IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 25, 2022, accepted April 25, 2022, date of publication May 12, 2022, date of current version May 18, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3174713

A Frequency-Time Synchronization Scheme for Real-Time Wireless Sensor Networks

GUANGMING QIAN AND RI KANG
College of Information Science and Engineering, Hunan Normal University, Changsha 410081, China
Corresponding author: Guangming Qian (qqyy@hunnu.edu.cn)

ABSTRACT In a real-time wireless sensor network (RT-WSN), an unpredictable time length of the synchronization (or connection) process between nodes is generally a pity, though the communication after the connection may be controllable. The purpose of this paper is to solve this kind of pity based on the *multiple-request-single* occasion (multiple slave nodes request to send data to a single master node simultaneously before getting synchronized using the frequency channel hopping technique). Suppose that the master sends the *synchronization packet* (or *beacon*) and the slaves scan for this packet with different channels for connection. A slave *getting synchronized* with the master means that both nodes have just selected an identical *frequency* channel during a *time* region and the slave has received the synchronization packet successfully in this region, which is called *frequency and time synchronization*, abbreviated as *FTS*. For many existing wireless protocols, if they are directly adopted in this situation, two deficiencies exist as for real-time performances: First, the time length required for a slave to join the network is often not deterministic if one or more channels are disturbed. Second, when multiple slaves do their scanning simultaneously, which slave can synchronize with the master first is unpredictable so that a slave with a lower priority may be serviced prior to others. In this paper, two FTS examples with poor real-time performances are provided first. Then, a synchronization method named 1/2n FTS is presented and proved. With this method, a slave scans for the synchronization packet of the master with n different available channels repeatedly until it gets the packet while the master transmits the packet 2n times in 2n continuous timeslots. The width of the scan widow of the salve takes twice the width of the slot. In this way, every slave has the opportunity to get synchronized with the master at the end of the 2n slots even if one or more (not all) channels are disturbed. Then, the slaves can send their requests to the master in different slots so that the master can schedule subsequent communications according to their requests and priorities. Also, if the mater broadcasts the beacon periodically, the time length for a slave to join or rejoin the master is not difficult to predict. The theorems associated with the 1/2n FTS method are demonstrated in experiments with NORDIC Semiconductor chips.

INDEX TERMS Frequency and time synchronization, channel hopping, predictability, 1/2n FTS method.

I. INTRODUCTION

The real-time communication in wireless sensor networks (WSNs) has received extensive study. Numerous relevant results have been published on typical techniques, especially with the medium access control (MAC) layer, such as polling-based [1], TDMA [2], and token-based [3] schemes. Frequency hopping, or channel hopping, should also be regarded as another important real-time wireless technique. The first reason is that it is widely used in representative standards and protocols, such as WirelessHART [4] and IEEE 802.15.4-TSCH [5]. Another reason is that frequency conflicts in shared wireless medium may occur at any time thus finding an undisturbed frequency channel is an essential operation in real-time applications.

Being contention free and capable of avoiding various interferences are two basic targets in the design of real-time wireless sensor networks (RT-WSNs). Combining TDMA with frequency hopping is a natural choice. WirelessHART, for example, selects its operating frequency from 16 channels in the 2.4GHz ISM (Industrial Scientific Medical) band every timeslot, the width of which is equal to 10 ms [6]. However, scheduling real-time flows to satisfy their deadlines is a NP hard problem in multi-hop networks. Notice that timeslots, multiple frequencies, multiple hops and multiple nodes. These multiple factors make the analysis more complicated.

The associate editor coordinating the review of this manuscript and approving it for publication was Biju Issac.

This paper focuses on the synchronization problem with multiple frequencies and nodes.

Consider two wireless nodes, a sender and a receiver, to communicate with each other. Suppose that there are multiple frequency channels to be selected. Before any contact, a node does not know which channel is being used by its counterpart. In order to connect with each other, first they must try with some frequencies to reach frequency synchronization and get the knowledge of how to change the frequency in subsequent communications. Then, *getting synchronized* means that both nodes have just selected an identical *frequency* channel during a *time* region and the receiver has received the information for synchronization from the sender successfully in this region. This type of synchronization is called *frequency and time synchronization* in this paper, abbreviated as *FTS*.

If only one frequency channel is available for the communication of the nodes, then the FTS becomes a simple problem of *time synchronization* (*TS*) that has been discussed intensively. In [7], for example, three algorithms are proposed to make the clocks of the nodes converge as quickly as possible and keep them most similar as possible.

To the best of our knowledge, *the establishment of FTS* has not been deeply and extensively exploited in the literature. To evaluate the real-time performances with FTS, both *frequency* and *time* should be considered, even in simple RT-WSNs.

In this paper, the problem of FTS in the single-hop RT-WSN of star topology is discussed. A *central node*, or a *master*, inquires all the *sensor nodes*, or *slaves*, and then select one or more slaves to communicate with. To avoid collision, no packet is allowed to deliver from any slave without inquiring from the master.

An *improper* method to fulfill this FTS problem can lead to two major shortcomings that affect the real-time performances of the RT-WSN:

- *Shortcoming* 1. The time for a slave node to get synchronized with the master is not deterministic. Some slaves may take too much time. Obviously, this feature is not good for real-time applications.
- *Shortcoming* 2. With the occasion of *multiple-request-single*, that is, when multiple slaves have communication requests with the master at the same time, a slave with a lower priority may get synchronization and be serviced first than other slaves with higher priorities.

A. PAPER CONTRIBUTIONS

Detailed analysis of the FTS problem and examples are provided. A synchronization method named 1/2n FTS is presented and proved. The analysis is based on the single-hop RT-WSN described above. With this method, every slave node has the chance to get synchronized with the master at a timeslot predefined by the master, hence the above two shortcomings will be overcome if the synchronization packet can reach the slaves correctly. This 1/2n FTS method is verified and demonstrated using NORDIC chips nRF24L01 [8] and nRF52840 [9].

B. PAPER STRUCTURE

Section II shows two FTS examples with poor real-time performances. Associated definitions with the FTS problem, Theorem 2 and Theorem 3 are provided in Section III. The synchronization method 1/2n FTS based on these theorems are presented and discussed in Section IV. In Section V, the detailed processes of the experiments are described and the typical results are listed. SectionVI discusses the related work and Section VII concludes the paper.

II. TWO EXAMPLES

In this section, two FTS examples with poor real-time features are provided and analyzed.

A. AN EXAMPLE OF UNPREDICTABLE FTS

Frequency channel hopping is used in the well-known Bluetooth standards [10]. Suppose that multiple scanning nodes (*scanners*) are scanning for the advertising packet from an advertising node (*an advertiser*) and desire to connect with it at the same time, that is, a *multiple-request-single* communication topology. Three primary frequency channels, 37 (f_1 = 2404MHz), 38 (f_2 = 2426MHz), and 39 (f_3 = 2480MHz), are used for the FTS between the scanners and the advertiser. However, there is no strict timing or advertising channel index selection rule as described in the specification of the standard [10]. Compared with our RT-WSN topology, here the scanner and the advertiser are equivalent to the salve and the master respectively. The advertising packet can be regarded as an inquiring one from the master.

