IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 29, 2021, accepted June 15, 2021, date of publication July 1, 2021, date of current version July 9, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3093893*

# Traffic-Adaptive CFP Extension for IEEE 802.15.4 DSME MAC in Industrial Wireless Sensor Networks

**SANG-WOO LEE[1], (Graduate Student Member, IEEE), JUNG-HYOK KWON[1], (Member, IEEE), XUE ZHANG[2], AND EUI-JIK KIM[1], (Senior Member, IEEE)**

[1]School of Software, Hallym University, Chuncheon 24252, South Korea
[2]College of Ocean Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China

Corresponding author: Eui-Jik Kim (ejkim32@hallym.ac.kr)

This work was supported by the Hallym University Research Fund, in 2021, under Grant HRF-202103-008.

**ABSTRACT** This paper presents a traffic-adaptive contention-free period (CFP) extension (TaCFPext) protocol for IEEE 802.15.4 deterministic and synchronous multichannel extension (DSME) medium access control (MAC), which aims to satisfy the traffic adaptability requirement of industrial wireless sensor networks (IWSNs). The legacy DSME standard has limitations in accommodating highly varying traffic load in IWSNs due to its fixed multi-superframe structure. TaCFPext enables a node to adaptively use a contention access period (CAP) as an extended CFP (extCFP) on the demands of traffic loads in a distributed manner. The node starts TaCFPext when it determines that the CFP of the current multi-superframe is insufficient to accommodate the traffic load. Then, the node selects a CAP to be used as an extCFP, on which it is allocated an extended guaranteed time slot (extGTS). When the traffic load decreases, the extGTS is deallocated before the GTS in CFP, and the extCFP returns to CAP again. An experimental simulation was performed to verify the superiority of TaCFPext. The results demonstrated that TaCFPext outperforms the legacy DSME for aggregate throughput and average delay under various traffic conditions.

**INDEX TERMS** CFP extension, DSME MAC, industrial Internet of Things, industrial wireless sensor network, traffic adaptability.

## I. INTRODUCTION

Industrial wireless sensor networks (IWSNs) are essential for the Industrial Internet of Things (IIoT) framework, which can be widely adopted in various industrial applications such as environmental sensing, monitoring, control, and automation. In IWSNs, a number of sensors and actuators are connected to sense, collect, and deliver data from their internal states or the external environment through industrial wireless network technologies such as IEEE 802.15.4, Wireless HART, and ISA110.11a [1]–[7]. Compared to other networks, an IWSN requires critical performance imposed by industrial applications such as latency, reliability, and traffic adaptability [8].

Recently, the IEEE 802.15.4-2015 standard has been considered de facto for IWSNs because it provides the deterministic and synchronous multichannel extension (DSME) and time-slotted channel hopping (TSCH) medium access control (MAC) schemes, which commonly aim to ensure a deterministic delay and high reliability performance by using multiple channels and scheduled contention-free transmission [9], [10]. DSME and TSCH have advantages in different traffic conditions [11]–[15]. DSME provides high scalability by supporting slot scheduling in carrier sense multiple access with collision avoidance (CSMA/CA) manner in the contention access period (CAP) section; thus, it is robust under dynamic traffic conditions [16]–[20]. In contrast, TSCH is suitable for static traffic conditions because many parts of its slot scheduling are left to an upper layer, IETF 6TiSCH [21]–[25]. Therefore, DSME is a promising MAC solution to satisfy the traffic adaptability requirement of IWSNs.

However, DSME can suffer from inefficiency in the highly varying traffic load environments of IWSNs. DSME uses a multi-superframe structure composed of multiple

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

superframes, each of which is divided into a fixed CAP to contention-free period (CFP) ratio determined by several MAC parameters. In the CAP, the nodes exchange control packets for allocating a guaranteed time slot (GTS) using the contention-based CSMA/CA. Then, in the CFP, the nodes transmit data packets using the allocated GTSs. Under heavy traffic load, when there is no space to allocate new GTSs in the CFP of the current multi-superframe, the node should defer data packets that could not be sent to the next multi-superframe or drop them.

Furthermore, the waste of bandwidth is caused by not using the remaining CAPs. DSME can enable the CAP reduction option: the node changes the CAPs for remaining superframes except for the first superframe in a multi-superframe to a CFP to enable additional GTS allocation. However, this CAP reduction option should be enabled at the start of the network, and even if the traffic load decreases again, the CAP reduction option cannot be disabled again. This may prevent slot scheduling for new data packets due to a lack of CAPs.

Many studies have been conducted to improve adaptability to changes in the traffic load in the DSME networks. Kauer *et al.* [26] proposed a decentralized slot scheduling method for wireless multi-hop DSME networks; the nodes predict the amount of traffic to be injected into each communication link in advance, and allocate or deallocate GTSs accordingly. Lee and Chung [27] proposed a slot scheduling method for moving nodes in a DSME network. In [27], the nodes periodically transmit the slot allocation information, such as a list of allocated GTSs and the number of packets to be transmitted, to the coordinator. With this information, the coordinator reallocates the GTSs assigned to the moved node to another node.

Battaglia *et al.* [28] proposed the concept of a shareable GTS for the DSME networks, which can be redundantly allocated to one or more periodic traffics with a period longer than the length of the multi-superframe, thereby improving the usability of the CFP. However, these studies commonly use the fixed multi-superframe structure of the DSME standard, which is difficult to adapt to highly varying traffic loads.

The authors in [29], [30] addressed this problem by considering dynamic adjustments for the DSME multi-superframe structure according to the change in traffic load. Sahoo *et al.* [29] proposed a dynamic GTS allocation method to reduce the retransmission delay in the DSME networks. In [29], the coordinator recognizes whether each transmission in GTSs in the CFP of one superframe is successful or failed through the beacon, including a group acknowledgment (ACK). Then, at the start of the next superframe, it uses as many slots in the CAP as the number of failed transmissions for retransmissions.

Kurunathan *et al.* [30] proposed a dynamic multi-superframe tuning method to provide better quality of service (QoS) in the DSME networks. In [30], the coordinator maintains the network information, such as a list of nodes and the status of GTS allocations. With this information, the coordinator calculates the number of GTSs required in the network to toggle the CAP reduction option and adjust the length of the multi-superframe. However, in these studies, it is assumed that nodes constantly transmit the same amount of traffic. Therefore, when the amount of traffic transmitted by each node changes, these approaches cannot change the multi-superframe structure.

In this paper, we propose a traffic-adaptive CFP extension protocol (TaCFPext) for IEEE 802.15.4 DSME MAC to support adaptability for the highly varying traffic load of IWSNs. TaCFPext enables nodes to adaptively schedule their time and frequency slots in a distributed manner according to the demands of constantly varying traffic loads. The node starts TaCFPext when it determines that the CFP of the current multi-superframe is insufficient to accommodate the traffic load. Under high traffic load conditions, the nodes can temporarily use a CAP as an extended CFP (extCFP) to allocate additional GTSs. Then, if the traffic load decreases, the extCFP returns to CAP again.

TaCFPext operation consists of two procedures: 1) changeable CAP selection and 2) extended GTS (extGTS) allocation. In changeable CAP selection, a node verifies if it must use extCFP, and if so, selects which CAP in the multi-superframe to change to extCFP. In extGTS allocation, the node exchanges control packets to allocate extGTS within extCFP. Then, if the extGTS allocation is successful, the node and its one-hop neighbors verify the channel and slot where the new extGTS is allocated and update their multi-superframe in a distributed manner. We performed the simulation while considering various traffic loads to verify the superiority of TaCFPext over the legacy DSME. The simulation results demonstrated that TaCFPext can improve aggregate throughput by 20.50–71.03% and reduce average delay by 16.85–41.61% in the network where traffic variations occur.

The rest of this paper is organized as follows. Section II presents an overview of the DSME standard. The detailed operation of TaCFPext is described in Section III. The simulation configuration and results are presented in Section IV. Finally, we conclude this paper in Section V.

## II. IEEE 802.15.4 DSME OVERVIEW

### A. DSME MULTI-SUPERFRAME STRUCTURE

In the DSME networks, the nodes perform data communication based on the multi-superframe structure. Fig. 1 is an example of the DSME multi-superframe structure. The coordinator nodes periodically transmit an enhanced beacon (EB) for network synchronization. The time period between two consecutive EBs transmitted by the same coordinator node is referred to as beacon interval (BI). The BI is composed of multiple multi-superframes, each of which includes multiple superframes. Each superframe is divided into 16 slots numbered from 0 to 15 and consists of a beacon frame, CAP, and CFP.

In the beacon frame (slot 0), the coordinator nodes transmit EB. In CAP, which starts following the beacon frame and ends

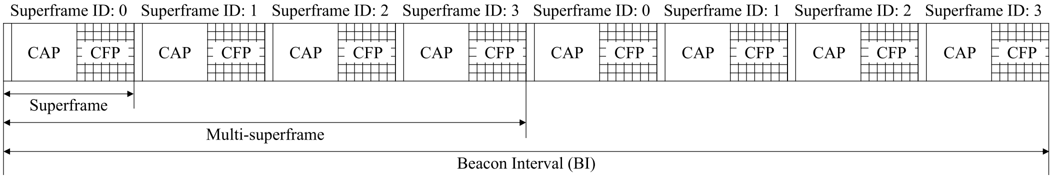

**FIGURE 1.** **DSME multi-superframe structure.**

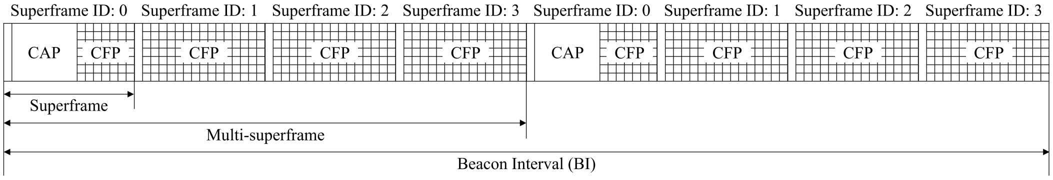

**FIGURE 2.** **DSME multi-superframe structure with CAP reduction option.**

before slot 9, the nodes transmit aperiodic traffic and control packets using the CSMA/CA. In CFP (slots 9–15), each slot is used as a GTS, exclusively allocated to a specific sender-receiver pair. In the GTSs, the nodes transmit periodic traffic using time division multiple access (TDMA). The DSME multi-superframe structure is defined by three configurable MAC parameters—*macSuperframeOrder* (SO), *macMultisuperframeOrder* (MO), and *macBeaconOrder* (BO)—chosen so that $0 \leq \text{SO} \leq \text{MO} \leq \text{BO} \leq 14$ and determines the following related values:

- Number of multi-superframes in a BI: $2^{(BO-MO)}$
- Number of superframes in a multi-superframe: $2^{(MO-SO)}$
- Superframe duration (SD)
  : $aBaseSuperframeDuration \times 2^{SO}$ symbols
- Multi-superframe duration (MD)
  : $aBaseSuperframeDuration \times 2^{MO}$ symbols
- Beacon interval (BI)
  : $aBaseSuperframeDuration \times 2^{BO}$ symbols

where *aBaseSuperframeDuration* is a constant value of 960 symbols.

