Deterministic Transmittable Time-based Asynchronous Scheduler for Fronthaul Networks

Shiyan Zhang* **Xiangyang Gong*** **Puye Wang*** **Long Wang*** **Tongtong Wang†** **Xirong Que*** **Ye Tian***

*State Key Laboratory of Networking and Switching Technology
Institute of Network Technology
Beijing University of Posts and Telecommunications, Beijing, China
{zshiyan, xygong, magicwpy, wanglong, rongqx, yetian}@bupt.edu.cn

†Huawei Technologies Co. Ltd
tongtong.wang@huawei.com

Abstract—**Time-sensitive Networking (TSN) has been extensively utilized to enable the transport of time-sensitive fronthaul flows in Ethernet-bridged networks. However, recently emerged ultra-low delay and jitter requirements of the TSN pose great challenges to Common Public Radio Interface (CPRI) over Ethernet. Considering this, we propose a Deterministic Transmittable Time-Based Asynchronous Scheduler (DTT-BAS) in this paper. The DTT-BAS uses the deterministic transmittable timestamp as the core scheduling mechanism, which numerically compares the timestamps to optimize the flow scheduling in the network. For the fronthaul scenario, simulation results validate that the DTT-BAS achieves a high probability of zero jitter and reduces the influence of clock skews with a delay performance close to IEEE 802.1CM frame preemption algorithm.**

Keywords—**Fronthaul, Asynchronous Scheduler, CPRI over Ethernet, Time-sensitive Networking.**

I. Introduction

Cloud Radio Access Networks (C-RANs) [1] have recently become the main architecture in the newest generation of fronthaul networks [2]. With Remote Radio Head (RRH) and Bandwidth Based Unit (BBU), which are able to support long-distance transmission, the fronthaul has expanded to the range of 10km [3]. Furthermore, an increasing number of RRHs are connected to the same virtualized BBU pool, escalating the transmission demand of the fronthaul. In this context, the 802.1CM task group is studying Time-sensitive Networking (TSN)-based fronthaul [4], which is proposed to deploy modified Ethernet equipment to the alleviate ongoing cost and maintenance constraints of the fronthaul networks.

The Ethernet-based fronthaul is an integral part of 5G, and the standard TSN profile for fronthaul is compatible with the CPRI/eCPRI protocol. The in-phase and quadrature (IQ) samples of the baseband signal are carried as high priority traffic class through the CPRI streams [5], [6]. Bandwidth is not the only limiting factor due to the exponential traffic growth and transmission speed of 5G networks.The new time-sensitive fronthaul interface will require a stable timing reference, as well as strict delay and jitter budgets for the aggregation and exchange of long-distance transmissions, combined with the wide application and easy deployment of Ethernet.

In the fronthaul networks, multiple Radio Equipments (REs) are connected to one Radio Equipment Control (REC) allowing streams that are originating from different REs to converge at a switch on the uplink [7]. Furthermore, a switching device that adopts a token-based round-robin switching mechanism may select an unspecified amount of frames from each direction for each exchange motive; thus, the output order of the devices may differ from one exchange order to another. This issue called fan-in phenomenon [8] may cause uncertainty in transmission and result in additional delays and jitter.

The transmission algorithms and the equipment used in conventional Ethernet, no longer meet the requirements of the fronthaul [2]. Synchronization-based scheduling algorithms, such as time-aware shaper in 802.1Qbv, can theoretically provide stable network transmission [7], [9]. However, their applicability to the fronthaul is still questionable as they are prone to be affected by clock synchronization between the devices. When the clock synchronization error is high, the performance of these algorithms deteriorates as expected.

Even though the strict priority or frame preemption proposed by the 802.1CM working group can enforce the transmission of high-priority flows, these mechanisms still cannot address the unpredictable blocking caused by the low-priority flows [10], [11]. Therefore, the devices fail in eliminating all of the unpredictable blocking effects, which increases the jitter and results in transmission delay causing more problems later on.