FIGURE 1 shows a possible example of advertising and scanning. The advertiser starts *an advertising event* from *the time point* t_{M_0}. Scanner 1 and scanner 2 begin to scan earlier or not later than t_{M_0}. Suppose that the width of *a scan window* (the time interval in which the transceiver of a scanner is in its receiving mode with a specific frequency) is constant and about equal to the time length of the advertising event. Also, assume that scanner 1 starts *an* f_3 *scan window* (in the receiving mode with *scanning frequency* f_3) at t_{M_0}

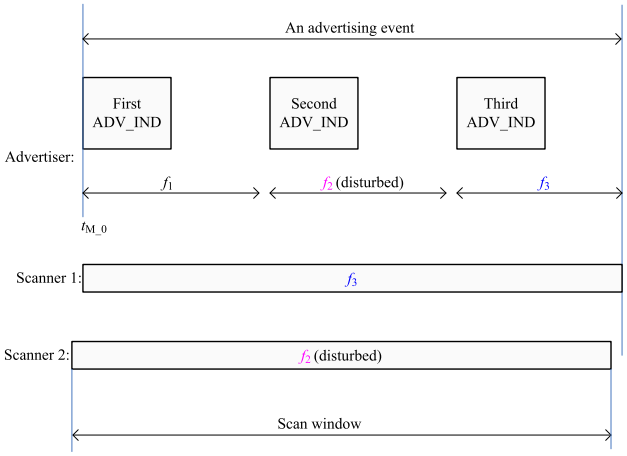

FIGURE 1. An example of advertising and scanning.

and scanner 2 is in an f_2 scan window for most of the event time, as shown in the figure.

The master first sends a PDU (protocol data unit) with a *transmitting frequency* (or *advertising frequency*) f_1. The type of the PDU is ADV_IND, which indicates a connectable and scannable undirected advertising event [10]. Because the scanning frequencies of scanner 1 and scanner 2 are different from the advertising frequency f_1, neither of them gets the first PDU.

The second PDU containing the second ADV_IND with f_2 can not be received yet by scanner 1 since it is still in the f_3 scanning window. Assuming f_2 channel is disturbed by a strong interference source, then scanner 2 can not get this PDU either. Next, scanner 1 has the opportunity to obtain the third PDU because this PDU is transmitted with an undisturbed f_3, which is the same as the scanning frequency of scanner 1. However, scanner 2 does not have any chance to get the third PDU due to a different scanning frequency.

In FIGURE 1, therefore, scanner 1 has the opportunity to get the advertising event and hence connect with the master while scanner 2 does not have any chance. If this example occurs in a real-time application where the priority of scanner 2 is higher than that of scanner 1, then *the node with a lower priority will be serviced first*, which is an undesirable result.

Actually, the above advertising and scanning can be considered as a kind of synchronizing process. Generally, because the relative positions between the advertising event and scan windows are not fixed, which scanner gets synchronized with the master first is random. *This does not produce a predictable FTS*.

B. AN EXAMPLE OF LENGTHY FTS

FIGURE 2 shows another FTS example. Consider a simple topology that two salves request to communicate with one master. From t_{M_0}, the master starts the process in the sequence of round 1 with f_1, round 2 with f_2, and then round 3 with f_3. Every round consists of three slots with equal length (T_{slot}), indexed by 0, 1, and 2. In each round, the master sends a synchronizing packet in slot 0, and then in slot 1 and slot 2, waits for the requests from slave 1 and slave 2 respectively, as shown in FIGURE 2. The process stops as soon as a request is received. If no request is received until the end of round 3, then the master will enter round 1 again.

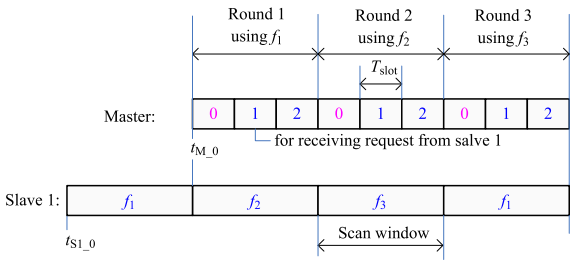

FIGURE 2. An example of unsuccessful synchronization.

In this figure, the scanning frequency of slave 1 is f_1, then f_2, and then f_3, one after the other. It begins scanning for the synchronizing packet from the time point t_{S1_0} with f_1, which is earlier than t_{M_0}. The difference between these two points is equal to the width of the scan window. If salve 1 receives the synchronizing packet, it will send a request. Unfortunately, this will never happen because its scanning frequency in any round will never be the same as the transmitting frequency of the master. In round 1, for example, the master transmits the packet with f_1 while the salve receives it with f_2, an unreasonable cooperation.

Therefore, the synchronization in FIGURE 2 is *an infinite process*, which should be deleted in RT-WSNs.

III. FREQUENCY ALIGNMENT

Consider a single-hop RT-WSN composed of one master and m (a positive integer) slaves, $m \geq 1$. In order to fulfill an FTS, first we should get a *frequency alignment*, or precisely, an *undistorted frequency alignment* that is defined in this section.

***Definition 1 (f_x Channel**: A communication channel with a central carrier frequency f_x is called an **f_x frequency channel**, or **f_x channel**. In other words, the channel is indexed by f_x.*

***Definition 2 (F List):** Suppose that there are n (a positive integer) available channels indexed from f_1 to f_n. These channels are listed by the increasing order of the $f's$ subscript, that is, $f_1, f_2, \ldots,$ and f_n. Then this list is called the **available channel list**, or **F list**.*

***Definition 3 (f_x Scan Window):** A continuous time region during which a slave is ready to receive the synchronization packet with channel f_x is defined as **an f_x scan window**, or **f_x window**, denoted as **w_f_x**.*

***Definition 4 (f_x Alignment):** Suppose that channel f_x is used to transmit one synchronization packet from t_{TX_start} to t_{TX_end} by the master. If t_{TX_start} is not earlier than the start of a scan window w_f_x of a slave and t_{TX_end} is not later than the end of this window, then we declare that the transmission of this packet **is included by** the scan window, or there is an **f_x alignment**. If f_x channel is interfered that this packet can not be received successfully, then this alignment is called a **distorted alignment**; otherwise it is an **undistorted alignment**.*

The above definitions will be explained below.

For the convenience of description, several assumptions are listed as follows:

- A scanning salve can get a correct synchronization packet transmitted from the master by *every undistorted alignment* in the discussed single-hop RT-WSN.
- Only one channel is used in each timeslot of any node. Also, a slave utilizes only one channel in each scan window.
- The conversion time from one channel to another in any node is negligible.

- Suppose that a packet transmitted from node A has been received by node B. Let the ending time point of the transmission at A be t_{TX_end} and that of the reception at B be t_{RX_end}. We assume $t_{TX\_end} = t_{RX\_end}$.

Theorem 1: *Suppose that* ***slave*** *k, $k \in (1, 2, \ldots, m)$, is designed to scan for packets periodically, starting from a time t_{Sk_0}. Each period contains n consecutive scan windows of equal width (T_{window}), starting from w_f_1 till w_f_n in the order of the F list. Assume that, from a time t_{M_0} not earlier than t_{Sk_0}, the* ***master*** *starts to transmit 2n synchronization packets in 2n consecutive timeslots of equal length (T_{slot}) using channel f_i, $i \in (1, 2, \ldots, n)$, one packet corresponding to one timeslot. If $T_{window} = 2T_{slot}$, then there must be at least one f_i alignment. The action of the 2n consecutive transmissions by the master is called a CT_2n action, the process of which is called a CT_2n process.*

Proof: See Appendix B.