Furthermore, DSME enables the use of the CAP reduction option. Fig. 2 is the DSME multi-superframe structure with the CAP reduction option. If the CAP reduction is enabled, only the first superframe of each multi-superframe includes a CAP, and other superframes are used as CFPs, each of which contains 15 GTSs.

### B. CHANNEL DIVERSITY

The IEEE 802.15.4 standard defines two channel diversity methods for the DSME networks: channel hopping and channel adaptation. In channel hopping, the allocated GTS hops over the predefined series of frequency channels (hopping sequence) to smooth the impact of external interference. In channel adaptation, the sender node allocates GTSs in the single frequency channel or different frequency channels to the receiver node based on the knowledge of current link quality. If the link quality of the allocated GTS degrades, it is recommended to deallocate the existing GTS and allocate new GTS in the frequency channel with superior link quality. Our study considers channel hopping as the base channel diversity method because it provides higher reliability than channel adaptation due to robustness to external interference.

In channel hopping, the hopping sequence is the same for all nodes, but each node starts hopping at a different channel frequency. The channel frequency at which the node starts hopping is determined by the channel offset of the node. The channel offset is selected when a node joins the network, and neighboring nodes avoid having the same channel offset. When two nodes want to communicate through the GTS, the sender and receiver nodes use the channel frequency determined by the receiver node's channel offset.

The channel frequency used during the GTS is derived as follows:

$$\begin{aligned} C &= macHoppingSequenceList[m], \\ m &= (i + j \times l + macChannelOffset \\ &\quad + macPanCoordinatorBsn) \\ &\quad \%\ macHoppingSequenceLength \end{aligned} \tag{1}$$

where *macHoppingSequenceList* is the hopping sequence, *macHoppingSequenceList*[*m*] is the *m*-th channel frequency number in *macHoppingSequenceList*, *i* is the index of the slot in a CFP, *j* is the SD index which indicates the index of the superframe in a BI, *l* is the number of slots in a CFP, *macChannelOffset* is the channel offset of the receiver node, *macPanCoordinatorBsn* is the sequence number of EB sent by the PAN coordinator, and *macHoppingSequenceLength* is the length of the hopping sequence.

Fig. 3 illustrates channel scheduling in channel hopping. The hopping sequence is [1, 2, 3, 4, 5, 6, 7], and the two nodes use 1 and 3 as channel offsets, respectively. The node using the channel offset of 1 (Node 1) starts hopping from a channel frequency of 1. Likewise, the node using the channel offset of 3 (Node 2) starts hopping from a channel frequency of 3.

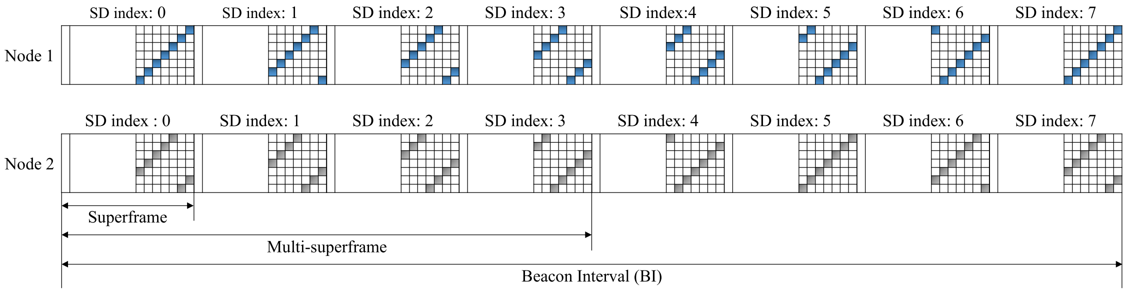

**FIGURE 3.** Example of channel scheduling in channel hopping.

## C. DSME GTS ALLOCATION AND MANAGEMENT

For allocating and managing the GTSs, the nodes in a DSME network maintain two data attributes: a slot allocation bitmap (SAB) and an allocation counter table (ACT). The SAB is a bitmap that stores the status of the GTSs in a multi-superframe allocated to the node and its one-hop neighbors. Fig. 4 illustrates the SAB bitmap format in channel hopping. The SAB includes multiple SAB sub-blocks as many as the number of superframes in a multi-superframe. One SAB sub-block consists of 7 bits if the CAP reduction option is disabled or 15 bits if enabled. In a SAB sub-block, each bit indicates whether one slot in the CFP is already used as a GTS (set to one) or available (set to zero).

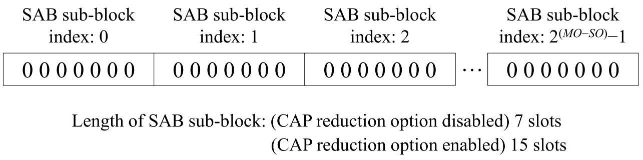
SAB sub-block index: 0
SAB sub-block index: 1
SAB sub-block index: 2
SAB sub-block index: $2^{(MO-SO)}-1$
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
···
0 0 0 0 0 0 0
Length of SAB sub-block: (CAP reduction option disabled) 7 slots
(CAP reduction option enabled) 15 slots

**FIGURE 4.** Bitmap format of SAB in channel hopping.

Furthermore, ACT is a data table containing the information for managing the allocated GTSs such as superframe ID, slot ID, channel ID, direction, address, and idle counter. The superframe ID is the index of the superframe to which the GTS is allocated in a multi-superframe. The slot ID is the index of the slot used as the GTS in a CFP. The channel ID is the channel offset of the receiver node. The direction specifies whether the node is a sender (TX) or a receiver (RX). The address specifies the sender's address if the direction is RX or the receiver's address if TX. The idle counter counts the number of idle multi-superframes since the last usage of the GTS.

Fig. 5 illustrates the GTS allocation procedure, where Node B requests GTS allocation by sending the *DSME GTS Request* command to Node C. The *DSME GTS Request* command specifies the command type (allocation), number of required GTSs, preferred superframe, preferred slot, and SAB sub-blocks.

Upon receiving the *DSME GTS Request* command, Node C compares its allocation information (SAB) with the SAB sub-blocks in the received *DSME GTS Request* command. Then, it schedules the GTSs and broadcasts the *DSME GTS Response* command for approving GTS allocation. The *DSME GTS Response* command includes the command type, the result for the request (approval), the address of the sender node, the channel offset of the receiver node, and the SAB sub-blocks indicating newly allocated GTSs.

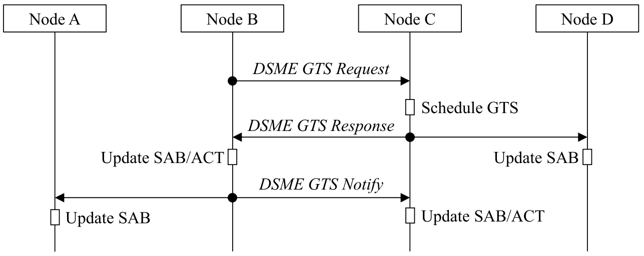

**FIGURE 5.** GTS allocation procedure.

Upon receiving the *DSME GTS Response* command, Node B updates its SAB and ACT using the SAB sub-blocks in the received *DSME GTS Response* command and then broadcasts the *DSME GTS Notify* command. The *DSME GTS Notify* command includes the same information in the *DSME GTS Response* except for the sender's address. The address of the *DSME GTS Notify* command indicates the address of the receiver node.

Upon receiving the *DSME GTS Notify* command, Node C updates its SAB and ACT. Furthermore, one-hop neighbor nodes, Nodes A and D, overhear the *DSME GTS Response* and *DSME GTS Notify* commands, respectively. Then, they compare the SAB sub-blocks in the commands with their ACTs. If the newly allocated GTSs do not conflict with the existing GTSs, they update their SABs. Otherwise, they send *DSME Duplicated Allocation Notification* command to the sender node or the receiver node to inform that the newly allocated GTS is not valid and should be canceled.

Likewise, the GTS deallocation procedure is performed by exchanging *DSME GTS Request*, *DSME GTS Response*, and *DSME GTS Notify* commands. The GTS deallocation procedure is initiated by both sender and receiver nodes when the allocated GTS expires. The expiration of GTS is detected by verifying the idle counter in ACT. The sender node increments its idle counter by one if an ACK frame has not been received in the GTS. The receiver node increments its idle counter by one when a data frame has not been received

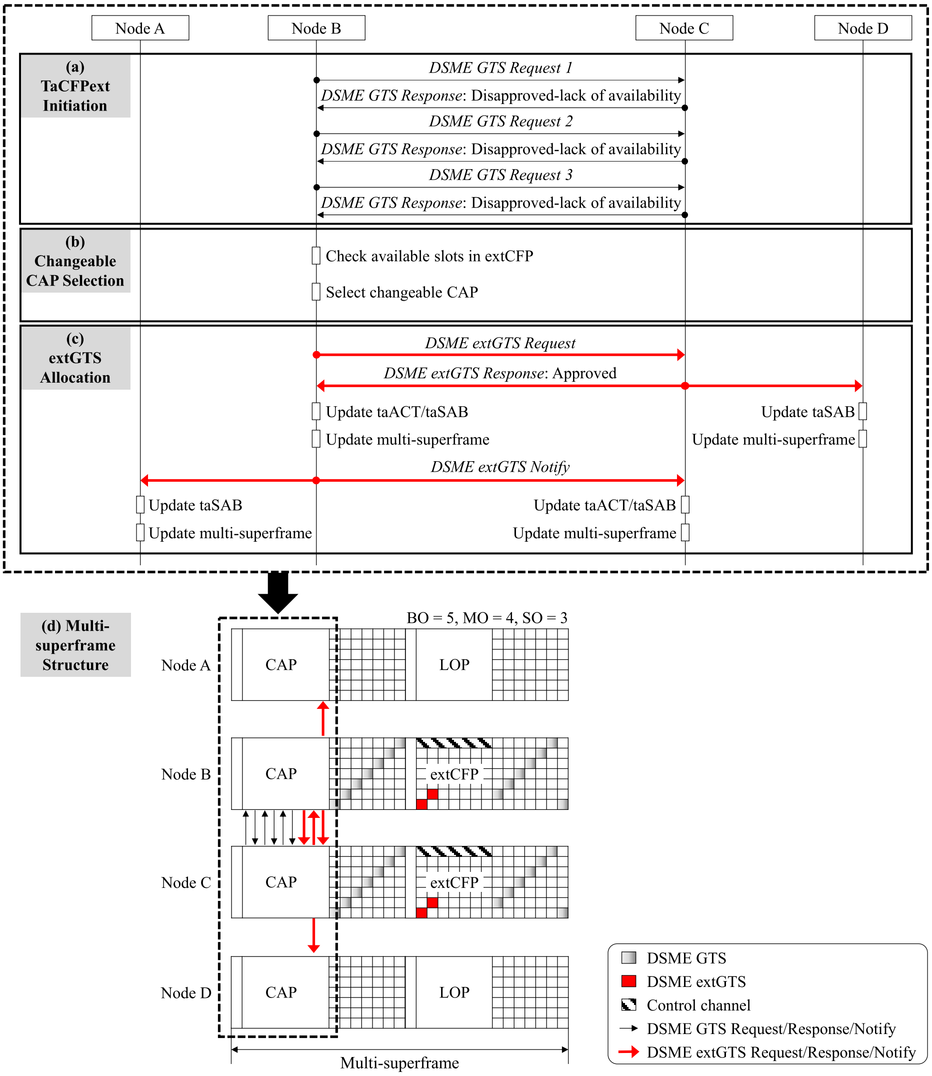

**FIGURE 6.** Overall operation of TaCFPext.

in the GTS. The sender or receiver node determines that the GTS has expired when its idle counter is greater than *macDsmeGtsExpirationTime*, which is 7 by default.