In this paper, we propose a Deterministic Transmittable Time-Based Asynchronous Scheduler (DTT-BAS) for TSN-based fronthaul networks. This scheduling scheme is based on a deterministic transmittable timestamp mechanism that is key to provide a deterministic method of scheduling. It applies a queue selection mechanism and a specific state maintenance algorithm to differentiate the source and the flow types from each other. The main contributions of this article are as follows: 1) our scheduling scheme ensures a stable and efficient transmission for high-priority traffic in the network; 2) this scheme solves the problem of jitter caused by indeterminate flow exchange in the existing synchronous/asynchronous scheduling algorithms; 3) this scheme reduces the impact of clock synchronization errors between the devices.

The remainder of this paper is organized as follows. First, we start with summarizing the related existing studies in Section II. Then, we give the DTT-BAS architecture and detail our proposed algorithms in Section III. This is followed by the DTT-BAS simulation and the experimental comparison in

Section IV. Afterwards, the article is concluded.

II. RELATED WORKS

Our work is simply based on TSN standards and scheduling algorithms. As stated, the IEEE 802.1CM task group is studying the related content to provide stable, effective and low-cost transmission methods for the fronthaul networks [4]. The group is seeking to introduce the TSN-based standards into the fronthaul networks to alleviate the ongoing transmission-based problems. TSN for fronthaul therefore aims at resolving the issues involved with CPRI over Ethernet.

The scheduling algorithms applied to the TSN can be categorized into two different classes: synchronous and asynchronous. The synchronization-based scheduling algorithms typically include 802.1Qbv [9] and 802.1Qch [12]. However, their performance is severely affected by the accuracy of the clock synchronization, which is hard to control. Due to clock synchronization errors, the synchronization-based algorithms may possibly cause great fluctuations in scheduling as they miss the appropriate transfer window [13]. The scheduling mode based on Time-Division Multiplexing (TDM) slot allocation however, can effectively reduce the transmission jitter. The authors of [14] therefore proposed a time slot allocation algorithm called Comb-fitting (C-Fit), which is a low-jitter fronthaul network scheduling method guaranteeing low delay by settling the conflict on time slot allocation.

In asynchronous scheduling algorithms, frame preemption is used as a common mechanism for the TSN. However, the frame preemption mechanism is not able to prevent the influence of the background flow [15]. The uncertainty in transmission introduced by the frame preemption mechanism thus affects the scheduling of subsequent network transmission and eventually causes to greater jitter.

For all existing asynchronous scheduling algorithms developed specific to the TSN, Urgency-Based Scheduler (UBS), considered as a solution in the 802.1Qcr draft, provides a bounded delay for high link utilization through a specialized timestamp mechanism and a directional classification mechanism [16], [17]. In [18], for multi-stream aggregation problem of the real-time Ethernet deterministic flows with the UBS, a cluster-based heuristic algorithm is proposed to ease the allocation of flow-to-queue and queue-to-priority level assignments through a Satisfiability Modulo Theories solver. The UBS, in accordance with the flow source classification approach, can separate the network flows from all individual directions in a peer-to-peer network, and isolate the flow of each influence resolving the fan-in problem. However, the time stamping mechanism in the UBS affects the subsequent transmission under unstable situations, which makes it non-ideal for high priority flow protection. In other terms, the UBS is useless for the fronthaul.

To overcome the above-mentioned problems in the practical implementation of the TSN-based fronthaul, we propose the DTT-BAS which consists of a deterministic scheduling mechanism employed an optimized clock stamp for both reducing the impact of uncertainty and ensuring stable transmission for the high-priority traffic throughout the network.

III. DETERMINISTIC TRANSMITTABLE TIME-BASED ASYNCHRONOUS SCHEDULER

After analyzing the technical requirements of the 802.1CM standard, we designed a totally new architecture for a switching device, and proposed an asynchronous resolution which is based on a deterministic transmittable timestamp. This is to improve the transmission stabilization and resolve the issues on the asynchronous scheduler in the TSN-based fronthaul. The DTT-BAS also reduces the impact of uncertainties in the Ethernet-based fronthaul.