FIGURE 3 is prepared to explain the above definitions and Theorem 1. In this figure, the master is the transmitter while slave 1 is a receiver. Suppose that there are three channels in the F list: f_1, f_2, and f_3. In other words, we have $n = 3$ and $2n = 6$. Six timeslots of equal length (T_{slot}), indexed by 1 to 6, are used to send six synchronization packets with f_1 by the master, that is, a CT_6 action. Each slot is allocated for one packet.

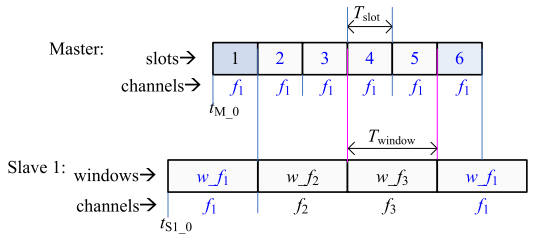

FIGURE 3. An example of two f_1 alignments.

In FIGURE 3, slave 1 scans for the synchronization packet with scan windows of equal width (T_{window}) periodically. Consecutive windows, w_f_1, w_f_2, and w_f_3 are used in each period. Note that $T_{window} = 2T_{slot}$ and $t_{M\_0} > t_{S1\_0}$, as shown in the figure. Theorem 1 declares that there will be at least one f_1 alignment. The figure shows two f_1 alignments: slot 1 is included by the first w_f_1 and slot 6 by the w_f_1 of the next period.

In Theorem 1, $t_{M\_0} \geq t_{Sk\_0}$ is assumed. If $t_{M\_0} < t_{Sk\_0}$, that is, the CT_2n action begins earlier than the start of the scanning, then we have Theorem 2.

Theorem 2*: In Theorem 1, there must be one f_i alignment in the CT_2n process starting from t_{M_0} if $t_{M\_0} < t_{Sk\_0} \leq t_{M\_0} + T_{slot}$.*

Proof: See Appendix C.

Combining Theorem 1 and Theorem 2, it is concluded that there must be at least one f_i alignment if $t_{Sk\_0} \leq t_{M\_0} + T_{slot}$. In other words, if $t_{Sk\_0} > t_{M\_0} + T_{slot}$, there may not be an alignment in the current CT_2n process. Notice the time point $t_{M\_0} + T_{slot}$. In the associated experiments, it will be used to evaluate the longest time between t_{Sk_0} and the end of synchronization in salve k.

IV. A NEW SYNCHRONIZING METHOD

A. THE FTS METHOD

Now we give a new synchronizing method named 1/2n FTS based on the above theorems. Its pseudo code is given in Algorithm 1. First we explain its basic principle with the help of FIGURE 4, and then discuss the pseudo code.

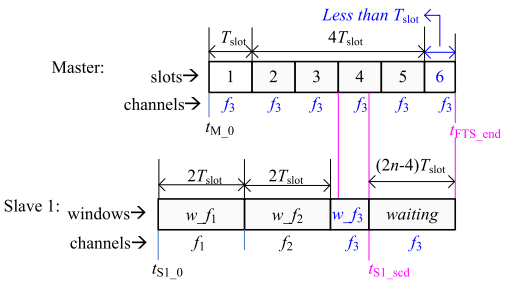

FIGURE 4. Slave 1 gets synchronized with the master at t_{FTS_end}.

We assume that the F list has three available channels in FIGURE 4, that is, $n = 3$ and $2n = 6$. The master is designed to send six *synchronizing packets (SYNPs)* with six slots using a proper channel f_3. The way to select such a proper channel will be discussed later. According to Theorem 1, there must be at least one f_3 alignment. In this example, the slot 4 of the master is included. In this figure, the first five slots have the same length of T_{slot} while the length of slot 6 is less than T_{slot}. In the design, T_{slot} is greater than the transmission time of a SYNP so that slot 6 ends at t_{FTS_end}, which is the completion of sending the sixth SYNP. Because slot 4 is included by the scan window w_f_3, the fourth SYNP is received by slave 1. This is an f_3 alignment. As soon as this SYNP is received at the time point t_{S1_scd}, salve 1 finishes window w_f_3 and takes the index 4 from this packet. Then, slave 1 keeps its frequency channel unchanged and waits to t_{FTS_end}. The width between t_{S1_scd} and t_{FTS_end} is calculated by

$$(2n - \textit{the packet index})T_{slot} = (6 - 4)T_{slot} = 2T_{slot}. \quad (1)$$

Note that t_{S1_scd} implies the end of the scanning process of slave 1 and t_{FTS_end} indicates that of the synchronizing process. The width of w_f_3 would be $2T_{slot}$ without this f_3 alignment.

In Algorithm 1, the pseudo code of 1/2n FTS is divided into two parts: *the master part* and *the slave part*. In the master part, a timer is prepared and started at line 1 and line 2. It causes an interrupt every T_{slot}. In other words, every slot except the last one ends as long as a timing interrupt occurs. The action CT_2n is implemented from line 3 to 9. A SYNP indexed by a unique number is transmitted in line 4. If it is not the last slot (not indexed by 2n), the length of the slot equals T_{slot}, which is controlled by the code from line 5 to 8; otherwise the CT_2n is completed as soon as the last SYNP is sent.

Algorithm 1 1/2N FTS Synchronization Method

The master part:

Input: $n, f_i \in F$, and T_{slot}.

1: set a timer that causes an interrupt every T_{slot}
2: start the timer and clear *its interrupt flag*
3: **for** ($ind_{\text{slot}} = 1$; $ind_{\text{slot}} < 2n + 1$; $ind_{\text{slot}} + +$) **do**
4: send a SYNP indexed by ind_{slot} with channel f_i
5: **if** ($ind_{\text{slot}} < 2n$) **then**
6: wait until *the interrupt flag* $= 1$
7: clear *the interrupt flag*
8: **end if**
9: **end for**
10: the end of synchronizing process

The slave part:

Input: n, F, and T_{slot}.
1: clear *winflag*, *recflag*, and *pacflag*
2: $s = 1$ and set the scanning channel $= f_s$
3: set a timer that causes an interrupt every T_{slot}
4: start the timer and clear its interrupt flag
5: start listening for a SYNP
6: **while** (*pacflag*$= 0$)**do** //for packet
7: **if** (*winflag* $= 1$)**then** //for window
8: *winflag* $= 0$
9: s++ if $s \neq n$, otherwise $s = 1$
10: set the scanning channel $= f_s$
11: **end if**
12: **if**(*recflag* $= 1$)**then** //for received event
13: clear *recflag* and read the packet
14: **if**(the SYNP is invalid)**then**
15: continue to listen for a packet
16: **else**
17: *pacflag* $= 1$
18: wait for a time equal to $(2n - ind_{\text{slot}})T_{\text{slot}}$
19: **end if**
20: **end if**
21:**end while**
22:the end of synchronizing process

In the slave part of this method, a timer that causes an interrupt every T_{slot} is also needed (line 3 and 4). However, a flag named *winflag* must be prepared to control the width of the window. This flag becomes 1 every 2 T_{slot} in the interrupt program of the timer, which is not shown in the pseudo code. The slave starts to scan with channel f_1 (line 2). At the end of the window, the scanning channel is changed, which is implemented from line 7 to 11. Lines 12 to 20 are related to the receiving process. If the flag *recflag* $= 1$, it indicates that a packet is received. If the packet is not a valid SYNP, the slave will continue the scanning process; otherwise, another flag *pacflag* will be set, then the slave will stop scanning and wait for $(2n - ind_{\text{slot}})T_{\text{slot}}$ to the end of the synchronizing process.

