slot within which one UM-BUS protocol packet is transmitted. The dynamic slot (DS) use an ID to indicate the master that has the right of bus usage. If the total number of UM-BUS masters is denoted as N_M, then the DS with an ID x is mapped to the master $M_{DS,x}$, where

$$M_{DS,x} = x\%N_M. \tag{2}$$

In each dynamic slot, a packet from the corresponding master is transmitted if present. In that case, the duration of the dynamic slot is determined by the length of the transmitted packet. Otherwise, the duration of the dynamic slot is one MS. The duration of DS with ID x is denoted as $T_{DS,x}$. It is calculated as follows.

$$T_{DS,x} = \begin{cases} T_{CR} & \text{if a packet is transmitted} \\ T_{MS} & \text{if no packet is transmitted} \end{cases} \tag{3}$$

In (3), T_{CR} denotes the process time of an UM-BUS communication request. According to the transmission scheme of UM-BUS described in Section II-A, T_{CR} is divided into three parts as shown in Figure 2. In this case, T_{CR} is given by

$$T_{CR} = T_C + T_{R-1} + T_{R-2}. \tag{4}$$

Here, T_C, T_{R-1} and T_{R-2} represent the transmission time of command-packet, the response latency of slave node and the transmission time of response-packet, respectively. The transmission time of a packet is determined by its length which is 16-byte for short-packet or 1041-byte for long-packet, while the response latency depends greatly on what type of information the slave node dealing with.

III. SYSTEM MODEL AND PROBLEM DEFINITION

In this section, we first present the system model. Then considering the situation of single-master UM-BUS, the corresponding message model is proposed together with a motivational example to illustrate the underlying idea of our work. Meanwhile, the target problem is formulated as an optimization problem for single-master UM-BUS. Finally, the message model and problem definition are extended to solve the same problem of multi-master UM-BUS.

A. SYSTEM MODEL

In this paper, a networked real-time embedded system based on UM-BUS is composed of a set of periodic messages, each statically assigned to the master nodes connected to the bus. These messages are typically control messages and have deadlines that can not be violated. The messages are expected to complete their transmission at their respective optimal time. The messages are queued by software tasks running on the host CPU of the master nodes which are invoked by some events [26].

The response time of a message can be divided into two parts, software processing time and transmission time in the bus. The software processing time is denoted as the queuing jitter which is the time lag between the corresponding task invoked by an event and subsequent queuing of the message [27]. In this paper, we focus on the transmission performance and message scheduling problem of UM-BUS. For this purpose, we assume the response time of a message is equal to its transmission time.

B. MESSAGE MODEL FOR SINGLE-MASTER UM-BUS

We consider a set of n periodic messages $M = \{m_1, m_2, \ldots, m_n\}$ communicated in the single-master UM-BUS embedded system. Every invocation of a message is referred as an instance. All messages are assumed independent and the first instances of them are ready at time *0*. Each message m_i has a 5-tuple of parameters (R_i,T_i,D_i,P_i,E_i) where:

- R_i denotes the release time of m_i. For every instance of a message, we assume the release time is at the beginning of the period.
- T_i denotes the period of m_i.
- D_i denotes the deadline of m_i. We assume messages have implicit deadlines equal to their period (i.e. D_i=T_i).
- P_i denotes the response time of m_i. As mentioned before, it is equal to the transmission time in the bus.
- E_i denotes the expected completion time of m_i. In every period of m_i, E_i is assumed at the fixed time before the deadline.

The hyperperiod of the message set is denoted as T_{lcm}, which is the least common multiple of T_i for $i = 1, 2, \ldots, n$. We will focus on the scheduling problem within one hyperperiod because of the periodicity of messages. The repetition times f_i of message m_i in one hyperperiod and the total number of instances N are calculated by the following equations respectively.

$$f_i = T_{lcm}/T_i \tag{5}$$

$$N = \sum_{i=1}^{n} f_i \tag{6}$$

In a hyperperiod, the invocation instance of m_i is denoted as m_i^j ($j = 1, 2, \ldots, f_i$). For m_i^j, it is associated with release time R_i^j, deadline D_i^j, expected completion time E_i^j and response time P_i which is equal for every instance of m_i. According to the message definition, these parameters are derived from the initial 5-tuple (R_i, T_i, D_i, P_i, E_i) of m_i.

C. MOTIVATIONAL EXAMPLE

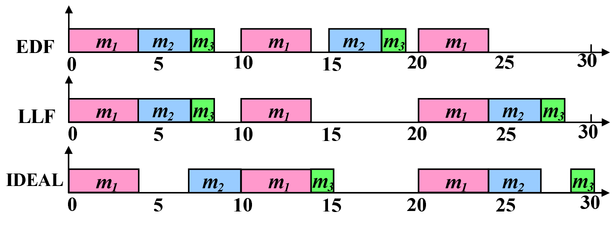
We use a simple example with 3 periodic messages to illustrate the motivation of the target problem. Table 2 shows the parameters of the message set. In the column *Release time*, *Deadline* and *Expected time* of the table, R_i^j, D_i^j and E_i^j are presented for all the innovation instances of these three messages. Here the expected time of m_i^j is set as $\{R_1^j + 4, R_2^j + 10, R_3^j + 15\}$ with $j = 1, 2, \ldots, f_i$.

TABLE 2. Message parameters of the example system.

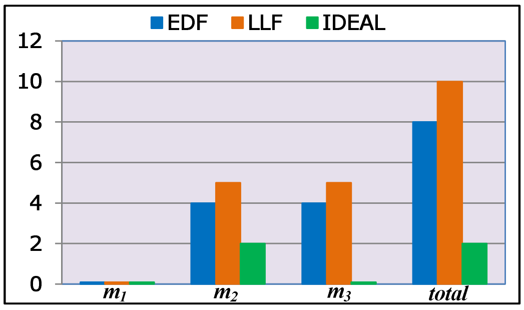
Message	Period	Response time	Release time	Deadline	Expected time
m_1	10	4	0,10,20	10,20,30	4,14,24
m_2	15	3	0,15	15,30	10,25
m_3	15	1	0,15	15,30	15,30

We first use Earliest Deadline First (EDF) and Least Laxity First (LLF) strategies to schedule the message set in Table 2. Without considering the time predictability requirements of real-time embedded systems, these two classical algorithms can not schedule the messages to satisfy both the deadline and the delay jitter. To address this issue, we use the total deviation between actual completion time and expected completion time as the objective of optimization. Meanwhile all the instances should not violate their deadlines. Adopting this optimized design, the timing predictability of the system can be guaranteed better.

Figure 5(a) shows the runtime behavior of the messages scheduled by EDF and LLF. The third one is the ideal case that our approach want to achieve. Figure 5(b) compares the timing predictability of these three cases. The vertical axis stands for the delay jitter value. The smaller the vertical value represents the better the timing predictability.



(a)



(b)

FIGURE 5. A motivation example under different scheduling algorithm.