A. Definitions & Initialization Settings

In the network, the switching device, depicted in Fig. 1, has $Port\_num$ ports P_i, where each port is a full-duplex Ethernet port, and the physical link rate of the port connection is S_{line}. Here, the ingress port $p_{(i,in)}$ and the egress port $p_{(i,out)}$ work independently. The frame that enters from the $p_{(i,in)}$ is denoted as $frame_{(in)}$, and the frames are exchanged by the switch fabric of the switching device in store-and-forward mode.

For the egress port $p_{(i,out)}$ of the switching device, it is necessary to provide a queue set Q with at least $Port\_num$ q, i.e., $q_i \in Q$. The q in P_i, is denoted as q_i, which is also divided into Q^h and Q^l. $q_0 \sim q_{portnum-2} \in Q^h$ is used to serve the IQ-data, where the remaining one, $q_{(portnum-1)}$, serves all the BG streams, i.e., $q_{(portnum-1)} \in Q^l$. The flow reservation information, i.e., the Reservation Info, obtains the rate information from the SRP (802.1Qat) protocol, and combines the source port mapping information of the flow.

We define a flow f with rate s, where the frame f_i is denoted as $frame_i$. f_{in} represents the flow originating from $p_{(i,in)}$ at rate s_{in}. Here, the Reservation Info requires some basic knowledge to complete the initial set-up. It stores the corresponding number of recovery speed information according to the total number of ports on the device. The timestamp recovery rate uses a floating point-type tag to match the state recovery speed of the queue in bits/sec, denoted as r_i. $r_0 \sim r_{portnum-2}$ is used for Q^h, and $r_{portnum-1}$ is used for Q^l, which can also be denoted as r_{BG}.

As stated, the Reservation Info obtains the rate information of the flow reservation from the SRP (802.1Qat) protocol, and sets the corresponding r_i according to the source port mapping information of the flow. Then, r_{BG} is set either manually or according to the remaining available resources of the link.

For the each queue of the IQ-data, the control rate r_i is

$$r_i = \sum_{in \in P} S_{j,in},$$

and the maximum control rate r_{BG} for the BG is

$$r_{BG} = r_{portnum-1} = S_{line} - \sum_{i \in P} r_i.$$

This item can also be manually set to be less than the amount of r_{BG}.

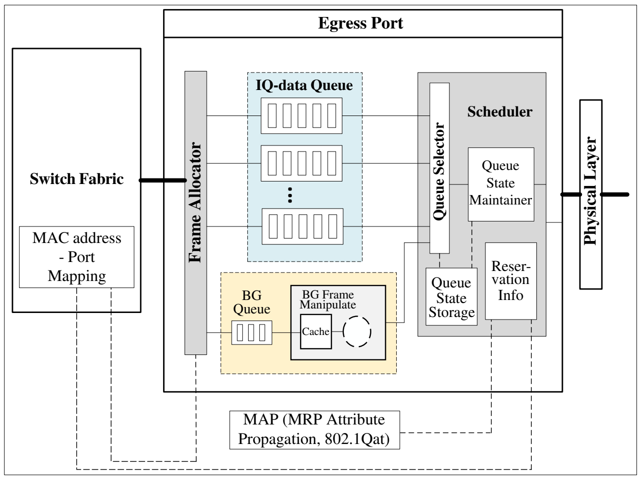


Fig. 1: DTT-BAS architecture.

B. *DTT-BAS Model Structure*

To better meet the requirements of the Ethernet-based fronthaul, the DTT-BAS scheduling algorithm model is proposed. Fig. 1 shows the architecture of a basic DTT-BAS device.

The Frame Scheduler is built upon the Transmittable Time, which is based on the timestamp mechanism. Since the Queue Selector uses the Transmittable Time to mark the transmittable state of the each queue, the Transmittable Time is maintained and updated by the Queue State Maintainer depending on the stream reservation and queue status information. The Queue State is updated and stored by the Queue State Storage.