The temporal complexity of the method: To fulfill this method, the master needs only to prepare and then transmit a SYNP in every slot. One CT_2n action is completed in less than $2n$ slots. In total, $2n$ times of preparation and transmission of a SYNP are required, which can be considered as the temporal complexity of this method in the master. Comparatively, the slave has three types of operation: hopping channel at the beginning of each window, reading and checking SYNP on f_i alignment, and waiting to $t_{\text{FTS\_end}}$ if a valid SYNP is received. Suppose $t_{\text{M\_0}}$ is not earlier than the start of the scanning. Then in each CT_2n, the temporal complexity in the slave can be described as follows: channel hopping is required at most n times; packet reading and checking is needed at most 2 times.

Two advantages of the method: The first advantage is that every slave that starts its scanning not later than $t_{\text{M\_0}}$ has the opportunity to get a SYNP and thus get synchronized with the master. If slaves send their responses to the master in allocated slots after $t_{\text{FTS\_end}}$, then *a slave of a higher priority will not be preempted by the one of a lower priority*. The second advantage is about *time predictability*. The time length of a CT_2n action is fixed. Slaves get synchronized with the master at an identical time $t_{\text{FTS\_end}}$. Thus, it is not difficult to predict the time length from $t_{\text{M\_0}}$ to a certain time point after $t_{\text{FTS\_end}}$. Both advantages are important for many real-time applications.

B. SELECTING A PROPER CHANNEL

In the 1/2n FTS method, a proper frequency channel should be selected before a CT_2n action is taken by the master. We propose three ideas for this purpose:

- *Idea* 1: *round robin selection*. If f_i of the F list is used in a CT_2n action, then f_{i+1}, or f_1 if $i = n$, will be used in the next CT_2n action. This is called *round robin selection.*
- *Idea* 2: *selection with an AUX node*. An auxiliary node (*AUX node*) designed for this purpose is placed near the master. Before a CT_2n action, the master can start a test process by sending a test packet to this AUX node with a frequency channel and wait for its response. If the response is received correctly, then this channel will be used in this CT_2n action for the slaves; otherwise the test process is repeated with a different channel. This is called *selection with an AUX node*.
- *Idea* 3: *selection on historic information*. This idea means that the selection of a proper channel used by the master to start a CT_2n action depends on previous information of communication, such as the percentage of correct packets received in a previous interval of time.

V. EXPERIMENTS

In the experiments, two types of wireless nodes are utilized. The node of the first type uses the nRF24L01 chip as the transceiver and the STM32F103RCT6 as the processor [11]. We call it an *nRF24L01 node* shortly. A node of the second type is called an *nRF52840 node* in which an nRF52840 chip functions as the transceiver as well as the processor. We give the experimental results with the nRF52840 node as follows.

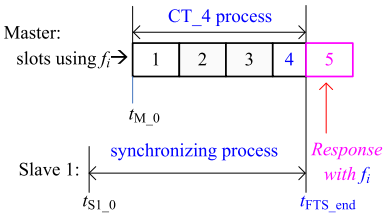

FIGURE 5. A response can be received within five slots if $2n = 4$.

A. VERIFICATION OF THEOREM 1

We have done the experiments for verifying theorem 1 with the cases of $n = 1$, $n = 2$, $n = 3$, $n = 4$, and $n = 5$. Remember that n is the total number of the available channels.

In the experiments, two nodes are required: one is the master and the other is slave 1. We set several important parameters as follows:

$$\begin{aligned}\text{The air data rate} &= 2\text{Mbps}.\\ T_{\text{slot}}(\text{the length of the slot}) &= 800\mu\text{s}.\\ \text{The payload length of a packet} &= 32\text{bytes}.\end{aligned}$$

An illustrative example with $n = 2$ is given in FIGURE 5. Slave 1 starts its synchronizing process, or scanning process, from t_{S1_0} and sends a response packet to the master immediately after t_{FTS_end}. In order to ensure no packet loss, that is, one of the SYNP can be received by slave 1 and the master can get a correct response from the slave, we place these two nodes within a distance of 60 cm.

In the program of the master, *a counter* is designed to count timeslots from t_{M_0}. If $n = 2$, then Theorem 1 is correct as long as the response packet of slave 1 is received when the value of the counter is equal to five. Generally, the response should be received within $2n+1$ slots from t_{M_0}. We introduce *calcu_slot_num* to represent $2n+1$ and another symbol *real_slot_num* to indicate the real number of the slots required to get the response in the tests. We need to verify that

$$real\_slot\_num = calcu\_slot\_num = 2n + 1. \tag{2}$$

This verification should be done with different values of t_{M_0}. Let Δ_1 denote t_{M_0} minus t_{S1_0}, $\Delta_1 \geq 0$. We do multiple times of the verification, each time with a different value of Δ_1. In the program of the master, we take a different value of t_{M_0} every 2μs. Because the scanning period is $4 \times 800 = 3200\mu$s if $n = 2$, the total number of the Δ_1 values equals 1600, which is the times of the experiments to be done in one period, denoted as *veri_exp_times*. Generally, we have

$$veri\_exp\_times = 2nT_{\text{slot}}/2 = nT_{\text{slot}}. \tag{3}$$

The experiment results: Totally, the results of 12000 experiments are listed in TABLE 1. In every experiment, we get *real_slot_num* = *calcu_slot_num*, which shows Theorem 1 is correct. In each of the 800 experiments with $n = 1$, for example, we have *real_slot_num* = *calcu_slot_num* = 3.

If the distance between the two experimental nodes is enlarged, then packet loss may occur and the *real_slot_num* may be greater than the *calcu_slot_num*.

TABLE 1. Results of verification experiments.

n	*veri_exp_times*	*calcu_slot_num*	*real_slot_num*
1	800	3	3
2	1600	5	5
3	2400	7	7
4	3200	9	9
5	4000	11	11

B. TIME PREDICTABILITY

To show the time predictability of the RT-WSN based on the 1/2n FTS method, a network prototype is constructed and described with the help of FIGURE 6. In this figure, there are two channels (f_1 and f_2) in the F list. The master collects data from three slaves periodically. Each *round*, or *period*, consists of three stages: *stage* 1(CT_4 process), *stage* 2(*data stage*), and *stage* 3. In stage 2, salves send data to the master in their allocated slots. Stage 3 is a *post-processing stage* reserved for the ending operation of the current round and the preparation for the next round. Suppose that channel f_2 is selected and used by the master in round 1. Each slave starts scanning with a window w_f_1. According to Theorem 1 and 2, slave 1 and 2 can get synchronized with the master in round 1 because they start scanning not later than the end of slot 1 of the master. This is not true for slave 3 because it has no w_f_2 window from t_{S3_0} to t_{FTS_end}.

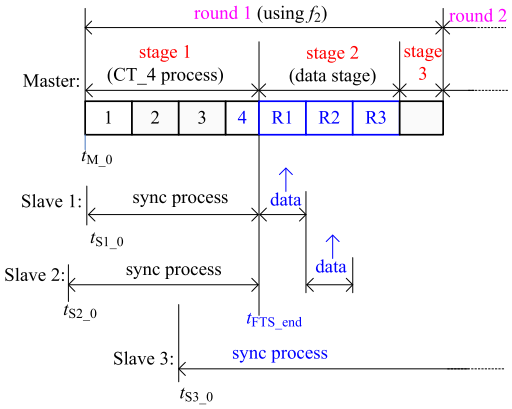

FIGURE 6. The master periodically acquires data from three slaves. Slave 1 and 2 can get synchronized with the master in round 1 if channel f_2 is used by the master in this round.