D. PROBLEM DEFINITION FOR SINGLE-MASTER UM-BUS

In this paper, we expect to find the optimal starting time of every invocation instance within a hyperperiod during message scheduling. The objective is to minimize their total deviation between the expected completion time and the actual completion time, while meeting their deadlines. We use S_i^j to represent the starting time of m_i^j. Based on the given parameters, the specific formulation for single-master UM-BUS is described as follows.

$$obj.: \; min. \sum_{i=1}^{n} \sum_{j=1}^{f_i} |S_i^j + P_i - E_i^j| \tag{7}$$

$$s.t.: R_i^j \leqslant S_i^j \leqslant D_i^j - P_i \tag{8}$$

$$\nparallel (S_i^j, S_i^j + P_i) \tag{9}$$

where $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, f_i$, and $\nparallel$ denotes all the intervals $(S_i^j, S_i^j + P_i)$ are non-overlapping.

E. EXTENDED MODEL FOR MULTI-MASTER UM-BUS

In the UM-BUS system with single-master, all the messages are queued by the host CPU of master without competition to other nodes, whereas the transmission mode of multi-master UM-BUS is very different. According to the multi-master arbitration mechanism of UM-BUS, a message can only be transmitted within dynamic slots that are extended from the minislots assigned to the corresponding master.

On this occasion, we also consider a set of n periodic messages $M = \{m_1, m_2, \ldots, m_n\}$ where each message corresponds to a master. So on the basis of the initial 5-tuple $(R_i, T_i, D_i, P_i, E_i)$, the message model is extended for multi-master UM-BUS by adding a parameter Q_i to denote the master number. The maximal value of Q_i depends on the total number of masters in the system, which is defined as N_M.

In a hyperperiod T_{lcm}, the total number of dynamic slots denoted as N_{DS} and the number of dynamic slots belong to master Q_i denoted as $N_{DS,i}$ are computed by

$$N_{DS} = N + \frac{T_{lcm} - \sum_{i=1}^{n} f_i \cdot P_i}{T_{MS}} \tag{10}$$

$$N_{DS,i} = N_{DS}/N_M. \tag{11}$$

Hence, for message m_i, the IDs of feasible dynamic slots (DS) are as $D = \{(Q_i + N_M \cdot n_x) | n_x = 0, 1, 2, 3, \ldots, N_{DS,i}\}$.

The begin time $B_{DS,x}$ and end time $E_{DS,x}$ of DS with an ID x are described as

$$B_{DS,x} = \sum_{i=1}^{x-1} T_{DS,i} \tag{12}$$

$$E_{DS,x} = B_{DS,x} + T_{DS,x}. \tag{13}$$

Based on the this computation, we introduce the set U_M to represent the feasible occupied intervals of m_i as $U_M = \{(B_{DS,x}, E_{DS,x}) | x \in D\}$.

According to the message model of multi-master UM-BUS described above, the problem definition is somehow different compared to single-master UM-BUS. The optimization objective is invariable for single-master or multi-master UM-BUS. However, the constraint conditions are extended based on (8) and (9) because of the limits of dynamic slots. Therefore, the optimization problem for multi-master UM-BUS can be formulated as follows.

$$obj.: \ min. \sum_{i=1}^{n} \sum_{j=1}^{f_i} |S_i^j + P_i - E_i^j| \tag{14}$$

$$s.t.: \ R_i^j \leqslant S_i^j \leqslant D_i^j - P_i \tag{15}$$

$$\nparallel (S_i^j, S_i^j + P_i) \tag{16}$$

$$(S_i^j, S_i^j + P_i) \in U_M. \tag{17}$$

Here the parameters in (14), (15) and (16) are same as that in (7), (8) and (9) for single-master UM-BUS.

On the basis of the above models for UM-BUS whether it is single-master or multi-master, we define the target optimization problem as Minimize Expected Completion Time Deviation (MECTD) problem. This problem is an extension of the job shop scheduling problem which is known to be NP-hard [28], [29]. Therefore, the next section proposes a heuristic algorithm to solve the problem with more time efficiency.

IV. SCHEDULING ALGORITHM

In this section, a heuristic algorithm is presented to solve the MECTD problem. We first use the Earliest Deadline First (EDF) strategy to schedule the message set. As the EDF algorithm take into account the schedulability of messages, the deadlines of all messages can be guaranteed. However, without considering the timing sensitive requirements of real-time embedded system, it can not schedule the messages to satisfy both the deadlines and minimum delay jitter at the same time. Based on this consideration, we then use the total deviation to expected completion time of messages as the optimization objective to improve the initial schedule. Meanwhile all the messages should not violate their deadlines. Adopting this optimized design, the timing predictability of the system can be satisfied better. The proposed algorithm consists of two stages:

- In the first stage, we apply EDF scheme to create the initial schedule.
- In the second stage, the initial schedule is improved by executing the improvement procedure iteratively until the total deviation cannot be reduced any more.

As shown in A.1, our heuristic algorithm is defined as Optimization Completion Time Based on EDF Algorithm (OCTBEA). It first generates the 5-tuple $(R_i^j, T_i, D_i^j, P_i, E_i^j)$ of m_i^j according to the input parameters (line 5). Then an instance sequence I in a hyperperiod is formed (line 8). Next, Procedure *EDF_Schedule* is called to schedule I based on the EDF scheme (line 9). Finally, Procedure *Improvement* is called iteratively to further improve the schedule (line 12).

A. 1 Optimization Completion Time Based on EDF Algorithm (OCTBEA)

```
Require: $M = \{m_1, m_2, \ldots, m_n\}$
1:  Calculate the hyperperiod $T_{lcm}$ of $M$;
2:  for $i = 1; i \leqslant n; i++$ do
3:      $f_i = T_{lcm}/T_i$;
4:      for $j = 1; j \leqslant f_i; j++$ do
5:          Generate 5-tuple $(R_i^j, T_i, D_i^j, P_i, E_i^j)$ of $m_i^j$;
6:      end for
7:  end for
8:  Form the instance sequence:
    $I = \{m_i^j | i = 1, 2, \ldots, n; j = 1, 2, \ldots, f_i\}$;
9:  Call EDF_Schedule($I, S$);
10: while $S'$ is better than $S$ do
11:     $S = S'$;
12:     Call Improvement($S, S'$);
13: end while
```

In the following, we first take the single-master UM-BUS as an example to give the implementation details and the formal descriptions of the procedures *EDF_Schedule* and *Improvment*. After that, the implementation of these two procedures for multi-master UM-BUS is presented by comparison to the single-master one.

A. SCHEDULING ALGORITHM FOR SINGLE-MASTER UM-BUS

For single-master UM-BUS, the two stages of OCTBEA can be formally described as follows.

1) STAGE I: CREATING AN INITIAL SCHEDULE

In order to guarantee the schedulability of the messages, we first form an initial schedule according to EDF scheme. Then, a batch sequence is defined where each batch B_q contains the instances being transmitted in succession. For example, if the start time of message instance m_2^1 is the same as the completion time of another message instance m_1^1. Then m_1^1 and m_2^1 belong to the same batch. The definition of batch sequence is to enhance the efficiency and decrease the time complexity of the subsequent procedures in the second stage. Based on the rules of batch construction, the following variables can be calculated to record the batch sequence, as shown in Table 3.