## III. DESIGN OF TaCFPext

This section describes the detailed operation of the TaCFPext protocol, which enables nodes to adaptively schedule their time and frequency slots in a distributed manner according to the demands of constantly varying traffic loads.

### A. OVERALL OPERATION

In TaCFPext, a node temporarily uses a CAP as an extCFP when it can no longer allocate a GTS to exchange data packet within the CFP of the current multi-superframe due to high traffic load conditions. Then, if the traffic load decreases again, the extCFP returns to the CAP. Fig. 6 illustrates the overall operation of the TaCFPext. When a node determines that the CFP in the current multi-superframe does not have sufficient space to allocate its GTS, it starts the TaCFPext operation to extend the CFP. In (a) of Fig. 6, Node B tries to allocate a GTS to communicate with Node C by sending a *DSME GTS Request* command but fails three times in a row. Node B assumes no space to allocate the GTS in the CFP of the current multi-superframe and runs the TaCFPext, which consists of two procedures: 1) changeable CAP selection and 2) extGTS allocation.

In the changeable CAP selection procedure, a node verifies if it must use extCFP, and if so, selects which CAP in the multi-superframe to be changed to extCFP. If there is already an extCFP in the current multi-superframe with sufficient slots for new extGTSs, the node proceeds to the next procedure to allocate them. Otherwise, the node chooses one of the CAPs except for the CAP of the first superframe to use it as an extCFP. The CAP of the first superframe is not used as an extCFP to ensure the common CAP for the nodes in a DSME network. In (b) of Fig. 6, Node B first verifies whether the current multi-superframe has an extCFP with sufficient slots for new extGTSs, and then chooses the CAP of the second superframe as a changeable CAP, in which it may be allocated extGTS via the extGTS allocation procedure.

In the extGTS allocation procedure, the node is allocated extGTS in the selected changeable CAP by sequentially exchanging *DSME extGTS Request*, *DSME extGTS Response*, and *DSME extGTS Notify* commands, which contain the sub-blocks of traffic-adaptive SAB (taSAB) to express the allocation information in the CAPs and extCFPs. In (c) of Fig. 6, Node B starts an extGTS allocation procedure by sending a *DSME extGTS Request* command to Node C. The *DSME extGTS Request* command contains the taSAB sub-block to express the allocation information of the CAP in the second superframe.

Upon receiving the *DSME extGTS Request* command, Node C compares its allocation information and the taSAB sub-block in the *DSME extGTS Request* command. After verifying whether there are sufficient slots for new extGTSs in the CAPs of both nodes' second superframes, Node C determines the extGTS allocation and responds with the *DSME extGTS Response* command, which contains a taSAB sub-block indicating new extGTSs.

Node B, which receives the *DSME extGTS Response* command updates its taSAB and traffic-adaptive counter table (taACT) to store the new extGTSs and then sends the *DSME extGTS Notify* command to Node C. The *DSME extGTS Notify* command contains the same taSAB sub-block as the *DSME extGTS Response* command to indicate new extGTSs.

Upon receiving the *DSME extGTS Notify* command, Node C updates its taSAB and taACT. Furthermore, Nodes A and D overhear the *DSME extGTS Response* and *DSME extGTS Notify* commands and update their taSABs to maintain up-to-date allocation information in CAPs and extCFPs. After updating the taACT and taSAB, Nodes A, B, C, and D update their multi-superframe structures using the updated taACT and taSAB.

Nodes in a DSME network may have different multi-superframe structures as a result of TaCFPext operations. When a pair of nodes use the CAP in a specific superframe as an extCFP, their one-hop neighbors designate their CAPs as the listen-only-period (LOP) during the extCFP. In extCFP, the allocated extGTS hops over all the multi-channels except for the control channel. In the LOP, the nodes (Nodes A and D in (d) of Fig. 6) merely listen to the control channel to ensure their neighbors' (Nodes B and C in (d) of Fig. 6) collision-free transmission in extCFP. When they receive a *DSME extGTS Response* or *DSME extGTS Notify* command from other nodes, they update their allocation information of the multi-superframe.

### *B. TaCFPext INITIATION AND CHANGEABLE CAP SELECTION*

As described earlier, the TaCFPext is initiated under high traffic load conditions where the CFP of the current multi-superframe does not have space to allocate an additional GTS. Thus, a node in the DSME network should first recognize this high traffic load condition in a distributed manner before the TaCFPext operation is initiated. When a pair of nodes exchange *DSME GTS Request* and *DSME GTS Response* commands to allocate the GTS, they compare their SAB for knowledge of CFP availability. If the number of SAB sub-blocks contained in a *DSME GTS Request* command is smaller than the number of superframes in a multi-superframe, a pair of nodes should exchange the commands multiple times to compare their entire CFPs.

Assume that the number of SAB sub-blocks contained in a *DSME GTS Request* command is $k$. Accordingly, it is necessary to successively exchange the *DSME GTS Request* and *DSME GTS Response* commands by the number of superframes in a multi-superframe divided by $k$ to compare every CFP in the multi-superframe. Therefore, when a node fails to allocate the GTS consecutively $2^{(MO-SO)}/k$ times, it determines that the current DSME network is under high traffic load conditions and starts the TaCFPext operation.

After TaCFPext is initiated, the node performs a changeable CAP selection procedure to determine which CAP in the multi-superframe to be changed to extCFP. If there are already extCFPs with sufficient slots for new extGTSs, the node does not select an additional CAP, but uses the existing extCFPs to allocate extGTS. Otherwise, the node should select some of CAPs or LOPs in the multi-superframe to change into extCFPs.

Among CAP and LOP, the node should select the LOP first to minimize the change in use for the CAP of one-hop neighbors. The node can select the LOP with sufficient slots for new extGTSs by referring the bitmap that stores extGTSs allocated to the node and its one-hop neighbors. If the number of available slots within extCFPs and LOPs is insufficient, the node selects CAP except for the CAP of the first superfame, which changes to extCFP as of the next multi-superframe. The TaCFPext ensures a common CAP for the nodes in a DSME network by excluding the CAP of the first superframe from the changeable CAP selection procedure.

The node uses two new attributes for the changeable CAP selection procedure: traffic-adaptive SAB (taSAB) and traffic-adaptive ACT (taACT). The former is a two-dimensional bitmap that stores the status of extGTSs in a multi-superframe allocated to itself and one-hop neighbors. The latter is a data structure that stores information for managing the extGTSs assigned to it.

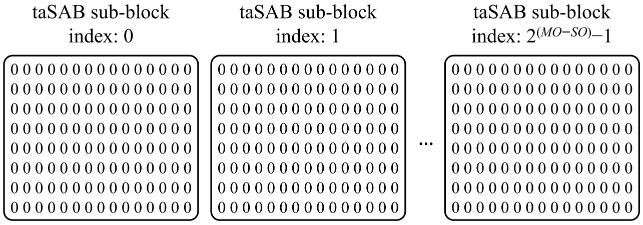

**FIGURE 7. Bitmap format of taSAB.**

Fig. 7 illustrates a bitmap format of taSAB. The successive taSAB sub-blocks are represented by two-dimensional bitmaps; the columns and rows represent channel offsets and slots, respectively, describing the information of extGTS allocation for all channel offsets within extCFP. This two-dimensional structure of taSAB enables the node to recognize all extGTSs allocated to one-hop neighbors, converting CAP to LOP in a distributed manner. In channel hopping of the legacy DSME, the SAB sub-block is represented as a one-dimensional bitmap that contains GTS allocation information only for one channel offset used by the node. Furthermore, taACT maintains the same field structure as the ACT in the legacy DSME, through which the node can manage the extGTSs within extCFP.

The detailed operation of changeable CAP selection is depicted in Algorithm 1, which includes the above-described two attributes (**taACT**, **taSAB**) and three variables (*channelOffset*, *NumberOfChannels*, and *numSlots*) to verify the presence of existing extCFP or LOP in a multi-superframe and, if present, available slots. The table entry of **taACT** includes the information of extGTS such as its superframe ID in a multi-superframe, slot ID in a superframe, and channel offset. The superframe IDs of table entries are represented by an array, as in Eq. (2).

$$\textbf{taACT}.\textbf{sid} = [sid_1, sid_2, \ldots, sid_i], \quad 1 \le i \le N \tag{2}$$

where $sid_i$ is the superframe ID of the $i$-th table entry in **taACT**, and $N$ is the number of the extGTSs allocated to the node. **taSAB** is represented by a three-dimensional array with [*superframe ID*][*slot ID*][*channel offset*] as an element. *channelOffset*, *NumberOfChannels*, and *numSlots* indicate the channel offset of extGTS to be allocated, the number of physical channels supported by the IEEE 802.15.4 physical (PHY) layer, and the number of slots needed to allocate new extGTSs, respectively.

Moreover, the algorithm uses four additional attributes (**extCFP**, **LOP**, **CAP**, **conArray**, and **ocpSC**) to track the status of CAPs in a multi-superframe. **extCFP**, **LOP**, and **CAP** are data structures that store the indices of superframes to which extCFP, LOP, and CAP belong, respectively. **conArray** is a concatenated array of **extCFP**, **LOP**, and **CAP**, used to select changeable CAPs. **ocpSC** is a vector of occupied slot counters, for which $2^{(MO-SO)}$ elements represent the number of slots occupied by extGTSs within extCFPs and LOPs. *temp_sid* and *slotCnt* are the variables for verifying the superframe IDs stored in **conArray** and counting the number of available slots in the changeable CAPs, respectively. Finally, **SID** is a returning array that includes the superframe IDs of the changeable CAPs.

After **ocpSC**, **extCFP**, **LOP**, **conArray**, **CAP**, **SID**, **slotCnt**, and *temp_sid* are initialized, the node investigates **taACT** and **taSAB** successively to verify the superframe ID of the existing extCFP, LOP, or CAP and the available slots within the extCFP and LOP.