As a result, slave 1 and slave 2 will send data to the master in the slot R1 and slot R2 of the data stage in round 1, respectively. However, slave 3 will continue scanning to round 2 in which it can get synchronized with the master and transmit data in slot R3 of round 2. In this figure, *sync* stands for synchronizing.

In our experiments, $L_{\text{stage}(3)}$ is used to indicate the time length of stage 3 and is less than T_{slot}. Also, we use $L_{\text{stage}(1)}$ and $L_{\text{stage}(2)}$ to denote those of stage 1 and stage 2, respectively.

Let us record *the time length* from the start of the scan to the end of the synchronization process, that is, $t_{FTS\_end} - t_{Sk\_0}$

for slave k. We introduce a symbol $L_{\text{sync(k)}}$ to represent it:

$$L_{\text{sync(k)}} = t_{\text{FTS\_end}} - t_{\text{Sk\_0}}. \tag{4}$$

Furthermore, we use $L_{\text{sync(k)|min}}$ and $L_{\text{sync(k)|max}}$ to denote the minimum and the maximum values of $L_{\text{sync(k)}}$.

Apparently, $L_{\text{sync(k)}}$ is *a very important parameter* in the evaluation of the time predictability for a real-time network.

In FIGURE 6, we have $L_{\text{sync(1)}} < 4T_{\text{slot}}$ for slave 1. For slave 3, however, it has to go through slot 3, slot 4, three slots of stage 2, stage 3, and then through the CT_4 process of round 2, thus we get $L_{\text{sync(3)}} < 10T_{\text{slot}}$. Remember that the width of slot 4 in stage 1 is less than T_{slot}.

Generally, if the master is not ready, then no communication is possible. Therefore, we focus our analysis on *the master-first mode: the master goes into round 1 before any slave starts to scan. In the following discussions, we assume that the master-first mode is adopted* unless otherwise specified.

If data is ready for transmission in a slave, it starts to scan and tries to get synchronized with the master. In this case, it is reasonable to hope that its prepared data should be sent to the master as quickly as possible in real-time applications. Then, $L_{\text{sync(k)|max}}$ plays a crucial role. It is possible that $t_{\text{Sk\_0}}$ equals the start of the last slot in the current CT_2n process and the slot 2n is included, thus we have

$$L_{\text{sync(k)|min}} < T_{\text{slot}}. \tag{5}$$

In the proof of Theorem 2, we will show that there may not be any alignment in the CT_2n process if $t_{\text{Sk\_0}} > t_{\text{M\_0}} + T_{\text{slot}}$. Thus $L_{\text{sync(k)|max}}$ should be discussed from $t_{\text{M\_0}} + T_{\text{slot}}$. For example, in FIGURE 6, it is not difficult to get $L_{\text{sync(k)|max}} < 11T_{\text{slot}}$ since the length from $t_{\text{M\_0}} + T_{\text{slot}}$ of round 1 to the end of the CT_4 process of round 2 is less than $11T_{\text{slot}}$.

With m slaves and n available channels, we have

$$\begin{aligned} L_{\text{stage(1)}} &< 2nT_{\text{slot}}, \\ L_{\text{stage(2)}} &= mT_{\text{slot}}, \text{ and} \\ L_{\text{stage(3)}} &< T_{\text{slot}}. \end{aligned}$$

Therefore, $L_{\text{sync(k)|max}}$ can be calculated by

$$\begin{aligned} &L_{\text{sync(k)|max}} \\ &\quad = L_{\text{stage(1)}} - T_{\text{slot}} + L_{\text{stage(2)}} + L_{\text{stage(3)}} + L_{\text{stage(1)}} \\ &\quad < (2n - 1 + m + 1 + 2n)T_{\text{slot}} = (4n + m)T_{\text{slot}}. \end{aligned} \tag{6}$$

Because the value of $(4n + m)T_{\text{slot}}$ is *deterministic*, the 1/2n FTS method features *good time predictability*. Let us call $(4n + m)T_{\text{slot}}$ the up bound of $L_{\text{sync(k)|max}}$.

In every slave, a timer that causes an interrupt every T_{slot} is utilized. *A counter* is designed in the program and incremented in the beginning of a slot. We measure $L_{\text{sync(k)|max}}$ by the value of the counter. In this way, the up bound of $L_{\text{sync(k)|max}}$ will be $(4n + m)T_{\text{slot}}$, or $(4n + m)$slots.

In the experiments, we take $m = 3$, and then test with $n = 1$, $n = 2$, $n = 3$, $n = 4$, and $n = 5$. For each value of n, we do 50 tests and record the maximum $L_{\text{sync(k)}}$ among these tests, that is, the $L_{\text{sync(k)|max}}$ in number of slots. In every test, $t_{\text{Sk\_0}} - t_{\text{M\_0}}$ is a random value. In other words, the slave starts to scan in a randomly selected time after $t_{\text{M\_0}}$.The output power of the transceiver is set to 8dBm and the nodes are put close enough to prevent packet loss. TABLE 2 shows the results of the tests.

TABLE 2. Values of $L_{\text{sync}(k)|\text{max}}$ in number of slots.

n	up bound of $L_{\text{sync}(k)\|\text{max}}$	$L_{\text{sync}(k)\|\text{max}}$ from 50 tests		
		slave 1	slave 2	slave 3
1	7	6	6	6
2	11	10	10	9
3	15	14	14	14
4	19	18	18	18
5	23	19	22	22

The experiment results: From the table, we can see that every value of $L_{\text{sync}(k)|\text{max}}$ from the tests is not greater than the corresponding up bound. With $n = 2$, for instance, the up bound of $L_{\text{sync}(k)|\text{max}}$ is 11 slots while the maximum value from the 50 tests is equal to 10 slots. This declares that Theorem 2 is correct and the network based on the 1/2n FTS method has good time predictability.

VI. RELATED WORK

Up to today, the literature on real-time wireless networks is multifaceted. Three big concepts, IIOT (Industrial Internet of Things), industry 4.0 and CPS (Cyber-physical System), are clarified and discussed in their similarities and differences in [12]. Several IIOT frameworks are introduced in [13], some of them have real-time features. Investigations on the applications of the technologies related with industry 4.0 in manufacturing and product managing can be found in [14]–[16]. In [14], for example, a comprehensive review on robotic wireless sensor networks, industrial artificial intelligence, and deep learning-assisted smart process planning in sustainable cyber-physical manufacturing systems is provided based on the data from McKinsey, Ovum, PwC, and World Economic Forum.

A detailed overview of the control and scheduling methods in MAC layer and routing protocols for RT-WSNs can be found in [17]. In terms of standard documents, WirelessHART, ISA100, and IEEE 802.15.4-TSCH are typical representatives [4]–[6].

To meet real-time constrains and ensure reliability in RT-WSNs, two things are very important: the first thing is that the *conflict* (generally refers to the time conflict) between nodes must be dealt with properly, the second one is that every node should have a certain *anti-interference* ability.