The formal description of procedure *EDF_Schedule* is shown in A.2.

A. 2 Procedure EDF_Schedule(I, S) Generate Initial Schedule and Batch Sequence

```
Require: $I = \{m_i^j | i = 1, 2, \ldots, n; j = 1, 2, \ldots, f_i\}$
 1: Create initial schedule $S$ according to EDF rules;
 2: Renumber the instances as $\{I_1, I_2, \ldots, I_N\}$ according to $S$;
 3: $q = 1$;
 4: $B_q = \varnothing$;
 5: $TDEA = 0$;
 6: while There are untreated instances in $S$ do
 7:     $S_q = min\{R_i | R_i$ is the start time of untreated instance $I_i\}$;
 8:     Add to $B_q$ instances satisfied the criteria of batch generation;
 9:     Record the number of instances in $B_q$ as $N_q$;
10:     $E_q = max\{C_i | C_i$ is the completion time of $I_i$ in $B_q\}$;
11:     Calculate $TDEA_q$ of $B_q$;
12:     Remove the instances of $B_q$ from the set of untreated ones;
13:     $TDEA = TDEA + TDEA_q$;
14:     $q = q + 1$;
15: end while
16: $N_B = q - 1$;
```

In Procedure *EDF_Schedule*(I, S), the instance sequence I is scheduled by EDF scheme to create an initial schedule S. Then according to the above-mentioned rules of batch definition, a batch sequence is generated by dividing the schedule into several smaller segments.

TABLE 3. Variable definition of the procedure.

Notation	Description
B_q	The batch number
N_B	The number of batches in the batch sequence
S_q	The start time of batch B_q
E_q	The end time of batch B_q
$TDEA$	The total deviation to expected completion time

2) STAGE II: PROCEDURE IMPROVEMENT

In the first stage, OCTBEA creates an initial schedule which meets the deadline of all messages without considering the optimal Total Deviation between Expected completion time and Actual time (TDEA). In the improvement stage, the TDEA is improved by iteratively performing three procedures in turn as follows:

- Procedure *Move_Batch* optimizes TDEA by moving the original schedule without changing the batch sequence. Every batch of the schedule can be moved to the left or the right to decrease its TDEA. At the same time, the schedulability of messages must be guaranteed;
- Procedure *Exchange_Batch* optimizes TDEA by exchanging the order of the message instances in the batches. For a batch, the best execution interval of each instance is calculated to determine the optimal order of them to decrease TDEA. However, the occupied interval of a batch is invariable and the schedulability can not be violated;
- Procedure *Insert_Batch* optimizes TDEA by inserting idle time in the batches. That is, the adjacent message instances in a batch might be separated for the purpose of decreasing TDEA. After this procedure, the batch will be regenerated according to the criteria mentioned above.

In what follows we give formal descriptions of the improvement procedures.

Procedure *Move_Batch*(S, S') adopts two loops to move the batches in turn to the left and to the right respectively to decrease TDEA. The maximum adjustable interval of a batch is calculated according to its order in the batch sequence and the instances it contained.

Procedure *Exchange_Batch*(S, S') first calculates the deviation of each instance in a batch when they start at the same time and find the minimal deviation along with the corresponding instance. Then the orders of instances are exchanged according to their deviations on condition that the requirement of release time and deadline are satisfied. These processes are repeated until all the batches are traversed.

Procedure *Insert_Batch*(S, S') first traverses the batch sequence to calculate the maximal adjustable intervals of each batch which are denoted as t_1 and t_2. Then in a batch, the maximal insertable intervals of each instance is calculated and compared to t_1 and t_2 respectively. The minimal value of comparative result is the maximal interval what an idle time can be inserted. According to these calculation,

A. 3 Procedure Move_Batch(S,S') Improve the Schedule S by Moving the Batches

```
Require: Schedule $S$
1:  for $q = 1; q \leqslant N_B; q++$ do
2:      if $q = 1$ then
3:          $t_1 = S_q$;
4:      else
5:          $t_1 = S_q - E_{q-1}$;
6:      end if
7:      $t_2 = min\{ML_i | ML_i$ is the maximal adjustable interval to the left of instance $I_i$ in $B_q\}$;
8:      $Max\_l = min\{t_1, t_2\}$;
9:      Move all the instances in batch $B_q$ to the left no more than $Max\_l$ to minimize $TDEA_q$;
10:     Update the interval $[S_q, E_q)$ of batch $B_q$;
11: end for
12: for $q = N_B; q \geqslant 1; q--$ do
13:     if $q = N_B$ then
14:         $t_1 = T_{lcm} - E_q$;
15:     else
16:         $t_1 = S_{q+1} - E_q$;
17:     end if
18:     $t_2 = min\{MR_i | MR_i$ is the maximal adjustable interval to the right of instance $I_i$ in $B_q\}$;
19:     $Max\_r = min\{t_1, t_2\}$;
20:     Move batch $B_q$ to the right no more than $Max\_r$ to minimize $TDEA_q$;
21:     Update the interval $[S_q, E_q)$ of batch $B_q$;
22: end for
```

the idle time T_l to the left or T_r to the right is inserted to decrease TDEA. After that, the batch sequence is regenerated due to the rules of it.

B. SCHEDULING ALGORITHM FOR MULTI-MASTER UM-BUS

For multi-master UM-BUS, the overall process of scheduling algorithm is the same as single-master UM-BUS except the constraints of feasible dynamic slots. On this occasion, the dynamic slot assignment has to be fixed such that all the messages meet their deadlines and achieve the minimal TDEA further. The start time and the duration time of dynamic slot are variable according to the scheme of UM-BUS multi-master arbitration. They are determined by the transmission requests of masters and the corresponding message schedule. Hence, for any dynamic slot x within a hyperperiod, it is necessary to record and update the relevant variables of it including the start time $B_{DS,x}$, the duration time $T_{DS,x}$ and the end time $E_{DS,x}$ in the process of generating the initial schedule and the next iterative improvements. We now describe the algorithm that constructs a schedule first and then improves it respecting the dynamic slots constraints. Here, we only discuss the different part of the algorithm compared to that of single-master UM-BUS.

A. 4 Procedure Exchange_Batch(S,S') Improve the Schedule S by Exchanging the Order of Instances in the Batches

```
Require: Schedule $S$
1:  for $q = 1; q \leqslant N_B; q++$ do
2:      for $i = 1; i \leqslant N_q; i++$ do
3:          The start time of Instance $m_i$ is denoted as $Start\_i$;
4:          for $j = i; j \leqslant N_q; j++$ do
5:              The completion time of Instance $m_j$ is denoted as $End\_j$;
6:              if Instance $m_j$ can start at $Start\_i$ and Instance $m_i$ can complete at $End\_j$ then
7:                  Calculate the deviation of Instance $m_j$ when its start time is $Start\_i$;
8:              end if
9:              Record the Instance $m_x$ which has the minimal deviation;
10:         end for
11:         if $m_x \neq m_i$ then
12:             Exchange the orders of $m_x$ and $m_i$;
13:         end if
14:     end for
15: end for
```

1) STAGE I: CREATING AN INITIAL SCHEDULE

In the first stage, the initial schedule is generated according to EDF policy as that of single-master UM-BUS. The formal description of procedure *EDF_Schedule* is as follows. In this algorithm, the omitting part is same as algorithm A. 2.