Since the frame size and the frame interval of the BG stream are uncertain, our goal is to shape the BG stream into a Common Bit Rate (CBR)-compatible stream to minimize the interference to the IQ-data stream occurred due to BG stream. The low-speed frame processing module cooperates with the Transmittable Time mechanism through the active occupancy function of the device link which allows the low-speed traffic that is aggregated into the BG queue and provides a stable output effect of the CBR compatible mode.

C. *Algorithm in the DTT-BAS System*

The previous section describes the overall architecture of the DTT-BAS system elaborating on the modules that the system is built upon. This section overviews the algorithms applied to each module within the DTT-BAS structure.

Algorithm 1 Frame Allocate Algorithm

```
Input: frame
1: if $frame.EtherType == 0x8100$ and $frame.PCP == 7$ then
2:     $in = FindIngressWithMacAddress(frame.src)$
3:     $Calculate\ queue\ number\ i\ by\ in$
4:     $queue_i.enqueue(frame)$
5: else
6:     $queue_{BG}.enqueue(frame)$
7: end if
```

Frame Allocate Algorithm

The Frame Classification Algorithm is applied to the frame classifier to isolate the frames from different categories and directions. The pseudo-code of the scheduling algorithm for the frame classifier is shown in Algorithm 1.

When the frame enters the frame classifier, the Priority Code Point (PCP) field of the frame is first identified. If the field value is 7, then the frame is an IQ-data frame; otherwise, it is regarded as a BG frame by default. For the IQ-data class frames, the frame classifier further passes the Media Access Control address of the frame upon its entry into the device. Next, according to the obtained port number, the calculation of the queue number is initialized.

Queue Manage Algorithm

The Queue Manage Algorithm, which is mainly employed in the Frame Scheduler module, can be divided into two parts: Queue Select Algorithm and Queue State Update Algorithm. These algorithms are deployed on the Queue Select Module and the Queue State Maintainer Module of the frame scheduler, respectively.

1) Queue Select Algorithm: The Queue Select Algorithm, which is operated on the Queue Select Module, is used to select one effective queue from the entire set of queues. There is a slight difference between the transmission of IQ-data and BG queues due to the specialized operations. The pseudo-code of the Queue Select Algorithm is shown in Algorithm 2.

When the state of the egress port is set to the IDLE, the queue meeting $txMachineState$ condition is selected from high-priority IQ-data and low-priority BG queues, then the d-equeuing operation is performed. For the IQ-data queues, if the queue is non-null, then it is directly sent traversing all queues. The queue with the smallest timestamp is selected in the valid status queue. After the selection of the following algorithm, the selected queue performs the dequeue operation, and is directly transmitted. For the BG queue, once it is selected for transmission, regardless of whether a frame transmission exists and the frame size satisfies the condition, the low-speed frame processing module will occupy the sending device throughout this period. If the selected IQ-data queue is not empty, then it is directly transmitted.

Algorithm 2 Queue Select Algorithm

```
1: while $TRUE$ do
2:     $wait\ until\ txMachineState = IDLE$
3:     for each $queue \in Q$ do
4:         if $queue\ is\ not\ empty$ then
5:             $m = transmittableTime_m < transmittableTime_{queue}$
               $?m : queue$
6:         end if
7:     end for
8:     if $transmittableTime_m \leq t_{now}$ then
9:         if $queue_m \in Q_l$ then
10:            $StartTransmitBG()$
11:        else
12:            if $queue_m\ is\ not\ empty$ then
13:                $frame = queue_m.dequeue()$
14:                $StartTransmit(frame)$
15:            end if
16:        end if
17:    end if
18: end while
```

2) Queue State Update Algorithm: The Queue State Update Algorithm, which is performed by the Queue State Maintainer

Module in the system, calculates the number and update the status of each queue based on the recovery information stored in the Reservation Info. The value being updated is stored in the Queue State Storage Module as seen in Algorithm 3.