The concept of *constructive interference* is used in the time-triggered schemes using concurrent transmissions based on Glossy and it is attractive in real-time applications [18]–[20]. Glossy delivers all packets to all nodes. A packet is transmitted multiple times in the network. If the temporal offset (the difference between the start of the reception of a packet with that of its delayed replica) among concurrent transmitters does not exceed a certain value, then

the reliability (the probability of the successful reception of a flooding packet) can be very high. The upper bound of the temporal offset differs with protocols. With IEEE 802.15.4 in [18], this bound takes a value of 0.5μs. With the study over Bluetooth 5.0 in [21], although it is more fragile than over 802.15.4, concurrent transmissions are viable if the temporal offset is not greater than 0.25μs. The Blink protocol based on concurrent transmissions provides good real-time performances in some application scenarios [19]. However, there is no discussion on the FTS problem in these reference documents.

It has been widely studied by many researchers to eliminate conflicts and make a network *collision-free* [2], [3]. For example, TDMA is collision-free because nodes transmit only within their allocated slots [2]. In token-based networks, only nodes received the token are permitted to initiate data transmissions [3].

Frequency hopping and allocation (or simply *frequency hopping* or *channel hopping*) is an important means of anti-interference [5]. Using channel hopping, the communication between nodes will not be disturbed by some *external interference* from other networks in the same band. In a RT-WSN, *mutual interferences* can be prevented if different nodes utilize different frequencies to send packets at the same time.

Subsequently, a common strategy adopted by researchers for RT-WSNs is to use both *frequency hopping and collision-free* techniques.

A real-time multi-hop wireless data acquisition protocol in [22] is based on TDMA and channel hopping. A TDMA frame in the protocol contains only one timeslot. Nodes are staggered in the network. A node acts as a parent for connecting nodes with its children during phase I and as a child to communicate with its parent in phase II. At the start of each round, it switches its radio channel based on a hopping sequence which is carefully selected.

WirelessHART is a widely adopted standard [6]. It is based on TDMA with multi-channel support, channel hopping, and redundancy in routes. A WirelessHART schedule repeats over every hyperperiod. There are also many published papers on WirelessHART [6], [23]. For example, the priority scheduling of transmissions for WirelessHART networks has been studied in [24], [25]. However, there is no in-depth discussion on the FTS problem in the above literature.

In the IEEE 802.15.4-TSCH network, a frame is constructed of timeslots of equal length [5]. A TSCH schedule is a matrix that consists of cells, each of which is associated with a timeslot and a channel offset. In the matrix, two types of cells exist: dedicated cells and shared cells. The periodic transmissions of the network advertising frames named enhanced beacons (EBs) are implemented in shared cells. The scheduling of EB transmission plays a crucial role in the synchronization and network formation [26]–[29]. In fact, it is an FTS problem.

An autonomous EB scheduling method is used to illuminate EB collisions in [26]: each node takes a unique pre-allocated cell for its advertising purpose in a multi-slotframe. In [27] a dynamic algorithm to allocate resources during network bootstrap is given to guarantee fast join to TSCH network and distribute routing information properly. In [28], it is declared that the TSCH standard defines neither the advertisement strategy nor the rate of the transmission of proper EBs, then a Fast and Active Network formation (FAN) is proposed to accelerate the (re)association process. A key operation of FAN is to allow joiners to send EB requests to trigger EBs. In [29], the influence of the scan period on the initial synchronization, as well as the associated power consumption, is exploited.

At a certain time, suppose that only one active node can send its EB and several other nodes are scanning for this EB. For the purpose of anti-interference, channel hopping can be adopted both in sending and scanning. Some channel designed for this EB may be disturbed. Even so, to ensure that these scanning nodes get the EB packet at a deterministic time point so that the node with the highest priority can take action first is a usual requirement in some real-time applications. Unfortunately, it is not discussed in [26]–[29].

The frequency domain polling MAC protocol (FDP-MAC) presented in [30] is from a different view. In a frame of this protocol, the first stage is frequency domain polling, then a dynamic TDMA and acknowledgment stages. To achieve simultaneously polling, different *subcarriers* are used to signal the *access point* (AP) by different nodes that they have pending packets. This implies that *a single* AP can receive packets with *multiple frequencies simultaneously* (MFS). Similarly, the star topology introduced in [31] is also an MFS model. Multiple SAMs (sensor/actuator module) connect with one BS (base station) that is responsible for the allocation of channels and slots used by SAMs. The BS has one central control unit and multiple transceivers. Different SAMs communicate with the BS with different frequencies so that the collisions between SAMs are avoided.

Unlike the MFS models in [30] and [31], in this paper, only one channel can be used for the receiver of the master node at a time. Generally, the less the number of the frequencies simultaneously used, the simpler the structure of the transceiver and the less the related interfering sources.

Frequency hopping is an important function of Bluetooth networks [10]. It is adopted in the process of nodes joining the network as well as during the communications after connection. The advertiser can send out advertising packets at different times with different frequency channels for the scanners to receive. However, for the real-time establishment of the connection, the coordination of time and frequency between the advertiser and scanners has not been given in the Bluetooth specifications. For various settings of relevant time values, such as the length of the interval of the adjacent advertising events and the width of the scan windows, only a certain range of choices are provided in the standards. However, our analysis and experiments show that how to choose from these options is important for the fast and deterministic FTS of RT-WSNs, which is left open in the present Bluetooth specifications.

In [32] written in Chinese, a six synchronous successive transmission (SSST) algorithm to deal with the FTS problem where the number of the available channels is fixed at three is provided, which is a preliminary research of this paper.

VII. CONCLUSION

In summary, with the FTS (frequency and time synchronization) problem in the RT-WSN of star topology, the following important points are shown in this paper:

Firstly, frequency alignment is vital for salves to get synchronized with the master. For this purpose, the scan window with a frequency channel at the slave side is wider than the slot with the same channel at the master side, that is, $T_{\text{window}} > T_{\text{slot}}$.

Secondly, with $T_{\text{window}} = 2T_{\text{slot}}$ and the number of the available frequency channels equal to n, there must be a least one frequency alignment in a CT_2n action if the slave scans for the synchronization packet periodically with all the available channels and the master continuously sends the synchronization packet $2n$ times in this action. In this way, every slave that starts scanning not later than the beginning of this action, regardless of its priority, has the opportunity to get synchronized with the master. This is one major feature of the 1/2n FTS method presented in this paper.

Thirdly, another feature of the 1/2n FTS is that the time of the synchronization process is fixed from the start of the CT_2n action to its end. Hence, it is easy to make the time length (from one point to another) predictable in a well arranged and slot-based network. Obviously, these two features are critical to many real-time applications.

Finally, in the experiments, the correctness of the 1/2n FTS with $n = 1$ to $n = 5$ have been demonstrated. Also, a network prototype with three slaves is constructed. The network works in rounds dominated by the master. Each round consists of three stages. A slave sends its data in an allocated slot of stage 2 if it gets synchronized with the master in stage 1. Both theoretically and experimentally, it has been shown that the maximum value of the time length (from the start of the scan to the end of the synchronization), that is $L_{\text{sync}(k)|\max}$, is predictable.

In future research, the relationship between the maximum value of n and the accuracy of the node clock should be considered. Also, the practical application of the 1/2n FTS method in monitoring and control is encouraged.

APPENDIX A
SYMBOLS

TABLE 3 lists the major symbols used in this paper.

APPENDIX B
PROOF OF THEOREM 1

Proof: In FIGURE 7, slave k, $k \in (1, 2, \cdots, m)$, scans for a synchronization packet periodically with the F list. The master uses channel f_i, $i \in (1, 2, \cdots, n)$, to send the packet in every slot.