Procedure *EDF_Schedule*(I, S) first sorts the instances in accordance with EDF policy. Then dynamic slot assignment is fixed such that all the sorted instances meet their deadlines. The feasible set of dynamic slots is according to the map α_{DS} which determines what dynamic slot can be occupied by an instance. The map α_{DS} is updated every time after a dynamic slot is assigned to an instance. After that, the process of generating batch sequence is omitted because it is same as the procedure of single-master UM-BUS.

2) STAGE II: PROCEDURE IMPROVEMENT

In this stage, the initial schedule is improved by iteratively performing three procedures *Move_Batch*, *Exchange_Batch*, and *Insert_Batch* as that of single-master UM-BUS. However, each improved schedule should guarantee the restrictions of the multi-master UM-BUS protocol specification during the entire iteration process. In contrast to the implementation of single-master UM-BUS, the improvement procedures of multi-master UM-BUS add an additional process to keep the latest status of dynamic slots when scheduling the messages. In the following, we give the formal description of *Move_Batch* as an example to illustrate how to improve the schedule for multi-master UM-BUS by moving the batches. The operation mechanism of the other two procedures are similar to Procedure *Move_Batch*, so they are omitted here.

A. 5 Procedure Insert_Batch(S,S') Improve the Schedule S by Inserting Idle Time in the Batches

```
Require: Schedule $S$
1:  for $q = 1; q \leqslant N_B; q++$ do
2:      if $q = 1$ then
3:          $t_1 = S_q$;
4:      else
5:          $t_1 = S_q - E_{q-1}$;
6:      end if
7:      if $q = N_B$ then
8:          $t_2 = T_{lcm} - E_q$;
9:      else
10:         $t_2 = S_{q+1} - E_q$;
11:     end if
12:     for $j = 1; j \leqslant N_q - 1; j++$ do
13:         $t_3 = min\{ML_i|ML_i$ is the maximal adjustable interval to the left of instance $I_i$ in $B_q$ with $i = 1, ą, j\}$;
14:         $Max\_l = min\{t_1, t_3\}$;
15:         $t_4 = min\{MR_i|MR_i$ is the maximal adjustable interval to the right of instance $I_i$ in $B_q$ with $i = j + 1, ą, N_q\}$;
16:         $Max\_r = min\{t_2, t_4\}$;
17:         Insert idle time $T_l$ to the left no more than $Max_l$ or $T_r$ to the right no more than $Max_r$ to minimize $TDEA_q$;
18:         $t_1 = t_1 - T_l$;
19:         $t_2 = t_2 - T_r$;
20:     end for
21:     Update the interval $[S_q, E_q)$ of batch $B_q$;
22: end for
```

A. 6 Procedure EDF_Schedule(I, S)

```
Require: $I = \{m_i^j|i = 1, 2, \ldots, n; j = 1, 2, \ldots, f_i\}$
1: Sort the instances as $\{I_1, I_2, \ldots, I_N\}$ according to EDF rules;
2: Calculate the total number of dynamic slots $N_{DS}$ in a hyperperiod;
3: Calculate the number of dynamic slots belong to master $Q_i$ and denote it as $N_{DS,Q_i}$ ;
4: Construct the initial map $\alpha_{DS}$: $Q_i \rightarrow \{(B_{DS,x}, E_{DS,x})|x = Q_i, (Q_i + 1 \cdot N_M), (Q_i + 2 \cdot N_M), \ldots, (Q_i + (N_{DS,Q_i} - 1) \cdot N_M)\}, i = 0, 1, 2, \ldots, N_M$;
5: for $i = 1; i \leqslant N; i++$ do
6:     assign the instance $I_i$ a dynamic slot with minimal ID to meet its deadline according to $\alpha_{DS}$;
7:     Update the map $\alpha_{DS}$;
8: end for
9: ......;//The process of generating batch sequence is same as A. 2, so it is omitted here.
```

A. 7 Procedure Move_Batch(S,S') Improve the Schedule S by Moving the Batches

```
Require: Schedule $S$
1:  $t_s = T_{MS}$;
2:  for $q = 1; q \leqslant N_B; q++$ do
3:      if $q = 1$ then
4:          $t_1 = S_q$;
5:      else
6:          $t_1 = S_q - E_{q-1}$;
7:      end if
8:      $N_1 = t_1/t_s$;
9:      $t_2 = min\{ML_i|ML_i$ is the maximal adjustable interval to the left of instance $I_i$ in $B_q\}$;
10:     $N_2 = t_2/t_s$;
11:     $Max\_l = min\{N_1, N_2\}$;
12:     Move all the instances in batch $B_q$ to the left by decreasing their dynamic slots ID no more than $Max\_l$ to minimize $TDEA_q$ according to the map $\alpha_{DS}$;
13:     Update the interval $[S_q, E_q)$ of batch $B_q$;
14:     Update the map $\alpha_{DS}$;
15: end for
16: for $q = N_B; q \geqslant 1; q--$ do
17:     if $q = N_B$ then
18:         $t_1 = T_{lcm} - E_q$;
19:     else
20:         $t_1 = S_{q+1} - E_q$;
21:     end if
22:     $N_1 = t_1/t_s$;
23:     $t_2 = min\{MR_i|MR_i$ is the maximal adjustable interval to the right of instance $I_i$ in $B_q\}$;
24:     $N_2 = t_2/t_s$;
25:     $Max\_r = min\{N_1, N_2\}$;
26:     Move all the instances in batch $B_q$ to the right by decreasing their dynamic slots ID no more than $Max\_r$ to minimize $TDEA_q$ according to the map $\alpha_{DS}$;
27:     Update the interval $[S_q, E_q)$ of batch $B_q$;
28:     Update the map $\alpha_{DS}$;
29: end for
```

Procedure *Move_Batch*(S, S') first gets the duration of minislot T_{MS} for multi-master UM-BUS system. Then the maximal adjustable interval to the left or to the right is computed, which converts to the number of minislots by dividing it by T_{MS}. Next, the batches are moved in turn respectively to decrease TDEA according to the maximal minislots and the map α_{DS}. After that, the occupied interval of the batches and the map α_{DS} are all updated.

The time complexity of OCTBEA is related to the total number of instances N in one hyperperiod. In OCTBEA, generating the initial schedule based on EDF scheme can be done in $O(N^2)$ time. Procedure *Move_Batch*(S, S'), Procedure *Exchange_Batch*(S, S') and Procedure *Insert_Batch*(S, S') incur $O(N)$, $O(N^2)$ and $O(N)$ calculations respectively. The iteration times of the improvement procedures are relevant to the total number of messages. Therefore, the complexity of OCTBEA is $O(N^2)$.