The node first investigates the superframe IDs of table entries in **taACT** (**taACT.sid**) to verify the superframe to which the existing extCFP belongs, and appends it to **extCFP** if it is not an element of **extCFP**. Likewise, the node investigates the bits of **taSAB**, which are represented by **taSAB**[$i$][$j$][$k$] with three indices (superframe ID $i$, slot ID $j$, and channel offset $k$), to verify the superframe to which

**Algorithm 1** Changeable CAP Selection

```
INITIALIZE ocpSC to [$0_{(0)}, 0_{(1)}, \cdots, 0_{(2^{(MO-SO)}-1)}$], extCFP to empty array, LOP to empty
    array, CAP to empty array, conArray to empty array,
    SID to empty array, slotCnt to 0, temp_sid to 0
/* investigate taACT */
FOR each superframe ID in taACT.sid, $sid_i$, $i \in [1, N]$
    IF $sid_i \notin extCFP$
        extCFP [length(extCFP)] ← $sid_i$
    ENDIF
ENDFOR
/* investigate taSAB */
FOR each bit of taSAB, taSAB [$i$][$j$][$k$],
        $i \in \left[0, 2^{(MO-SO)} - 1\right], j \in [0, 7]$,
        $k \in [0, NumberOfChannels - 2]$
    IF taSAB [$i$][$j$][$k$] == 1
        IF $sid_i \notin$ extCFP && $sid_i \notin$ LOP
            LOP [length(LOP)] ← $sid_i$
        ENDIF
        IF $k$ == channelOffset % (NumberOfChannels-1)
            ocpSC [$i$] ← ocpSC [$i$] + 1
        ENDIF
    ENDIF
ENDFOR
FOR each superframe ID in a multi-superframe, $i$,
    $i \in \left[0, 2^{(MO-SO)} - 1\right]$
    IF $i \notin$ extCFP && $i \notin$ LOP && $i$ != 0
        CAP [length(CAP)] ← $i$
    ENDIF
ENDFOR
conArray ← concatenate (extCFP, LOP, CAP)
/* select changeable CAPs */
FOR each superframe ID in conArray, conArray
    [$i$], $i \in [0, size(conArray) - 1]$
    temp_sid ← conArray[$i$]
    IF ocpSC [temp_sid] < 8
        SID [length(SID)] ← temp_sid
        slotCnt ← slotCnt + (8–ocpSC [temp_sid])
    ENDIF
    IF slotCnt < numSlots
        BREAK
    ENDIF
ENDFOR
IF slotCnt < numSlots
    SID ← []
ENDIF
RETURN SID
```

the existing LOP belongs and the occupied slots within the extCFP and LOP. If the specific bit of **taSAB** is one, the node verifies whether its superframe ID $i$ is an element of **extCFP** and **LOP**. If the superframe ID $i$ is not an element of **extCFP**, it can be inferred that the corresponding slot is an extGTS allocated to the one-hop neighbor. Thus, the node appends the superframe ID $i$ to **LOP** if it is not an element of **LOP**.

The node then verifies whether the channel offset $k$ is the same as the channel offset of the extGTS to be allocated. If so, it can be inferred that the corresponding slot is an existing extGTS conflicting with the new extGTS to be allocated. Thus, the node increases the value of **ocpSC**[$i$] by one to count the number of occupied slots within the extCFP or LOP belonging to the $i$-th superframe in a multi-superframe. After investigating **taACT** and **taSAB**, the node appends the superframe IDs not included in **LOP** and **extCFP** to the **CAP**, except for the superframe ID of the first superframe.

After updating CAP, the three attributes, **extCFP**, **LOP**, and **CAP**, are concatenated and stored into **conArray**. Then, the node selects changeable CAPs based on the number of the occupied slots in each extCFP, LOP, or CAP. In **conArray**, the superframe IDs of extCFP, LOP, and CAP are sequentially arranged. With **conArray**, the node first stores the superframe ID of an extCFP into *temp_sid* and investigates the number of occupied slots in the extCFP (**ocpSC**[*temp_sid*]). If the number of occupied slots in the extCFP is less than eight, there are available slots in the extCFP. Therefore, in this case, the node appends the superframe ID of the extCFP to **SID** and adds the number of available slots in the extCFP (8–**ocpSC**[*temp_sid*]) to *slotCnt*.

If the number of available slots in the extCFP is less than the number of required slots (*numSlots*), the superframe ID of the next extCFP in **conArray** is appended to **SID** in the same way. Then, if the number of available slots in all extCFPs is less than *numSlots*, the superframe ID of LOP in **conArray** is appended to **SID**. Likewise, if the number of available slots in all extCFP and LOP is less than *numSlots*, the superframe ID of CAP in **conArray** is appended to **SID**. After updating **SID**, if *slotCnt* is greater than *numSlots*, **SID** is returned as changeable CAPs. Otherwise, an empty array is returned because the available slots are insufficient to allocate the required extGTSs.

### C. extGTS ALLOCATION AND MANAGEMENT

After the changeable CAP selection procedure is completed, a pair of nodes (sender and receiver) perform the extGTS allocation procedure through the exchange of three commands (*DSME extGTS Request*, *DSME extGTS Response*, and *DSME extGTS Notify*). The sender initiates extGTS allocation by transmitting a *DSME extGTS Request*, which contains the number of required GTSs, the preferred superframe, the preferred slot, and the taSAB sub-blocks of changeable CAPs.

Upon receiving the *DSME extGTS Request*, the receiver compares the taSAB sub-blocks in the received *DSME extGTS Request* with its taSAB, and, if the preferred slot is available, selects the preferred slot and its adjacent slots for new extGTSs. If not, the other available slots are selected for the new extGTSs. Then, the receiver responds with a *DSME extGTS Response* that contains the taSAB sub-blocks indicating the new extGTSs.

Upon receiving the *DSME extGTS Response*, the sender updates its taSAB and taACT and broadcasts a *DSME extGTS Notify* indicating the new extGTSs. Upon receiving the *DSME extGTS Notify command*, the receiver updates its taSAB and taACT. The one-hop neighbors of sender and receiver update their taSAB by overhearing the *DSME extGTS Response* or *DSME extGTS Notify*.

The extGTS deallocation is performed similarly by exchanging these three commands. The receiver and sender increment the counter value of the table entry in taACT by one whenever data frame and ACK frame are not received during extGTS, respectively. Then, when this counter value is greater than $macDsmeGtsExpirationTime/2$, it is assumed that the extGTS has expired, and the sender or receiver starts the extGTS deallocation procedure. In the legacy DSME, the GTS expires when the counter value of the ACT table entry is greater than *macDsmeGtsExpirationTime*. This enables extGTS to be deallocated before the GTSs in CFPs when traffic load decreases.

When extGTS allocation or deallocation occurs, the node and its one-hop neighbors convert the changeable CAP to extCFP or LOP, or vice versa. The nodes track these extCFP/LOP conversions in a distributed manner by maintaining an attribute, changeable CAP bitmap (CCB), that indicates the conversion status (CAP, LOP, and extCFP) of all CAPs in the multi-superframe. The CCB is a one-dimensional bitmap; the two-bit sub-block of CCB expresses CAP, LOP, and extCFP as 00, 01, and 10, respectively, as depicted in Fig. 8.

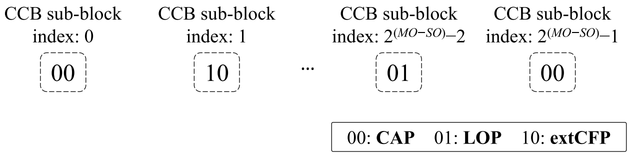
CCB sub-block index: 0
00
CCB sub-block index: 1
10
...
CCB sub-block index: $2^{(MO-SO)}-2$
01
CCB sub-block index: $2^{(MO-SO)}-1$
00
00: CAP 01: LOP 10: extCFP

**FIGURE 8. Bitmap format of CCB.**

After updating taSAB and taACT, the node performs the CCB update for each changeable CAP in the taSAB sub-blocks contained in the *DSME extGTS Response* or *DSME extGTS Notify* commands. The detailed operation of CCB update is depicted in Algorithm 2, which includes an attribute (**CCB**) and three variables (*node_extGTS_presence*, *node_extGTS_cnt*, and *all_extGTS_cnt*), to verify the status of changeable CAP and the extGTSs allocated in the changeable CAP, in addition to the attributes and variables used in Algorithm 1.

**CCB** is represented by the one-dimensional array which includes $2^{(MO-SO)}$ two-bit sub-blocks as elements. *node_extGTS_presence* is a Boolean variable indicating whether there is an extGTSs allocated to the node in a changeable CAP. *node_extGTS_cnt* and *all_extGTS_cnt* are counter values indicating the number of extGTSs allocated to the node and the total number of extGTSs in a changeable CAP, respectively.

Upon receiving the *DSME extGTS Response* or *DSME extGTS Notify* command, the node verifies its "Management Type" field. If it indicates "extGTS allocation" (*management_type* == 001), the node converts its changeable CAP to extCFP. If the node overhears the above command, the extGTS is allocated to the one-hop neighbor; thus, it changes the changeable CAP to LOP if the status of the changeable CAP is a CAP.

For "extGTS deallocation" (*management_type* == 000), the node investigates every superframe index ($sid_i$) in **taACT.sid** to verify if there are extGTSs allocated to it in the changeable CAP and count them. If a specific superframe index, $sid_i$, is equal to *changeableCAP_sid*, the node sets *node_extGTS_presence* to TRUE and increments *node_extGTS_cnt* by one. Then, the node investigates all bits of taSAB sub-block corresponding to changeable CAP (**taSAB**[*changeableCAP_sid*][$j$][$k$]) to count the number of all extGTSs in the changeable CAP. If a specific bit in taSAB sub-block is one, there is an extGTS in the changeable CAP; thus, the node increments *all_extGTS_cnt* by one.

After completing the investigation of **taACT.sid** and **taSAB**, the node determines the status conversion of the changeable CAP based on *node_extGTS_presence*, *node_extGTS_cnt*, and *all_extGTS_cnt*. If *node_extGTS_presence* is TRUE, an extGTS is allocated to the node in the changeable CAP, so the node converts its changeable CAP to extCFP. Otherwise, the node verifies if *all_extGTS_cnt* is greater than *node_extGTS_cnt*. If so, there are extGTSs allocated to the one-hop neighbors in the changeable CAP; thus, the node converts its changeable CAP to LOP. If *node_extGTS_presence* is FALSE, and *all_extGTS_cnt* is less than *node_extGTS_cnt*, the changeable CAP should remain a CAP.

In contrast to the GTS in CFPs, the extGTS hops over the channels according to the modified hopping sequence (*HoppingSequenceList_ext*), in which a control channel is excluded from the existing hopping sequence. The channel frequency used during the extGTS (*C_ext*) is derived as follows:

$$\begin{aligned} C\_ext &= HoppingSequenceList\_ext[m], \\ m &= (i + j \times l + macChannelOffset \\ &\quad + macPanCoordinatorBsn) \\ &\quad \%\ (macHoppingSequenceLength - 1) \end{aligned} \tag{3}$$

where *HoppingSequenceList_ext*[$m$] is the $m$-th channel frequency in the modified hopping sequence for extGTS, $i$ is the index of the slot in an extCFP, $j$ is the SD index that is the index of the superframe in a BI, $l$ is the number of slots within an extCFP (8), *macChannelOffset* is the channel offset of the receiver node, *macPanCoordinatorBsn* is the sequence number of EB sent by the PAN coordinator, and *macHoppingSequenceLength* is the hopping sequence length of the existing hopping sequence.