TABLE 3. The major symbols.

RT-WSN	Real-time wireless sensor network
FTS	Frequency and time synchronization
m	Total number of the sensor nodes (salves) in the discussed RT-WSN
n	Total number of available frequency channels
k	An index for a salve node, $k \in (1, 2, \cdots, m)$
SYNP	Synchronizing packet
CT_2n	Consecutive transmissions of $2n$ SYNPs by the master
1/2n FTS	A synchronizing method using CT_2n
f_i	A frequency channel, $i \in (1, 2, \cdots, n)$
F	The list of n frequency channels: $(f_1, f_2, \cdots, f_n)$
T_{slot}	The length of a timeslot
$t_{\text{M\_0}}$	The time for the master to start a CT_2n action
t_{Sk_0}	The time for slave k to start scanning
$t_{\text{FTS\_end}}$	The end time of a CT_2n action
T_{window}	The time length of a scan window
w_f_i	The scan window using channel f_i
t_{Sk_scd}	The time point as soon as a SYNP is received correctly by slave k

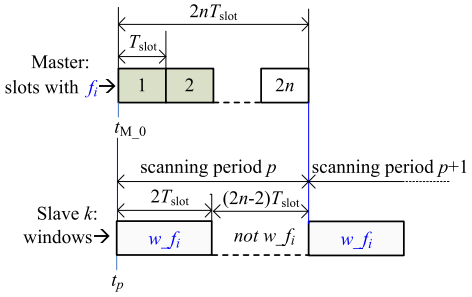

FIGURE 7. A demonstration of alignment if $n > 1$.

Let us begin a scanning period with a w_f_i window and denote the start time of period p (a positive integer) with t_p. If there is at least one f_i alignment no matter what value t_p takes in the region $[t_{\text{M\_0}}, t_{\text{M\_0}} + 2nT_{\text{slot}})$, then Theorem 1 is correct. This region can be divided into *sub region* A, *sub region* B, and *sub region* C, which are defined as follows:

$$\text{Sub region A} = [t_{\text{M\_0}}, t_{\text{M\_0}} + T_{\text{slot}}).$$
$$\text{Sub region B} = [t_{\text{M\_0}} + T_{\text{slot}}, t_{\text{M\_0}} + (2n-1)T_{\text{slot}}).$$
$$\text{Sub region C} = [t_{\text{M\_0}} + (2n-1)T_{\text{slot}}, t_{\text{M\_0}} + 2nT_{\text{slot}}).$$

If $n = 1$, then $2n - 1 = 1$ and $2n = 2$. This means that sub region B does not exist. As shown in FIGURE 8 with $n = 1$, sub region A is in slot 1 and C in slot 2 while B does not exist.

If $n > 1$, we have all the three sub regions. Therefore, it is necessary to prove the theorem by assuming t_p in sub region A, B, and C.

First of all, we discuss the problem with t_p in sub region A. If $t_p = t_{\text{M\_0}}$, we get *two* f_i *alignments* since slot 1 and slot 2 are included by the w_f_i of period p. If $t_{\text{M\_0}} < t_p < t_{\text{M\_0}} + T_{\text{slot}}$, *one* f_i *alignment* exists since the w_f_i of period p includes slot 2.

Secondly, we discuss the alignment when t_p belongs to sub region B. If t_p is equal to the start time of slot y, $y \in (2, \cdots, 2n-1)$, and $n > 1$, slot y and slot $y + 1$ must

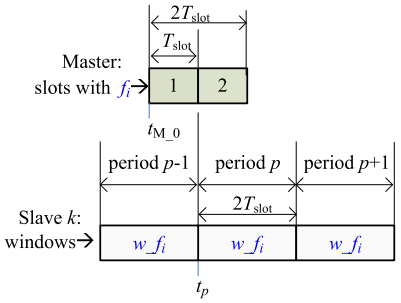

FIGURE 8. A demonstration of alignment if $n = 1$.

be enclosed by the w_f_i of period p. Thus we get *two* f_i *alignments*. If t_p does not equal the start time of any slot, then there will be *one* f_i *alignment*. For instance, slot $y + 1$ will be included if t_p takes a value in slot y.

Finally, we study the case when t_p is in sub region C. If $t_p = t_{M\_0} + (2n-1)T_{slot}$, there are *two* f_i *alignments* because slot 1 is included by the w_f_i of period $p-1$ and slot $2n$ by that of period p. If $t_{M\_0} + (2n-1)T_{slot} < t_p < t_{M\_0} + 2nT_{slot}$, there is *one* f_i *alignment* due to the included slot 1 by the w_f_i of period $p-1$.

In summary, there is at least one f_i alignment when t_p takes any value in any of the three sub regions. Therefore, Theorem 1 is correct.

APPENDIX C
PROOF OF THEOREM 2

Proof: First we prove Theorem 2 when $n = 1$ with the help of FIGURE 9. In this figure, channel f_i is used by the master in the CT_2 action and by the slave to do the scanning. It is easy to see that slot 2 in the master will be included by the window w_f_i if $t_{M\_0} < t_{Sk\_0} \le t_{M\_0} + T_{slot}$, Comparatively, there will be no alignment if $t_{Sk\_0} > t_{M\_0} + T_{slot}$. Therefore, Theorem 2 is correct with $n = 1$.

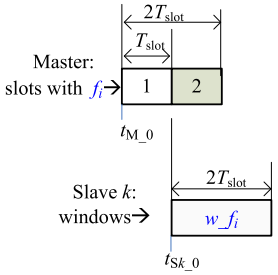

FIGURE 9. With $n = 1$, slot 2 is included if $t_{Sk\_0} = t_{M\_0} + T_{slot}$.

Then we show Theorem 2 is correct with $n > 1$ by the aid of FIGURE 10. Channel f_i is still used by the master. Note that the width of window w_f_i is $2T_{slot}$ and that of other window(s) is $(2n-2)T_{slot}$. In the first scanning period starting from t_{Sk_0}, w_f_i may be used earlier or later. According to the position of w_f_i, we have two cases:

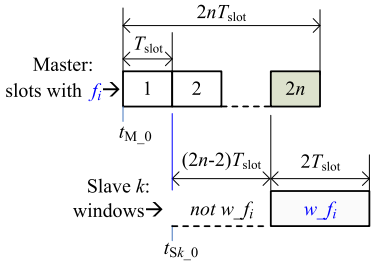

FIGURE 10. With $n > 1$, slot $2n$ is included if $t_{Sk\_0} = t_{M\_0} + T_{slot}$.

Case 1: w_f_i is the last window in the period. In this case, if $t_{M\_0} < t_{Sk\_0} \le t_{M\_0} + T_{slot}$, slot $2n$ of the master will be enclosed by the w_f_i window as shown in the figure. If $t_{Sk\_0} > t_{M\_0} + T_{slot}$, on the contrary, no slot will be included. Thus Theorem 2 is correct with this case.

Case 2: w_f_i is not the last window of the period. With the condition of $t_{M\_0} < t_{Sk\_0} \le t_{M\_0} + T_{slot}$, the end of the w_f_i must be in the range of $(t_{Sk\_0} + 2T_{slot}, t_{Sk\_0} + (2n-1)T_{slot}]$. In this range, if this window of the slave and some slot of the master end at the same time, then there will be two f_i alignments; otherwise only one alignment will occur. Anyhow, we can get at least one f_i alignment if $t_{M\_0} < t_{Sk\_0} \le t_{M\_0} + T_{slot}$. Therefore, Theorem 2 is proved with Case 2.