V. EXPERIMENT EVALUATION

In this section, the effectiveness of the proposed scheduling algorithm is evaluated by comparing it to the classical scheduling algorithms EDF and LLF. The two versions of OCTBEA for single-master UM-BUS and multi-master UM-BUS are tested respectively by different experiment sets.

Furthermore, for each algorithm version, two sets of experiments are conducted by configuring different amounts of lanes or messages to test the efficiency of the proposed heuristic algorithm under different bus bandwidth utilizations. These experiments can also evaluate the relationship between the delay jitter and the bandwidth utilization. Combining equations (5) and (6), the bandwidth utilization can be calculated by the following equation:

$$U = \frac{\sum_{i=1}^{n}(f_i \cdot P_i)}{T_{lcm}} \tag{18}$$

where the message transmission time P_i is in proportion with the number of UM-BUS lanes. According to the experiments in the previous work [23], the single lane of UM-BUS can attain the actual transmission rate of 40Mbps. Thus the transmission time of short packet and long packet of UM-BUS can be calculated as follows.

$$P_i = \frac{H_i \cdot 10}{40 \cdot L_i}\, us \tag{19}$$

In equation (19), H_i is 16 for a short packet message or 1041 for a long packet message. It multiplies by 10 because of $8b/10b$ data coding mechanism of UM-BUS. L_i is the number of UM-BUS lanes. Under different configurations of lanes and messages, we apply OCTBEA together with the EDF and LLF algorithms respectively to schedule the messages.

A. EXPERIMENTS FOR SINGLE-MASTER UM-BUS

According to the information exchanging requirements of real-time embedded systems, we present a design of system architecture based on UM-BUS consisting of one master node and six slave nodes as shown in Figure 6. In this single-master UM-BUS system, a set of periodic messages presented in Table 4 needs to be scheduled.

In Figure 6, the slave nodes 4, 5, 6 are I/O modules which are comprised of sensors or actuators to implement signal acquisition and control. According to the packet format definition of UM-BUS protocol, we assign short packet transmission to these nodes. Whereas the slave nodes 1, 2, 3 are image gathering modules which are used to collect image information. We adopt long packet to transmit these information to improve transmission efficiency. The periods of messages in Table 4 are derived from the real-world requirements of real-time embedded system. The expected completion time of message m_i is randomly selected from the set $\{(T_i - P_i)\cdot n/10 | n = 1, 2, \ldots, 10\}$.

1) CASE I: FIXED MESSAGES, DIFFERENT LANES

In this case, we study the impact of different bus utilizations caused by different lanes. The message set is fixed as Table 4,

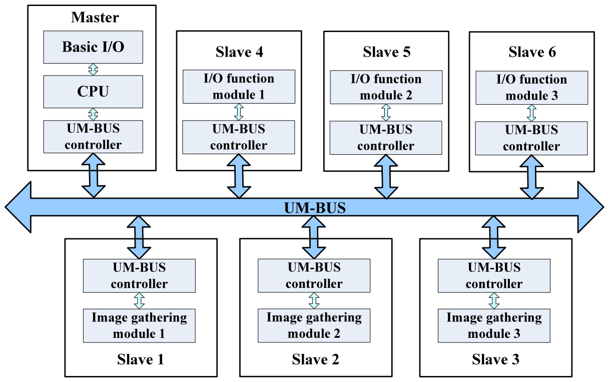


FIGURE 6. The architecture of a real-time embedded system based on single-master UM-BUS.

TABLE 4. The message set of single-master UM-BUS system.

Message	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
node	4	5	4	6	6	1	2	3
period(us)	500	600	2000	1000	800	500	1000	800
type	short	short	short	short	short	long	long	long

while the lanes are configured as 2, 3, 4, 5, 6, 8, 10, 12, 14 and 16 respectively. The first half of Table 5 shows the parameters of these ten experiments. In this table, the caption *#messages*(*short*/*long*) denotes the number of messages in the message set where the figure in bracket denotes the number of short packets and long packets respectively. The caption *#instances* denotes the number of invocation instances in one hyperperiod. The results are shown in Figure 7(a). The vertical axis stands for the Delay Jitter Ratio (DJR) which is calculated by the following equations.

For OCTBEA:

$$DJR = \frac{\sum_{i=1}^{n}\sum_{j=1}^{f_i} |S_i^j + P_i - E_i^j|}{T_{lcm} \cdot n} \tag{20}$$

For EDF or LLF:

$$DJR = \frac{\sum_{i=1}^{n}\sum_{j=1}^{f_i} |S_i^j + P_i - R_i^j - M_{d,i}|}{T_{lcm} \cdot n} \tag{21}$$

$$M_{d,i} = \frac{\sum_{j=1}^{f_i}(S_i^j + P_i - R_i^j)}{f_i} \quad i = 1, 2, \ldots, n \tag{22}$$

DJR represents the ratio of total delay jitter to a hyperperiod multiplying the message number. For OCTBEA, expected completion time E_i^j is the baseline to schedule m_i^j, so the sum of all the deviations of messages to expected completion time is regarded as the total delay jitter. However, for EDF algorithm or LLF algorithm, the delay jitter of m_i^j is the difference between actual delay of m_i^j and average delay of m_i. In equation (21), $M_{d,i}$ stands for the average delay of m_i, which is calculated by equation (22).

TABLE 5. Experimental parameters for single-master UM-BUS.

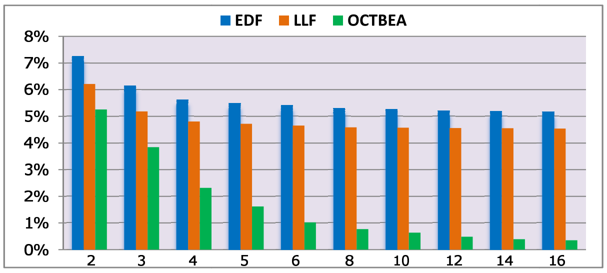
No.		#messages (short/long)	#instances	#lanes	$T_{lcm}(us)$	bandwidth utilization
Case I	1	8(5/3)	128	2	12000	56.5%
	2	8(5/3)	128	3	12000	37.9%
	3	8(5/3)	128	4	12000	28.3%
	4	8(5/3)	128	5	12000	22.9%
	5	8(5/3)	128	6	12000	19.0%
	6	8(5/3)	128	8	12000	14.1%
	7	8(5/3)	128	10	12000	11.5%
	8	8(5/3)	128	12	12000	9.6%
	9	8(5/3)	128	14	12000	8.3%
	10	8(5/3)	128	16	12000	7.1%
Case II	1	16(8/8)	208	4	12000	44.9%
	2	15(8/7)	199	4	12000	40.1%
	3	14(8/6)	186	4	12000	39.4%
	4	12(8/4)	164	4	12000	35.0%
	5	8(5/3)	128	4	12000	28.3%
	6	7(4/3)	106	4	12000	24.8%
	7	6(4/2)	113	4	12000	19.9%
	8	5(3/2)	77	4	12000	15.0%

2) CASE II: FIXED LANES, DIFFERENT MESSAGES

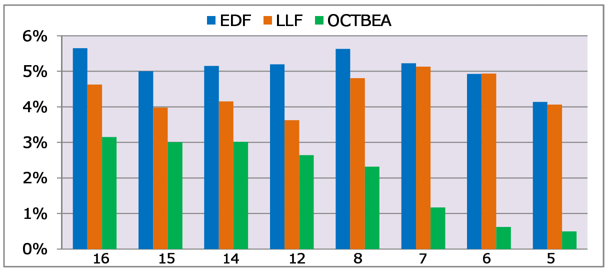
In this case, we study the impact of different bus utilizations caused by different message sets. The lane number of UM-BUS is configured as 4 while the number of messages is increased or decreased on the basis of the message set in Table 4. The lower part of Table 5 shows the parameters of these eight experiments. The results are shown in Figure 7(b).