**Algorithm 2** CCB Update

```
INITIALIZE node_extGTS_presence to FALSE,
  node_extGTS_cnt to 0, all_extGTS_cnt to 0
/* extGTS allocation */
IF management_type == 001
  IF the destination address is the node itself
    CCB [changeableCAP_sid] ← 10 // extCFP
  ELSE
    IF CCB [changeableCAP_sid] == 00
      CCB [changeableCAP_sid] ← 01 // LOP
    ENDIF
  ENDIF
/* extGTS deallocation */
ELSE IF management_type == 000
    FOR each superframe index in taACT.sid,
    $sid_i$, $i \in [1, N]$
      IF == changeableCAP_sid
        node_extGTS_presence ← TRUE
        node_extGTS_cnt ← node_extGTS_cnt
          + 1
      ENDIF
    ENDFOR
    FOR each bit of taSAB sub-block for
      changeablCAP_sid in taSAB,
      taSAB[changeablCAP_sid][j][k], $j \in [0, 7]$,
      $k \in [0, NumberOfChannels - 2]$
      IF taSAB[changeableCAP_sid][j][k] == 1
        all_extGTS_cnt ← all_extGTS_cnt + 1
      ENDIF
    ENDFOR
    IF node_extGTS_presence == TRUE
      CCB [changeableCAP_sid] ← 10 // extCFP
    ELSE IF all_extGTS_cnt > node_extGTS_cnt
      CCB [changeableCAP_sid] ← 01 // LOP
    ELSE
      CCB [changeableCAP_sid] ← 00 // CAP
    ENDIF
ENDIF
```

## IV. PERFORMANCE EVALUATION

We evaluated TaCFPext performance through experimental simulations using the MATLAB simulator. The simulation results were compared with those of the legacy DSME of the IEEE 802.15.4 standard to verify the effectiveness of TaCFPext. In the following subsections, we present the simulation settings and configuration and discuss the results in detail.

### A. SIMULATION CONFIGURATION

In the simulation, we consider network instances of 20, 40, 60, 80, 100, 120, 140, 160, 180, and 200 nodes. In each

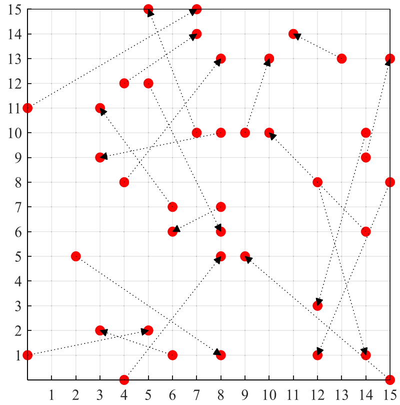

**FIGURE 9.** Example of network deployment with 40 nodes.

instance, the nodes are randomly deployed in a 15 × 15 $m^2$ region and have a transmission range of 10 m. Fig. 9 is an example of network deployment with 40 nodes. Each node forms a sender-receiver pair with one of its one-hop neighbors. The sender transmits a certain number of data packets per multi-superframe to the receiver, for which the dedicated GTSs are allocated in the CFP. If all required GTSs cannot be allocated due to high contention in the CAP or insufficient space in CFPs, the node drops the data packets that could not be transmitted in the current multi-superframe.

We verified the TaCFPext's adaptability to varying traffic by considering two traffic scenarios: static and dynamic. In a static traffic scenario, the sender transmits the same number of packets (7 packets) in every multi-superframe. In contrast, in a dynamic traffic scenario, nodes in the network are divided into two groups, each consisting of the same number of nodes. The nodes belonging to the same group generate the same pattern of dynamic traffic with a period of eight beacon intervals. Nodes in one group transmit 21 packets per multi-superframe (high traffic) in the third and fourth beacon intervals and only one packet per multi-superframe (low traffic) in the remaining beacon intervals.

Furthermore, nodes in other group generate high traffic in the seventh and eighth beacon intervals. This configuration for the dynamic traffic scenario is a method to track the operational patterns of GTS allocation and deallocation of nodes under varying traffic load environments.

TaCFPext performance was compared with that of the legacy DSME regarding the number of allocated GTSs, packet drop ratio, GTS allocation delay, average delay, aggregate throughput, and fairness index. For the legacy DSME, we considered both DSME network environments with and without the CAP reduction option. In both TaCFPext and the legacy DSME, we assumed that all nodes maintain the same DSME multi-superframe structure in which SO, MO, and BO are set to 3, 5, and 6, respectively. The simulation was iterated 100 times. The detailed simulation parameters are listed in Table 1.

**TABLE 1.** Simulation parameters.

| Parameters | Values |
|---|---|
| PHY | IEEE 802.15.4 |
| Modulation | O-QPSK (4bits/symbol) |
| Bit rate | 250 kbps |
| Number of channels | 16 |
| SO, MO, BO | 3, 5, 6 |
| *AIFS* | 12 symbols |
| *LIFS* | 40 symbols |
| *aUnitBackoffPeriod* | 20 symbols |
| *aBaseSlotDuration* | 60 symbols |
| PHY header | 6 bytes |
| MAC header | 10 bytes |
| Payload size | 127 bytes |
| ACK size | 11 bytes |
| Multi-superframe duration | 491.52 ms |
| Slot length | 7.96 ms |
| Superframe duration | 112.88 ms |

### B. SIMULATION RESULTS

Fig. 10 illustrates the variations in the number of allocated GTSs over time in the static traffic scenario with 100 nodes (50 sender-receiver pairs). The upper and lower figures in Fig. 10 illustrate the number of allocated GTSs during the total simulation time (60 s) and from the start of the simulation to 5 seconds, respectively. Each node in the network should allocate seven GTSs to transmit seven packets in one multi-superframe because one packet is transmitted during one slot. Thus, it is necessary to allocate 350 GTSs in one multi-superframe to accommodate the total traffic load for 50 node pairs.

Commonly, as depicted in the upper figure of Fig. 10, the number of allocated GTSs increases and then maintains a particular value. Given the interval in which the number of allocated GTSs is kept constant, the number of allocated GTSs for TaCFPext and DSME with the CAP reduction (350) is larger than that of the legacy DSME (323). The TaCFPext and the DSME with the CAP reduction can retain sufficient space to allocate many GTSs through the extCFP and the CAP reduction option. In contrast, the legacy DSME maintains a relatively limited length of CFP compared to TaCFPext and the DSME with the CAP reduction. Thus, in the legacy DSME, some nodes fail to allocate GTSs, and the number of allocated GTSs did not reach 350.

Furthermore, as depicted in the lower figure of Fig. 10, the number of allocated GTSs for TaCFPext and the legacy DSME increases more sharply than that of the DSME with the CAP reduction. In the TaCFPext and the legacy DSME, the nodes attempt to allocate GTSs in every superframe because every superframe in a multi-superframe contains a CAP. However, in the DSME with the CAP reduction, the nodes can attempt to allocate GTSs only in the first superframe because only the first superframe in the multi-superframe includes the CAP. Therefore, the number of allocated GTSs for the DSME with the CAP reduction increases only in the first superframe in the multi-superframe.

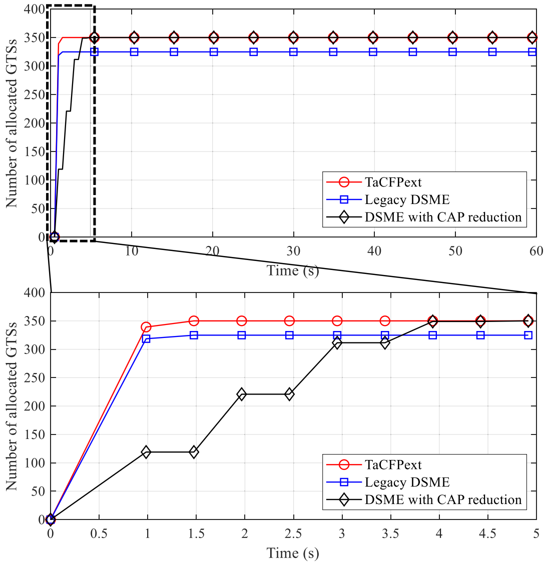

**FIGURE 10.** Number of allocated GTSs in static traffic scenario.

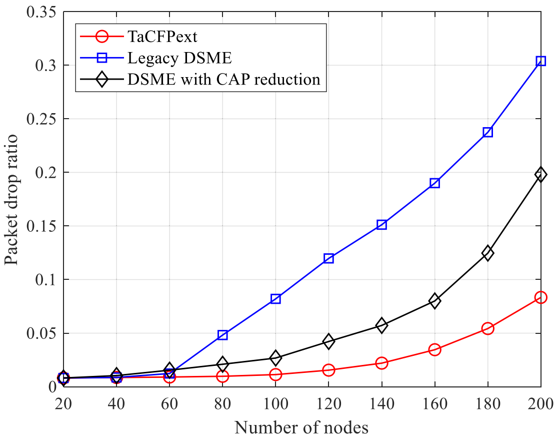

**FIGURE 11.** Packet drop ratio in static traffic scenario.

Fig. 11 illustrates the packet drop ratio in the static traffic scenario for varying numbers of nodes. In the simulation, if the node fails to allocate the required GTSs in a multi-superframe, it drops the packets that cannot be transmitted in the current multi-superframe. The packet drop ratio increases as the number of nodes increases because the nodes are more likely to fail to allocate their GTSs due to high contention in the CAP.

Furthermore, some nodes may fail to allocate GTSs due to insufficient space in CFPs, increasing the packet drop ratio. The TaCFPext exhibits a smaller packet drop ratio than the legacy DSME and the DSME with the CAP reduction. In the TaCFPext, every superframe in the multi-superframe contains the CAP, providing nodes with more opportunities to allocate GTSs under high contention.

Moreover, sufficient space for GTS allocation is provided by changing the CAP to the extCFP even if the number of required GTSs increases. However, in the legacy DSME, when the number of nodes is larger than 60, the packet drop ratio sharply increases due to insufficient space in CFPs. In the DSME with the CAP reduction, the multi-superframe contains only one CAP despite sufficient CFP space in a multi-superframe. Therefore, the DSME with the CAP reduction suffers from the limited length of CAP when the number of nodes increases.

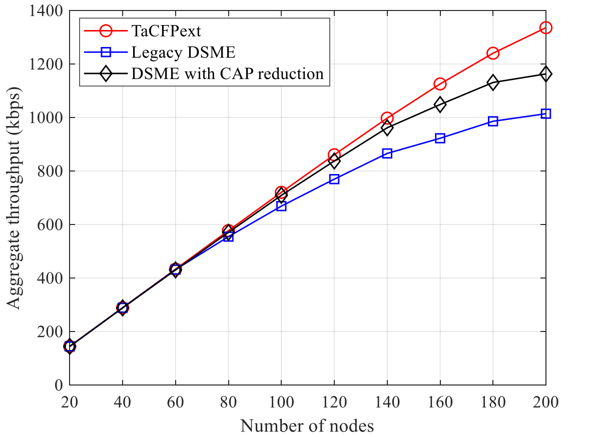

**FIGURE 12.** Aggregate throughput in static traffic scenario.

Fig. 12 illustrates the aggregate throughput for varying numbers of nodes in the static traffic scenario. As the number of nodes increases, the amount of traffic injected into the network increases, increasing aggregate throughput. When the number of nodes exceeds 60, the aggregate throughputs of TaCFPext, the legacy DSME, and the DSME with the CAP reduction start to make a difference due to the difference in the packet drop ratio (Refer to the result of Fig. 11).