REFERENCES

[1] A. Willig, "Polling-based MAC protocols for improving real-time performance in a wireless PROFIBUS," *IEEE Trans. Ind. Electron.*, vol. 50, no. 4, pp. 806–817, Aug. 2003.
[2] T. W. Carley, M. A. Ba, R. Barua, and D. B. Stewart, "Contention-free periodic message scheduler medium access control in wireless sensor/actuator networks," in *Proc. Int. Symp. Syst. Chip*, Dec. 2003, pp. 298–307.
[3] M. Ergen, D. Lee, R. Sengupta, and P. Varaiya, "WTRP—Wireless token ring protocol," *IEEE Trans. Veh. Technol.*, vol. 53, no. 6, pp. 1863–1881, Nov. 2004.
[4] S. Petersen and S. Carlsen, "WirelessHART versus ISA100.11a: The format war hits the factory floor," *IEEE Ind. Electron. Mag.*, vol. 5, no. 4, pp. 23–34, Dec. 2011.
[5] D. Zorbas, G. Z. Papadopoulos, and C. Douligeris, "Local or global radio channel blacklisting for IEEE 802.15.4-TSCH networks?" in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[6] A. Samaddar, A. Easwaran, and R. Tan, "A schedule randomization policy to mitigate timing attacks in WirelessHART networks," *Real-Time Syst.*, vol. 56, no. 4, pp. 452–489, Oct. 2020.
[7] L. T. Bruscato, T. Heimfarth, and E. P. de Freitas, "Enhancing time synchronization support in wireless sensor networks," *Sensors*, vol. 17, no. 12, p. 2956, 2017.
[8] *nRF24L01+ Single Chip 2.4GHz Transceiver Product Specification 1.0*, Nordic Semiconductor, Oslo, Norway, 2008.
[9] *nRF52840 Product Specification, v1.2*, Nordic Semiconductor, Oslo, Norway, 2021.
[10] *Bluetooth Core Specification v5.0*, Bluetooth SIG, Kirkland, WA, USA, 2016.
[11] *STM32F103xC STM32F103xD STM32F103xE Datasheet Production Data*, STMicroelectronics, Geneva, Switzerland, 2018.
[12] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.
[13] C. Paniagua and J. Delsing, "Industrial frameworks for Internet of Things: A survey," *IEEE Syst. J.*, vol. 15, no. 1, pp. 1149–1159, Mar. 2021.
[14] R. Blake, L. Michalkova, and Y. Bilan, "Robotic wireless sensor networks, industrial artificial intelligence, and deep learning-assisted smart process planning in sustainable cyber-physical manufacturing systems," *J. Self-Governance Manage. Econ.*, vol. 9, no. 4, pp. 48–61, 2021.

[15] A. Galbraith and I. Podhorska, "Artificial intelligence data-driven Internet of Things systems, robotic wireless sensor networks, and sustainable organizational performance in cyber-physical smart manufacturing," *Econ., Manage., Financial Markets*, vol. 16, no. 4, pp. 56–59, 2021.
[16] A. Dawson, "Robotic wireless sensor networks, big data-driven decision-making processes, and cyber-physical system-based real-time monitoring in sustainable product lifecycle management," *Econ., Manage., Financial Markets*, vol. 16, no. 2, pp. 95–105, 2021.
[17] B.-S. Kim, H. Park, K. H. Kim, D. Godfrey, and K.-I. Kim, "A survey on real-time communications in wireless sensor networks," *Wireless Commun. Mobile Comput.*, vol. 2017, Oct. 2017, Art. no. 1864847.
[18] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Proc. 10th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, Apr. 2011, pp. 73–78.
[19] M. Zimmerling, L. Mottola, P. Kumar, F. Ferrari, and L. Thiele, "Adaptive real-time communication for wireless cyber-physical systems," *ACM Trans. Cyber-Phys. Syst.*, vol. 1, no. 2, pp. 1–29, Feb. 2017.
[20] R. Jacob, L. Zhang, M. Zimmerling, J. Beutel, S. Chakraborty, and L. Thiele, "The time-triggered wireless architecture," 2020, *arXiv:2002.07491*.
[21] B. A. Nahas, S. Duquennoy, and O. Landsiedel, "Concurrent transmissions for multi-hop Bluetooth 5," in *Proc. Int. Conf. Embedded Wireless Syst. Netw. (EWSN)*, Feb. 2019, pp. 130–141.
[22] J. Zhang, J. Wu, Z. Han, L. Liu, K. Tian, and J. Dong, "Low power, accurate time synchronization MAC protocol for real-time wireless data acquisition," *IEEE Trans. Nucl. Sci.*, vol. 60, no. 5, pp. 3683–3688, Oct. 2013.
[23] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, and Y. Chen, "Real-time wireless sensor-actuator networks for industrial cyber-physical systems," *Proc. IEEE*, vol. 104, no. 5, pp. 1013–1024, May 2016.
[24] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-time scheduling for WirelessHART networks," in *Proc. 31st IEEE Real-Time Syst. Symp.*, Nov. 2010, pp. 150–159.
[25] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "End-to-end delay analysis for fixed priority scheduling in WirelessHART networks," in *Proc. 17th IEEE Real-Time Embedded Technol. Appl. Symp.*, Apr. 2011, pp. 13–22.
[26] A. Karalis, D. Zorbas, and C. Douligeris, "Collision-free advertisement scheduling for IEEE 802.15.4-TSCH networks," *Sensors*, vol. 19, no. 8, p. 1789, 2019.
[27] C. Vallati, S. Brienza, G. Anastasi, and S. K. Das, "Improving network formation in 6TiSCH networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 1, pp. 98–110, Jan. 2019.
[28] M. Mohamadi, B. Djamaa, M. R. Senouci, and A. Mellouk, "FAN: Fast and active network formation in IEEE 802.15.4 TSCH networks," *J. Netw. Comput. Appl.*, vols. 183–184, no. 6, 2021, Art. no. 103026.
[29] A. Karalis, D. Zorbas, and C. Douligeris, "Optimal initial synchronization time in the minimal 6TiSCH configuration," *IEEE Access*, vol. 9, pp. 69316–69334, 2021.
[30] J. Lin and W. Liang, "Polling in the frequency domain: A new MAC protocol for industrial wireless network for factory automation," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 20, no. 4, pp. 211–222, 2015.
[31] H.-J. Korber, H. Wattar, and G. Scholl, "Modular wireless real-time sensor/actuator network for factory automation applications," *IEEE Trans. Ind. Informat.*, vol. 3, no. 2, pp. 111–119, May 2007.
[32] Q. Guangming and Y. Chao, *Real Time Wireless Connection Scheme for Multi-Nodes*, vol. 48, no. 2. Beijing, China: Chongqing Southwest Information Co. Ltd., China Computer Science, 2021, pp. 446–463. [Online]. Available: https://www.jsjkx.com/CN/article/openArticlePDF.jsp?id=20380

GUANGMING QIAN was born in 1963. He is currently a Professor with the College of Information Science and Engineering, Hunan Normal University, Changsha, China. His research interests include real-time systems, embedded technology, and wireless networks.

RI KANG was born in 1996. He is currently a Graduate Student with the College of Information Science and Engineering, Hunan Normal University, Changsha, China. His research interest includes real-time wireless networks.