Figure 7 shows OCTBEA for single-master UM-BUS is efficient in practice. In Figure 7(a), the total delay jitter is getting smaller as bus bandwidth utilization has decreased by increasing UM-BUS lanes. When the bandwidth utilization is 56.5% with 2 lanes, the total delay jitter can be optimized by a ratio of 5.26% along with an improvement of 2% (1%) compared to EDF (LLF). Under high bandwidth utilization, the best execution intervals of messages are probably overlapped with each other, so the optimization effect of OCTBEA is limited. However, when the bandwidth utilization is 7.1% with 16 lanes, the delay jitter ratio can be decreased to 0.35% with an improvement of 4.8% (4.2%) compared to EDF (LLF). In Figure 7(b), the varying trend exhibits the same property as that in Figure 7(a) when the number of messages changes. In order to further illustrate the optimization efficiency of OCTBEA, in Figure 7(c), we compared the delay jitters of a message in experiment No.3 of CASE I. Here, the unit of the vertical axis is microsecond. The results show that the delay jitter can be effectively optimized by OCTBEA compared to EDF or LLF.

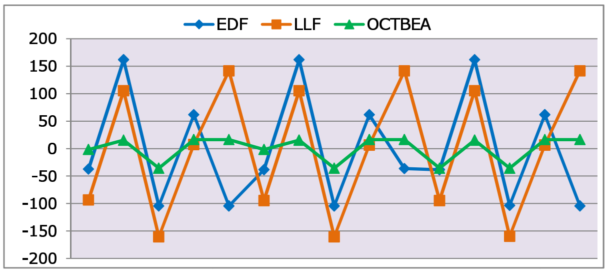
According to the above analysis, OCTBEA exhibits different optimization effect under different bandwidth utilization. The lower the bandwidth utilization, the better the delay jitter can be optimized. The relationship between delay jitter and bandwidth utilization under EDF, LLF and OCTBEA is depicted in Figure 8. For EDF and LLF, the delay jitter is more relevant to message set than bandwidth utilization. The delay jitter does not change appreciably when the bandwidth utilization changes. However, through OCTBEA, bandwidth can be fully used to decrease the delay jitter.



(a)



(b)



(c)

FIGURE 7. Evaluations of EDF, LLF and OCTBEA for single-master UM-BUS.

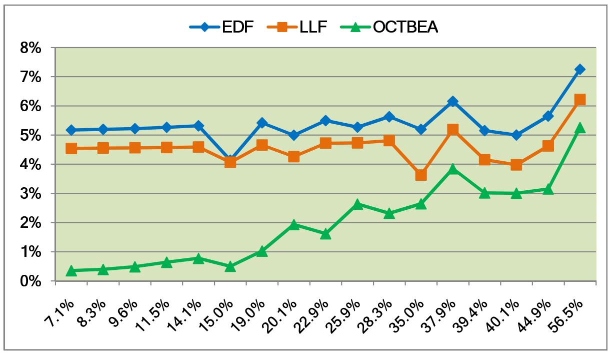


FIGURE 8. The delay jitter ratios under different bandwidth utilization.

B. EXPERIMENTS FOR MULTI-MASTER UM-BUS

In Today's embedded systems, more than one processor is needed because of the increasing controlling scale and

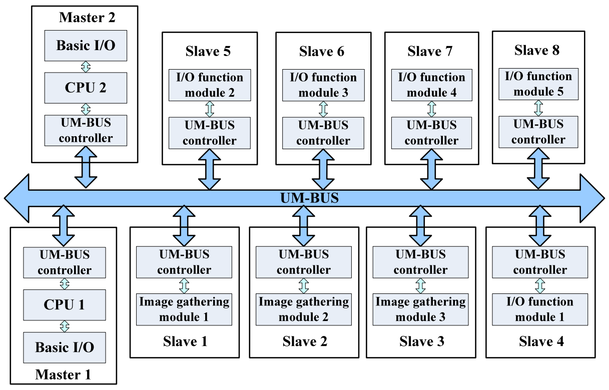


FIGURE 9. The architecture of a real-time embedded system based on multi-master UM-BUS.

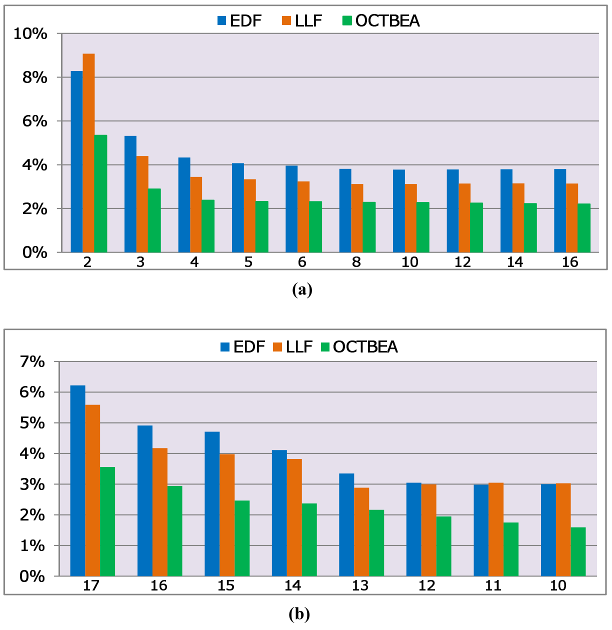
TABLE 6. The message set of multi-master UM-BUS system.

Message	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
master node	1	2	2	1	2	1	1	1
slave node	4	5	4	6	6	1	2	3
period(us)	500	600	2000	1000	800	500	1000	800
type	short	short	short	short	short	long	long	long
Message	m_9	m_{10}	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}	m_{16}
master node	2	2	1	2	1	2	2	1
slave node	7	8	8	5	2	3	1	2
period(us)	500	600	2000	4000	2000	1500	4000	2000
type	short	short	short	short	long	long	long	long

processing capacity of the system. Accordingly, this paper investigates the effectiveness of OCTBEA for multi-master UM-BUS. In this experiment, we present an architecture of embedded system based on multi-master UM-BUS, which contains two master nodes and eight slave nodes as shown in Figure 9.

In the above multi-master UM-BUS system, we assume a set of periodic messages presented in Table 6 need to be scheduled, where the parameter definition and configuration of messages are the same as that of single-master UM-BUS. Based on this message set, two sets of experiments configured with different UM-BUS lanes or different messages are conducted as that for single-master UM-BUS. Table 7 shows the parameters of these two sets of experiments. The results are shown in Figure 10.