In the simulation, when the number of nodes varies from 60 to 200, the average number of packets transmitted by a node decrease from 853.23 to 789.25 in TaCFPext, from 850.40 to 599.30 in the legacy DSME, and from 847.7 to 681.26 in the DSME with the CAP reduction, respectively. Therefore, TaCFPext exhibits a higher aggregate throughput than other cases because the nodes transmit a larger number of packets on average. Quantitatively, it achieves 16.20% and 6.03% higher aggregate throughput compared to the legacy DSME and the DSME with the CAP reduction, respectively.

Fig. 13 illustrates the GTS allocation delay for varying numbers of nodes in the static traffic scenario. In the simulation, the GTS allocation delay indicates the average time required for the node to complete the allocation for the required GTSs from the beginning of the simulation. The GTS allocation delay increases due to high contention in the CAP as the number of nodes increases. TaCFPext and the legacy DSME exhibit a much shorter GTS allocation delay than the DSME with the CAP reduction because multiple CAPs exist in one multi-superframe in TaCFPext and the legacy DSME, while in the DSME with the CAP reduction, one multi-superframe contains only one CAP.

Furthermore, compared to the legacy DSME, TaCFPext exhibits a slightly longer GTS allocation delay from the point

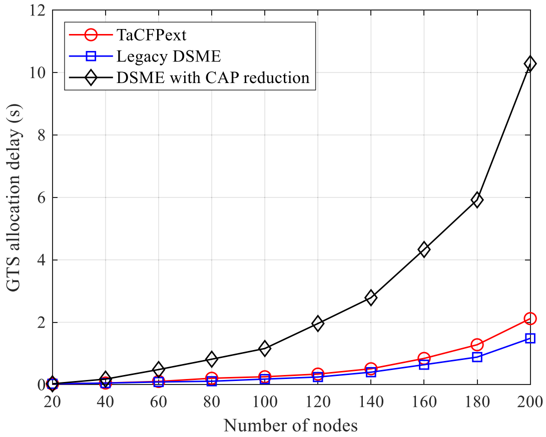

**FIGURE 13.** GTS allocation delay in static traffic scenario.

where the number of nodes is 80. When the number of nodes exceeds 60, some nodes in the legacy DSME fail to allocate GTSs due to insufficient space in the CFPs. In contrast, in TaCFPext, in this case, the CAP in the multi-superframe is changed to extCFP; the node can allocate extGTSs. Due to this additional allocation of extGTSs, TaCFPext exhibits a slightly longer GTS allocation delay than the legacy DSME.

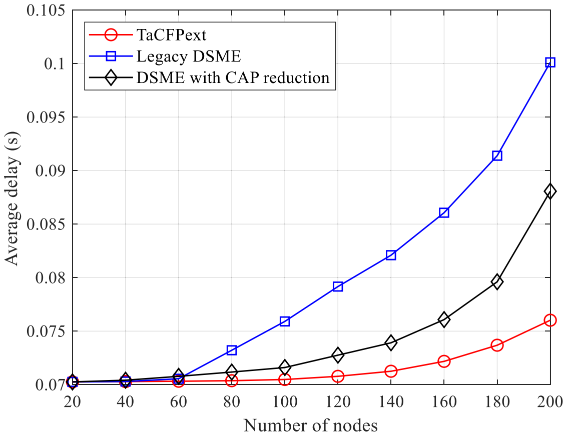

**FIGURE 14.** Average delay in static traffic scenario.

Fig. 14 illustrates the average delay for varying numbers of nodes in the static traffic scenario. In the simulation, the average delay indicates the average time required for a node to transmit a packet. The average delay of TaCFPext is shorter than those of the legacy DSME and the DSME with the CAP reduction. When the space in the CFPs is insufficient to allocate the required GTSs, TaCFPext enables the node to change the CAP to extCFP. However, in the legacy DSME, when the number of nodes exceeds 60, some nodes fail to allocate the required GTSs due to insufficient space in CFPs. The average number of packets transmitted within the multi-superframe decreases. Consequently, the average time each node transmits one packet increases sharply.

Furthermore, based on the results in Fig. 13, in TaCFPext, the nodes allocate the GTSs and transmit the packets earlier than the nodes in the DSME with the CAP reduction. Quantitatively, TaCFPext achieves 10.44% and 3.89% shorter average delay compared to the legacy DSME and the DSME with the CAP reduction, respectively.

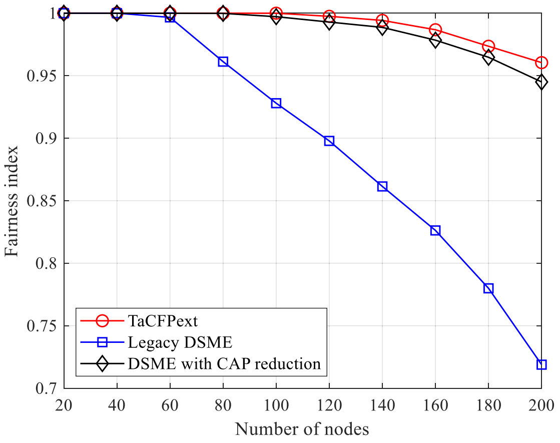

**FIGURE 15.** Fairness index in static traffic scenario.

Fig. 15 illustrates the fairness index for varying numbers of nodes in the static traffic scenario. The fairness index ($F$) can be calculated as follows [31]:

$$F = \frac{\left(\sum_{i=1}^{n} x_i\right)}{n \cdot \sum_{i=1}^{n} x_i^2} \tag{4}$$

where $n$ is the number of sender nodes, $i$ is the node ID of sender, and $x_i$ is the fairness parameter, indicating the number of packets that each sender node transmits.

In the simulation, a node drops the data packets as many as the number of GTSs it fails to allocate during a multi-superframe. Thus, from Eq. (4), the fairness index depends on whether each sender node has successfully allocated its GTSs. The fairness index decreases as the number of nodes increases. TaCFPext and the DSME with the CAP reduction maintain a high fairness index, which gradually decreases from when the number of nodes is more than 100. In contrast, when the number of nodes is 60, the fairness index of the legacy DSME rapidly decreases due to insufficient space in the CFPs.

Furthermore, when the number of nodes is more than 100, the fairness index of TaCFPext is slightly higher than that of the DSME with the CAP reduction. In the TaCFPext's multi-superframe, more CAPs are available than in the multi-superframe of the DSME with the CAP reduction. Accordingly, in TaCFPext, the number of nodes that succeed in GTS allocation during one multi-superframe is larger than in the DSME with the CAP reduction. Consequently, the difference in the number of transmitted packets between nodes is minimal.

Fig. 16 illustrates the number of allocated GTSs over time in the dynamic traffic scenario with 100 nodes. The upper,

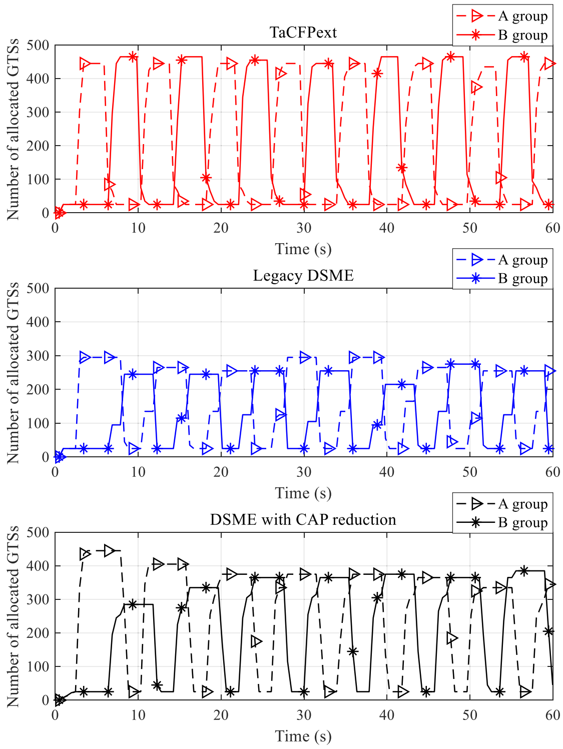

**FIGURE 16. Number of allocated GTSs in dynamic traffic scenario.**

middle, and lower figures in Fig. 16 illustrate the variations of the number of allocated GTSs in TaCFPext, the legacy DSME, and the DSME with the CAP reduction, respectively. In the dynamic traffic scenario, the node generates the high and low traffic loads alternately depending on the group it belongs to; thus, the number of allocated GTSs increases under the high traffic load and decreases under the low traffic load.

For TaCFPext (the upper figure), the intervals where the number of allocated GTSs increases and again decreases only slightly overlap between the two groups. In contrast, in the cases of the legacy DSME and the DSME with the CAP reduction (the middle and lower figures), these intervals overlap significantly. This difference between TaCFPext and the other cases is caused by the operational differences in extGTS deallocation and GTS deallocation. extGTSs are deallocated if they are not used during three multi-superframes, while GTSs are deallocated if they are not used during seven multi-superframes. Thus, the nodes in TaCFPext can deallocate their unnecessary GTSs earlier than the nodes in the other cases. Consequently, by deallocating the unnecessary GTSs earlier, TaCFPext ensures the larger space in CFPs, which is required for the next high traffic load.

In contrast, in the legacy DSME, the number of allocated GTS for one group increases and is maintained at a certain number under the high traffic load because the nodes no longer allocate GTSs due to insufficient space in CFPs. Then, the number of allocated GTSs increases again as the nodes belonging to another group deallocate their unnecessary GTSs.

In the DSME with the CAP reduction, the number of allocated GTSs increases by less than that of TaCFPext under the high traffic load. The multi-superframe of the DSME with the CAP reduction includes one CAP, whereas the multi-superframe of TaCFPext includes multiple CAPs. Thus, under the high traffic load, the nodes in the DSME with the CAP reduction wait more time than the nodes in TaCFPext when they fail to allocate GTSs.

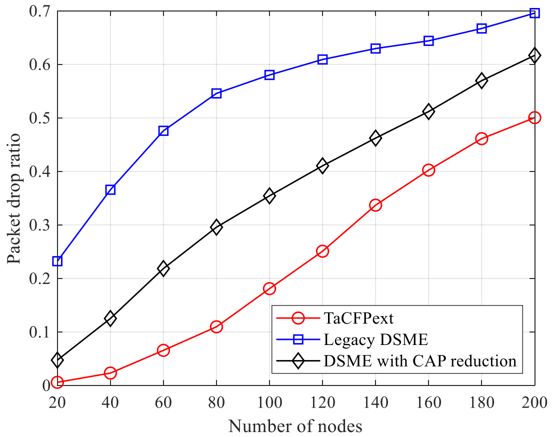

**FIGURE 17. Packet drop ratio in dynamic traffic scenario.**

Fig. 17 illustrates the packet drop ratio for varying numbers of nodes in the dynamic traffic scenario. The packet drop ratio increases as the number of nodes increases because the number of nodes that fail to allocate GTSs increases due to the high contention in CAPs and insufficient space in CFPs. In the dynamic traffic scenario, the packet drop ratio is higher than that of static traffic. In contrast to the static traffic scenario where one initial GTS allocation procedure is required, the nodes in the dynamic traffic scenario repeatedly perform GTS allocation and deallocation procedures whenever the traffic load changes. In the dynamic traffic scenario, the nodes that fail to allocate GTSs repeatedly occur.