Figure 10 shows OCTBEA is effective in decreasing delay jitter for the multi-master UM-BUS system. The varying trend under different lanes or messages presents the same feature as that in Figure 7. For example, in Figure 10(a), when the bus utilization is 75.9% with 2 lanes, the total delay jitter can be optimized by a ratio of 5.35% along with an improvement of 2.94% (3.72%) compared to EDF (LLF). When the bus utilization is 9.5% with 16 lanes, the delay jitter radio can be decreased to 2.21% with an improvement of 1.59% (0.94%) compared to EDF (LLF). In Figure 10(b), when the bus utilization is 53.2% with 17 messages, the delay jitter ratio can be optimized to 3.56% with an improvement



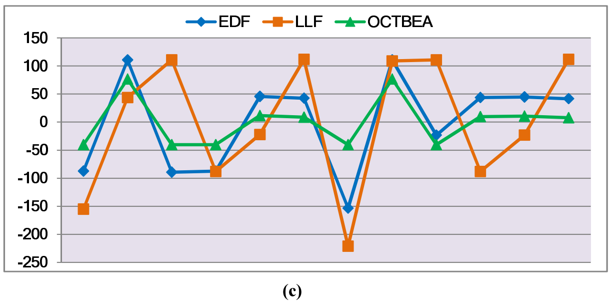


FIGURE 10. Evaluations of EDF, LLF and OCTBEA for multi-master UM-BUS.

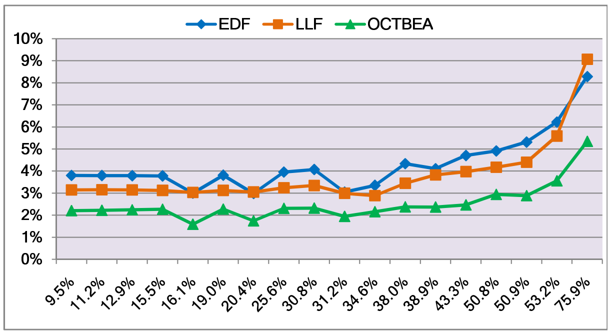


FIGURE 11. The delay jitter ratios under different bandwidth utilization.

of 2.66% (2.03%) compared to EDF (LLF). When the bus utilization is 16.1% with 10 messages, the delay jitter radio can be decreased to 1.59% with an improvement of 1.41% (1.43%) compared to EDF (LLF). Moreover, Figure 10(c) shows the delay jitters of a message obtained by experiment No.4 of CASE II. From this figure, we can see the delay jitter curve of OCTBEA is more smooth than that of EDF and LLF.

The relationship between delay jitter and bandwidth utilization in multi-master UM-BUS system is shown in Figure 11. Compared to Figure 8, the variation curve of

TABLE 7. Experimental parameters for multi-master UM-BUS.

No.		#messages (short/long)	#instances	#lanes	$T_{lcm}(us)$	bandwidth utilization
Case I	1	16(9/7)	204	2	12000	75.9%
	2	16(9/7)	204	3	12000	50.9%
	3	16(9/7)	204	4	12000	38.0%
	4	16(9/7)	204	5	12000	30.8%
	5	16(9/7)	204	6	12000	25.6%
	6	16(9/7)	204	8	12000	19.0%
	7	16(9/7)	204	10	12000	15.5%
	8	16(9/7)	204	12	12000	12.9%
	9	16(9/7)	204	14	12000	11.2%
	10	16(9/7)	204	16	12000	9.5%
Case II	1	17(9/8)	243	4	12000	53.2%
	2	16(8/8)	215	4	12000	50.8%
	3	15(8/7)	205	4	12000	43.3%
	4	14(7/7)	191	4	12000	38.9%
	5	13(7/6)	183	4	12000	34.6%
	6	12(6/6)	165	4	12000	31.2%
	7	11(6/5)	145	4	12000	20.4%
	8	10(6/4)	137	4	12000	16.1%

OCTBEA in multi-master UM-BUS system has not dropped sharply under low bandwidth utilization. Due to the fact that the messages from given master can only occupy the fixed slot, the optimization effects of OCTBEA for multi-master UM-BUS are not as well as that of single-master UM-BUS.

VI. CONCLUSION

UM-BUS is a novel serial bus designed for embedded system with multi-lane concurrent transmissions to enhance the bandwidth. In order to improve the timing sensitive requirements of real-time embedded system on UM-BUS, this paper proposes a scheduling scheme to reduce the total delay jitter of messages. Due to its NP-hardness, a heuristic algorithm is presented. It consists of two stages: creating the initial schedule based on EDF scheme and improving the schedule iteratively. By configuring various UM-BUS lanes or message sets, the bandwidth utilization is changed. In this case, two sets of experiments with different lanes or different message sets are implemented to evaluate the effectiveness of our algorithm. The results show that the proposed algorithm can effectively attain the objective compared to the other classical algorithms.

REFERENCES

[1] E. Ebeid, F. Fummi, and D. Quaglia, "Model-driven design of network aspects of distributed embedded systems," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 4, pp. 603–614, Apr. 2015.

[2] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Communications-oriented development of component-based vehicular distributed real-time embedded systems," *J. Syst. Archit.*, vol. 60, no. 2, pp. 207–220, Feb. 2014.

[3] P. Suriyachai, U. Roedig, and A. Scott, "A survey of MAC protocols for mission-critical applications in wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 2, pp. 240–264, Second Quarter 2012.

[4] K. Schmidt and E. G. Schmidt, "Systematic message schedule construction for time-triggered CAN," *IEEE Trans. Veh. Technol.*, vol. 56, no. 6, pp. 3431–3441, Nov. 2007.

[5] A. Aminifar, P. Eles, Z. Peng, and A. Cervin, "Stability-aware analysis and design of embedded control systems," in *Proc. Int. Conf. Embedded Softw. (EMSOFT)*, Sep./Oct. 2013, pp. 1–10.

[6] P. Uthaisombut, "Generalization of EDF and LLF: Identifying all optimal online algorithms for minimizing maximum lateness," *Algorithmica*, vol. 50, no. 3, pp. 312–328, Mar. 2008.

[7] A. Burns, M. Gutiérrez, M. A. Rivas, and M. G. Harbour, "A deadline-floor inheritance protocol for EDF scheduled embedded real-time systems with resource sharing," *IEEE Trans. Comput.*, vol. 64, no. 5, pp. 1241–1253, May 2015.

[8] A. A. Ayele, V. S. Rao, K. G. Dileep, and R. K. Bokka, "Combining EDF and LST to enhance the performance of real-time task scheduling," in *Proc. Int. Conf. ICT Bus. Ind. Government (ICTBIG)*, Nov. 2016, pp. 1–6.

[9] P. Balbastre, I. Ripoll, and A. Crespo, "Minimum deadline calculation for periodic real-time tasks in dynamic priority systems," *IEEE Trans. Comput.*, vol. 57, no. 1, pp. 96–109, Jan. 2008.

[10] E. Bini and G. Buttazzo, "The space of EDF deadlines: The exact region and a convex approximation," *Real-Time Syst.*, vol. 41, no. 1, pp. 27–51, Jan. 2009.

[11] K. Tanaka, "Real-time scheduling for reducing jitters of periodic tasks," *J. Inf. Process.*, vol. 23, no. 5, pp. 542–552, 2015.

[12] S. Hong, X. S. Hu, and M. D. Lemmon, "Reducing delay jitter of real-time control tasks through adaptive deadline adjustments," in *Proc. 22nd Euromicro Conf. Real-Time Syst.*, Jul. 2010, pp. 229–238.

[13] J. M. Rivas, J. J. Gutiérrez, J. C. Palencia, and M. G. Harbour, "Deadline assignment in EDF schedulers for real-time distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 10, pp. 2671–2684, Oct. 2015.

[14] B. Peng, N. Fisher, and T. Chantem, "MILP-based deadline assignment for end-to-end flows in distributed real-time systems," in *Proc. 24th Int. Conf. Real-Time Netw. Syst. (RTNS)*, Oct. 2016, pp. 13–22.

[15] H. Al-Omari, F. Wolff, C. Papachristou, and D. McIntyre, "An improved algorithm to smooth delay jitter in cyber-physical systems," in *Proc. 8th IEEE Int. Conf. Embedded Comput.*, Sep. 2009, pp. 81–86.

[16] B. Oklander and M. Sidi, "Jitter buffer analysis," in *Proc. 17th IEEE Int. Conf. Comput. Commun. Netw.*, Aug. 2008, pp. 1–6.

[17] X.-L. Zhang and P. Liu, "A new delay jitter smoothing algorithm based on Pareto distribution in cyber-physical systems," *Wireless Netw.*, vol. 21, no. 6, pp. 1913–1923, Aug. 2015.

[18] K. Schmidt and E. G. Schmidt, "Optimal message scheduling for the static segment of FlexRay," in *Proc. IEEE Veh. Technol. Conf. (VTC Fall)*, Sep. 2010, pp. 1–5.

[19] M. Lukasiewycz, M. Glaß, J. Teich, and P. Milbredt, "FlexRay schedule optimization of the static segment," in *Proc. Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, Oct. 2009, pp. 363–372.

[20] A. Anta and P. Tabuada, "On the benefits of relaxing the periodicity assumption for networked control systems over CAN," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2009, pp. 3–12.

[21] G. Cena, I. C. Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "On the accuracy of the distributed clock mechanism in EtherCAT," in *Proc. IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, May 2010, pp. 43–52.

[22] K. Schmidt and E. G. Schmidt, "Message scheduling for the FlexRay protocol: The static segment," *IEEE Trans. Veh. Technol.*, vol. 58, no. 5, pp. 2170–2179, Jun. 2009.

[23] J. Zhou, W. Zhang, K. Qiu, and X. Zhu, "UM-BUS: An online fault-tolerant bus for embedded systems," in *Proc. 17th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2016, pp. 198–204.

[24] D. Goswami *et al.*, "Challenges in automotive cyber-physical systems design," in *Proc. Int. Conf. Embedded Comput. Syst. (SAMOS)*, Jul. 2012, pp. 346–354.

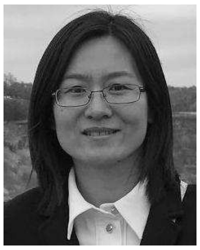
[25] R. Schneider, U. Bordoloi, D. Goswami, and S. Chakraborty, "Optimized schedule synthesis under real-time constraints for the dynamic segment of FlexRay," in *Proc. IEEE/IFIP 8th Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Dec. 2010, pp. 31–38.

[26] C. M. Krishna, "Fault-tolerant scheduling in homogeneous real-time systems," *ACM Comput. Surv.*, vol. 46, no. 4, Apr. 2014, Art. no. 48.

[27] K. W. Tindell, H. Hansson, and A. J. Wellings, "Analysing real-time communications: Controller area network (CAN)," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 1994, pp. 259–263.

[28] T. K. Liu, Y. P. Chen, and J. H. Chou, "Solving distributed and flexible job-shop scheduling problems for a real-world fastener manufacturer," *IEEE Access*, vol. 2, pp. 1598–1606, 2014.

[29] K. Kulkarni and J. Venkateswaran, "Hybrid approach using simulation-based optimisation for job shop scheduling problems," *J. Simul.*, vol. 9, no. 4, pp. 312–324, Nov. 2015.



JIQIN ZHOU received the B.S. degree in communication engineering from the Tongji University, Shanghai, China, in 2000, and the M.S. degree in mechatronic engineering from the Beijing Institute of Fashion Technology, Beijing, China, in 2008.

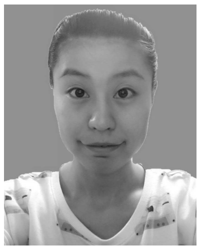
She is currently pursuing the Ph.D. degree with the School of Mathematical Sciences, Capital Normal University, Beijing, China. Her research interests include high-reliable embedded system architecture, high-speed networks, and networked control systems.

WEIGONG ZHANG received the B.S. degree in computer science and technology from the Dalian University of Technology, Dalian, China, in 1989, and the Ph.D. degree in microelectronics and solid state electronics from Xidian University, Xian, China, in 2005.

He is currently a Professor and the Doctoral Supervisor with the College of Information Engineering, Capital Normal University, Beijing, China. His research interests include reliable embedded system and reliable high-speed device interconnection.

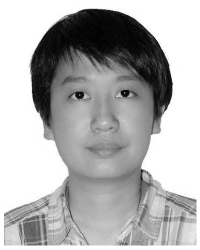
KENI QIU received B.S. degree from the Department of Information Technology, Central China Normal University, Wuhan, China, in 2001, the M.S. degree from the University of Chinese Academy of Sciences, Beijing, China, in 2004, and the Ph.D. degree from the City University of Hong Kong, China, in 2014, respectively.

She was with an Academy of Sciences, Beijing, China, from 2004 to 2010. She is currently an Associate Research Professor with the College of Information Engineering, Capital Normal University, Beijing, China. Her research interests include HW/SW codesign for embedded systems and non-volatile memories.



RUIYING BAI received the B.S. degree in computer science and technology from the Shanxi Normal University, Linfen, China, in 2014.

She is currently pursuing the master's degree with the College of Information Engineering, Capital Normal University, Beijing, China. Her research interests include reliable embedded system architecture, high-speed networks.



YING WANG received the B.S. degree in intelligent information engineering from the Capital Normal University, Beijing, China, in 2007.

She is currently a postgraduate with the College of Information Engineering, Capital Normal University, Beijing, China. Her research interests include reliable embedded system architecture, high-speed networks.

XIAOYAN ZHU received the B.S. degree in computer science and technology from the Dalian University of Technology, Dalian, China, in 1989.

She is currently a Senior Engineer with the College of Information Engineering, Capital Normal University, Beijing, China. Her research interests include reliable embedded system, reliable high-speed device interconnection.