TaCFPext exhibits a lower packet drop ratio compared to the legacy DSME and the DSME with the CAP reduction. In TaCFPext, the nodes use CAPs as extCFPs under the high traffic loads and ensure more space in CFPs for the next high traffic load by deallocating unnecessary GTSs earlier than the nodes in the other cases. Thus, TaCFPext minimizes the failures for GTS allocation caused by insufficient space in CFPs.

In contrast, in the legacy DSME, as the number of nodes increases, the number of nodes that fail to allocate GTSs increases sharply due to insufficient space in the CFPs. For the DSME with CAP reduction, the multi-superframe includes only one CAP. Therefore, even if it maintains a relatively larger space in CFPs, the number of nodes that fail to allocate GTSs is larger than that of TaCFPext due to a lack of CAPs.

Fig. 18 illustrates the aggregate throughput for varying numbers of nodes in the dynamic traffic scenario. The aggregate throughput increases due to the increased total amount

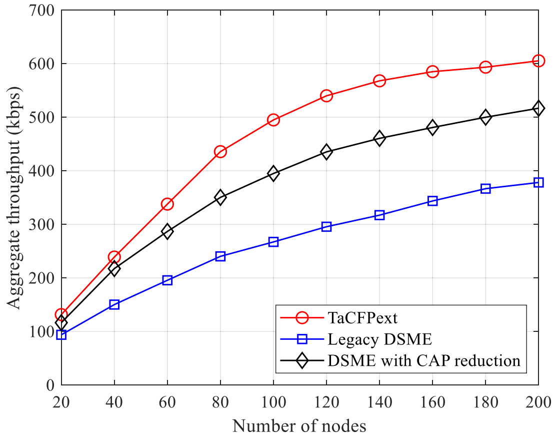

**FIGURE 18.** Aggregate throughput in dynamic traffic scenario.

of traffic loads as the number of nodes increases. TaCFPext, the legacy DSME and the DSME with the CAP reduction exhibit different aggregate throughputs, regardless of the number of nodes. In the dynamic traffic scenario, the number of dropped packets differs between TaCFPext, the legacy DSME, and the DSME with the CAP reduction even if there are a small number of nodes.

When the number of nodes increases from 20 to 120, the average number of packets that each node transmits decreases from 388.72 to 265.77 in TaCFPext, 277.42 to 130.74 in the legacy DSME, and 344.20 to 214.27 in the DSME with the CAP reduction. Consequently, TaCFPext exhibits a larger aggregate throughput than the other cases. Quantitatively, TaCFPext achieves 71.03% and 20.50% higher aggregate throughput compared to the legacy DSME and the DSME with the CAP reduction, respectively.

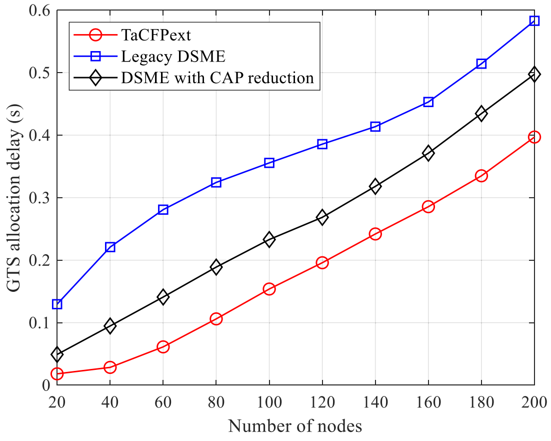

**FIGURE 19.** GTS allocation delay in dynamic traffic scenario.

Fig. 19 illustrates the GTS allocation delay for varying numbers of nodes in the dynamic traffic scenario. In the dynamic traffic scenario, the GTS allocation delay indicates the average time required for the node to complete the allocation for the required GTSs from the point where the demands for GTS allocation occur (the beginning of the simulation and the point where the low traffic load changes to the high traffic load). The GTS allocation delay increases as the number of nodes increases due to the high contention in CAPs.

TaCFPext exhibits a lower GTS allocation delay than those of the legacy DSME and the DSME with the CAP reduction. In TaCFPext, the nodes start to deallocate their GTSs earlier than the nodes in the other cases. In this case, the nodes belonging to one group complete the GTS deallocation before the nodes belonging to another group start to allocate their GTSs.

However, in the legacy DSME and the DSME with the CAP reduction, before the nodes belonging to one group allocate GTSs, the nodes belonging to another group may not complete the GTS deallocation. In this case, the nodes in both groups attempt the GTS allocation and deallocation simultaneously. Therefore, TaCFPext achieves the shorter GTS allocation delay due to the lower contention in CAPs compared to the legacy DSME and the DSME with the CAP reduction.

Furthermore, the legacy DSME exhibits a longer GTS allocation delay than the other cases. In the legacy DSME, due to insufficient space in CFPs, the nodes belonging to one group cannot allocate GTSs until the nodes belonging to another group finish deallocating the GTSs. Thus, in the legacy DSME, the GTS allocation delay increases significantly because the nodes wait for deallocating the GTSs.

The DSME with the CAP reduction exhibits a longer GTS allocation delay than TaCFPext due to the time that the node waits for retying the GTS allocation. In the DSME with the CAP reduction, the multi-superframe includes only one CAP; thus, if the nodes fail to allocate GTSs due to collisions in the CAP, the nodes are highly likely to wait for a longer time retrying the GTS allocation compared to the nodes using TaCFPext.

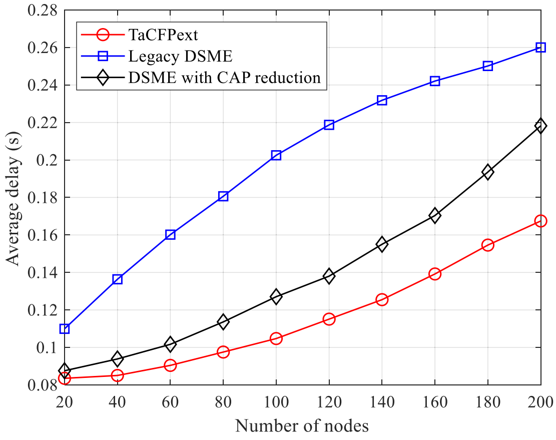

**FIGURE 20.** Average delay in dynamic traffic scenario.

Fig. 20 illustrates the average delay for varying numbers of nodes in the dynamic traffic scenario. As the number of nodes increases, the average delay increases due to the increase in the number of dropped packets. In contrast to the results in the static traffic scenario (Refer to Fig. 14), TaCFPext exhibits a shorter average delay compared to other cases even when the number of nodes is 60 or less. In the dynamic traffic scenario, the GTS allocation and deallocation are repeatedly performed due to the changes in traffic load. Therefore, a difference in the number of nodes failing to allocate the required GTSs occurs due to operational differences in GTS allocation and deallocation.

TaCFPext exhibits a shorter average delay compared to the legacy DSME and the DSME with the CAP reduction. Based on the results of Fig. 19, the nodes using TaCFPext allocate their GTSs and transmit the data packets earlier than the nodes in the other cases. Moreover, based on the results of Fig. 17, TaCFPext exhibits a lower packet drop ratio compared to other cases. Therefore, TaCFPext requires a shorter average time for each node to transmit one packet than in other cases. Quantitatively, TaCFPext achieves 41.61% and 16.85% shorter average delays compared to the legacy DSME and the DSME with the CAP reduction, respectively.

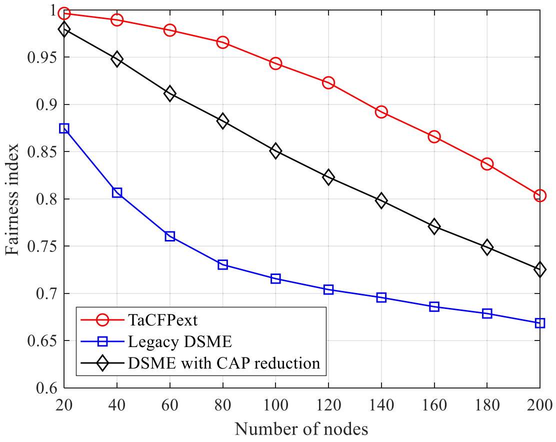

**FIGURE 21.** Fairness index in dynamic traffic scenario.

Fig. 21 illustrates the fairness index for varying numbers of nodes in the dynamic traffic scenario. In the dynamic traffic scenario, whenever a high traffic load occurs, the number of nodes that fail to allocate GTSs due to insufficient space in the CFPs increases. Accordingly, the fairness index in the dynamic traffic scenario is smaller than that in the static traffic scenario regardless of the number of nodes.

TaCFPext exhibits a higher fairness index than other cases. Based on the results of Figs. 16 and 17, TaCFPext maintains a high number of allocated GTSs even under high traffic load and a low packet drop ratio as the number of nodes increases, respectively. In TaCFPext, the number of nodes that fail to allocate GTSs due to insufficient space in the CFPs at a high traffic load is smaller than in other cases. Consequently, the difference in the number of transmitted packets between nodes is smaller than that of other cases.

However, in the legacy DSME, as the number of nodes increases, the number of nodes that fail to allocate GTSs due to insufficient space in CFPs at a high traffic load increases. Moreover, in the DSME with the CAP reduction, the nodes fail to allocate GTSs due to high contention in CAPs because the multi-superframe includes only one CAP. Quantitatively, TaCFPext achieves a 25.60% and 8.96% higher fairness index compared to the legacy DSME and the DSME with the CAP reduction, respectively.

## V. CONCLUSION

This paper presents a TaCFPext protocol for IEEE 802.15.4 DSME MAC that enables nodes to change the CAP in the multi-superframe to extCFP or vice versa in a distributed manner, according to the demands of traffic loads. The TaCFPext operation consists of two procedures: 1) changeable CAP selection and 2) extGTS allocation. In changeable CAP selection, a node verifies if it must use extCFP, and if so, selects which CAP in the multi-superframe to change to extCFP, considering the existing extGTSs of itself and its neighbors. In extGTS allocation, the node exchanges control commands to allocate extGTS. Then, if the extGTS allocation is successful, the node and its one-hop neighbors verify the channel and slot where the new extGTS is allocated and update their multi-superframe.

An experimental simulation was conducted under two traffic load conditions to verify the effectiveness of TaCFPext: static and dynamic traffic scenarios. TaCFPext performance was compared with that of the legacy DSME and the DSME with the CAP reduction. The simulation results demonstrated that TaCFPext improves network performance in terms of aggregate throughput and average delay when the traffic load changes. Quantitatively, in the static traffic scenario, TaCFPext exhibits 16.20% and 6.03% higher aggregate throughput, and 10.44% and 3.89% shorter average delay than the legacy DSME and the DSME with the CAP reduction, respectively. Furthermore, in the dynamic traffic scenario, TaCFPext achieves 71.03% and 20.50% higher aggregate throughput and 41.61% and 16.85% shorter average delay compared to the legacy DSME and the DSME with the CAP reduction, respectively.

## REFERENCES

[1] H. A. Salam and B. M. Khan, "IWSN–standards, challenges and future," *IEEE Potentials*, vol. 35, no. 2, pp. 9–16, Mar. 2016, doi: 10.1109/MPOT.2015.2422931.

[2] A. A. Kumar S., K. Ovsthus, and L. M. Kristensen., "An industrial perspective on wireless sensor networks—A survey of requirements, protocols, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1391–1412, 3rd Quart., 2014, doi: 10.1109/SURV.2014.012114.00058.

[3] M. Gidlund, S. Han, E. Sisinni, A. Saifullah, and U. Jennehag, "Guest editorial from industrial wireless sensor networks to industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2194–2198, May 2018, doi: 10.1109/TII.2018.2815957.

[4] J.-H. Kwon, H.-H. Lee, Y. Lim, and E.-J. Kim, "Dominant channel occupancy for Wi-Fi backscatter uplink in industrial Internet of Things," *Appl. Sci.*, vol. 6, no. 12, p. 427, Dec. 2016, doi: 10.3390/app6120427.

[5] *IEEE Standard for Low-Rate Wireless Networks*, IEEE Standard 802.15.4-2015, Apr. 2016.

[6] *Industrial Communication Networks-Wireless Communication Networkand Communication Profiles-WirelessHART*, IEC Standard 62591, 2016.

[7] *Wireless Systems for Industrial Automation: Process Control and Related Applications*, ISA Standard 100.11 a-2009, 2009.

[8] S. S. Oyewobi and G. P. Hancke, "A survey of cognitive radio handoff schemes, challenges and issues for industrial wireless sensor networks (CR-IWSN)," *J. Netw. Comput. Appl.*, vol. 97, pp. 140–156, Nov. 2017, doi: 10.1016/j.jnca.2017.08.016.

[9] H. Kurunathan, R. Severino, A. Koubaa, and E. Tovar, "IEEE 802.15.4e in a nutshell: Survey and performance evaluation," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1989–2010, 3rd Quart., 2018, doi: 10.1109/COMST.2018.2800898.

[10] D. De Guglielmo, S. Brienza, and G. Anastasi, "IEEE 802.15.4e: A survey," *Comput. Commun.*, vol. 88, pp. 1–24, Aug. 2016, doi: 10.1016/j.comcom.2016.05.004.

[11] I. Juc, O. Alphand, R. Guizzetti, M. Favre, and A. Duda, "Energy consumption and performance of IEEE 802.15.4e TSCH and DSME," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 2016, pp. 1–7, doi: 10.1109/WCNC.2016.7565006.

[12] P. Sahoo, S. Pattanaik, and S.-L. Wu, "A reliable data transmission model for IEEE 802.15.4e enabled wireless sensor network under WiFi interference," *Sensors*, vol. 17, no. 6, p. 1320, Jun. 2017, doi: 10.3390/s17061320.

[13] G. Alderisi, G. Patti, O. Mirabella, and L. L. Bello, "Simulative assessments of the IEEE 802.15.4e DSME and TSCH in realistic process automation scenarios," in *Proc. IEEE 13th Int. Conf. Ind. Informat. (INDIN)*, Jul. 2015, pp. 948–955, doi: 10.1109/INDIN.2015.7281863.

[14] N. Choudhury, R. Matam, M. Mukherjee, and J. Lloret, "A performance-to-cost analysis of IEEE 802.15.4 MAC with 802.15.4e MAC modes," *IEEE Access*, vol. 8, pp. 41936–41950, 2020, doi: 10.1109/ACCESS.2020.2976654.

[15] D. G. De Guglielmo Anastasi and A. Seghetti, "From IEEE 802.15.4 to IEEE 802.15.4e: A step towards the Internet of Things," in *Proc. Adv. Onto Internet Things*, Jan. 2014, pp. 135–152, doi: 10.1007/978-3-319-03992-3_10.

[16] C. Vallati, S. Brienza, M. Palmieri, and G. Anastasi, "Improving network formation in IEEE 802.15.4e DSME," *Comput. Commun.*, vol. 114, pp. 1–9, Dec. 2017, doi: 10.1016/j.comcom.2017.09.016.

[17] S. Capone, R. Brama, F. Ricciato, G. Boggia, and A. Malvasi, "Modeling and simulation of energy efficient enhancements for IEEE 802.15.4e DSME," in *Proc. Wireless Telecommun. Symp.*, Apr. 2014, pp. 1–6, doi: 10.1109/WTS.2014.6835017.

[18] F. Meyer, I. Mantilla-González, and V. Turau, "Sending multiple packets per guaranteed time slot in IEEE 802.15.4 DSME: Analysis and evaluation," *Internet Technol. Lett.*, vol. e167, pp. 1–6, Jun. 2020, doi: 10.1002/itl2.167.

[19] M. Taneja, "A framework to support real-time applications over IEEE802.15.4 DSME," in *Proc. IEEE 10th Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process. (ISSNIP)*, Apr. 2015, pp. 7–9, doi: 10.1109/ISSNIP.2015.7106918.

[20] J. Lee and W.-C. Jeong, "Performance analysis of IEEE 802.15.4e DSME MAC protocol under WLAN interference," in *Proc. Int. Conf. ICT Converg. (ICTC)*, Oct. 2012, pp. 741–746, doi: 10.1109/ICTC.2012.6387133.

[21] Q. Wang, X. Vilajosana, and T. Watteyne, *6TiSCH Operation Sublayer (6top) Protocol (6P)*, document RFC8480, IETF, Nov. 2018.

[22] M. R. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, L. A. Grieco, and T. Engel, "On-the-fly bandwidth reservation for 6TiSCH wireless industrial networks," *IEEE Sensors J.*, vol. 16, no. 2, pp. 550–560, Jan. 2016, doi: 10.1109/JSEN.2015.2480886.

[23] R. T. Hermeto, A. Gallais, and F. Theoleyre, "Scheduling for IEEE802.15.4-TSCH and slow channel hopping MAC in low power industrial wireless networks: A survey," *Comput. Commun.*, vol. 114, pp. 84–105, Dec. 2017, doi: 10.1016/j.comcom.2017.10.004.

[24] X. Vilajosana, T. Watteyne, T. Chang, M. Vucinic, S. Duquennoy, and P. Thubert, "IETF 6TiSCH: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 595–615, 1st Quart., 2020, doi: 10.1109/COMST.2019.2939407.

[25] H. Farag, S. Grimaldi, M. Gidlund, and P. Osterberg, "REA-6TiSCH: Reliable emergency-aware communication scheme for 6TiSCH networks," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1871–1882, Feb. 2021, doi: 10.1109/JIOT.2020.3016643.

[26] F. Kauer, M. Köstler, and V. Turau, "Reliable wireless multi-hop networks with decentralized slot management: An analysis of IEEE 802.15.4 DSME," Jun. 2018, *arXiv:1806.10521*. [Online]. Available: http://arxiv.org/abs/1806.10521

[27] Y.-S. Lee and S.-H. Chung, "An efficient distributed scheduling algorithm for mobility support in IEEE 802.15. 4e DSME-based industrial wireless sensor networks," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 2, pp. 1–14, Jan. 2016, doi: 10.1155/2016/9837625.

[28] F. Battaglia, M. Collotta, L. Leonardi, L. Lo Bello, and G. Patti, "Novel extensions to enhance scalability and reliability of the IEEE 802.15.4-DSME protocol," *Electronics*, vol. 9, no. 1, p. 126, Jan. 2020, doi: 10.3390/electronics9010126.

[29] P. Sahoo, S. Pattanaik, and S.-L. Wu, "A novel IEEE 802.15.4e DSME MAC for wireless sensor networks," *Sensors*, vol. 17, no. 12, p. 168, Jan. 2017, doi: 10.3390/s17010168.

[30] H. Kurunathan, R. Severino, A. Koubaa, and E. Tovar, "DynaMO—Dynamic multisuperframe tuning for adaptive IEEE 802.15.4e DSME networks," *IEEE Access*, vol. 7, pp. 122522–122535, 2019, doi: 10.1109/ACCESS.2019.2937952.

[31] A. V. Babu and L. Jacob, "Fairness analysis of IEEE 802.11 multirate wireless LANs," *IEEE Trans. Veh. Technol.*, vol. 56, no. 5, pp. 3073–3088, Sep. 2007, doi: 10.1109/TVT.2007.898397.

**SANG-WOO LEE** (Graduate Student Member, IEEE) received the B.S. degree in convergence software from Hallym University, Chuncheon, South Korea, in 2019. He is currently pursuing the integrated M.S. and Ph.D. degree in convergence software. His current research interests include design, analysis, and optimization of wireless communication systems, and the Internet of Things.

**JUNG-HYOK KWON** (Member, IEEE) received the B.S. degree in electronic engineering from Soongsil University, Seoul, South Korea, in 2010, the M.S. degree in electronics and computer engineering from Korea University, Seoul, in 2012, and the Ph.D. degree in convergence software from Hallym University, Chuncheon, South Korea, in 2019. From September 2012 to March 2013, he was with the Electrical and Electronic Engineering Research Institute (EERI), Korea University, Seoul. From April 2013 to June 2015, he was with the Software Research and Development Laboratory, LIG Nex1 Company Ltd., Seongnam, South Korea. Since March 2019, he has been a Research Professor with the Smart Computing Laboratory, Hallym University. His research interests include design and performance analysis of medium access control protocols for next-generation communication systems, vehicular communications (V2X), machine-to-machine communications, machine learning, wireless powered communication, and the Internet of Things. He was selected as a recipient of the Global Ph.D. Fellowship Program sponsored by the National Research Foundation of Korea, in 2016.

**XUE ZHANG** received the B.S., M.S., and Ph.D. degrees in electronic engineering from Hallym University, Chuncheon, South Korea, in 2013, 2015, and 2018, respectively. From January 2019 to August 2019, she worked with the School of Mechanical and Electronic Engineering, Shandong University of Science and Technology, Qingdao, China, where she has been a Lecturer with the School of Ocean Science and Engineering, since September 2019. Her current research interests include ocean sensors and semiconductor devices.

**EUI-JIK KIM** (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electronics and computer engineering from Korea University, Seoul, South Korea, in 2004, 2006, and 2013, respectively. From September 2005 to December 2005, he was with the Intel Korea Research and Development Center, Intel Corporation, Seoul. From February 2006 to July 2009, he was with the DMC Research and Development Center, Samsung Electronics Company Ltd., Suwon, South Korea. From August 2009 to August 2013, he was with the Advanced Institute of Technology, KT Corporation, Seoul. Since September 2013, he has been an Associate Professor with the School of Software, Hallym University, Chuncheon, South Korea. His current research interests include wireless/mobile networks with an emphasis on QoS guarantee and adaptation, wireless sensor networks, energy harvesting, wireless powered communication networks, and the Internet of Things.