Algorithm 3 Queue State Update Algorithm

```
Input: $frame_{in}$
1: if $frame_{in}$ is $frame_{BG}$ then
2:     if $transmittableTime_{BG} == 0$ then
3:         $transmittableTime_{BG} = t_{now} + pktLimitSize_{BG}/r_{BG}$
4:     else
5:         $transmittableTime_{BG} += pktLimitSize_{BG}/r_{BG}$
6:     end if
7: else
8:     if $transmittableTime_{in} == 0$ then
9:         $transmittableTime_{in} = t_{now} + frame_{in}.size/r_{in}$
10:    else
11:        $transmittableTime_{in} += frame_{in}.size/r_{in}$
12:    end if
13: end if
```

The initial value of the timestamp is determined when all frames arrive for the first time, where each subsequent update is added on the previous basis. For the BG frames, the BG queue is updated with a fixed value each time, i.e., it is based on sending one fixed-size frame at a time and updating the status. For the IQ-data frames, the corresponding timestamp is updated according to the recorded timestamp recovery rate.

BG Frame Manipulate Algorithm

The BG Frame Manipulate Algorithm is employed on the low-speed frame processing module. It performs a series of operations based on the size of both the processed frames and the sending window, including frame dequeuing, segmentation and buffering. The pseudo-code is shown in Algorithm 4.

When the queue selection module selects the BG queue, the BG frame processing module continues to occupy the device. Then, the size of the BG flow send window $RemainBytes$ is set to a specified BG fixed window size $pktLimitSize_{BG}$. If $frame.size$ is less than or equal to $RemainBytes$, then the frame can be directly transmitted. In addition, $RemainBytes$ must be updated, and the space represented by the $RemainBytes$ should be considered for the other frames. The Interval Frame Gap (IFG) overhead is also regarded when calculating the remaining transmittable resources. If $frame.size$ is greater than $RemainBytes$, then the frame must be cut meeting the IEEE 802.3 regulations. These regulations need to be followed during both the cut and the transmission. After the conditions are met for the cut, the frame is divided into left and right slices. While the left one is transmitted, the right slice is returned to the cache at the same time. The $RemainBytes$ characterizes the remaining space in the sending window.

IV. SIMULATION

In this section, we simulate a fronthaul network in NS-3, based on the example topology provided by the draft file of 802.1CM. The most problematic part in the fronthaul is the up-link, as it has a higher chance of conflict and much more bandwidth usage. Thus, we have designed a topology to simulate the convergence and dispersion of more streams.

Algorithm 4 BG Frame Manipulate Algorithm

```
1: wait until $queue_{BG}$ being selected
2: $RemainBytes = pktLimitSize_{BG}$
3: $txMachineState = BUSY$
4: while $RemainBytes > 0$ do
5:     if cache is empty then
6:         $cache.input(queue_{BG}.dequeue())$
7:     end if
8:     $frame = cache.get\_frame()$
9:     if $frame.size \leq RemainBytes$ then
10:        $RemainBytes -= (frame.size + IFG)$
11:        $StartTransmit(frame)$
12:        continue
13:    end if
14:    if $RemainBytes \leq 64Bytes$ or $frame.size < 124Bytes$ then
15:        break
16:    else
17:        if $frame.size - RemainBytes \geq 64Bytes$ then
18:            $left = frame.cut(RemainBytes)$
19:            $right = frame.remain(RemainBytes)$
20:        else
21:            $left = frame.cut(frame.size - 64)$
22:            $right = frame.remain(frame.size - 64)$
23:        end if
24:        $cache.input(right)$
25:        $RemainBytes -= (left.size + IFG)$
26:        $StartTransmit(left)$
27:    end if
28: end while
29: wait$(RemainBytes + IFG)/S_{line}$ seconds
30: set $txMachineState = IDLE$
```

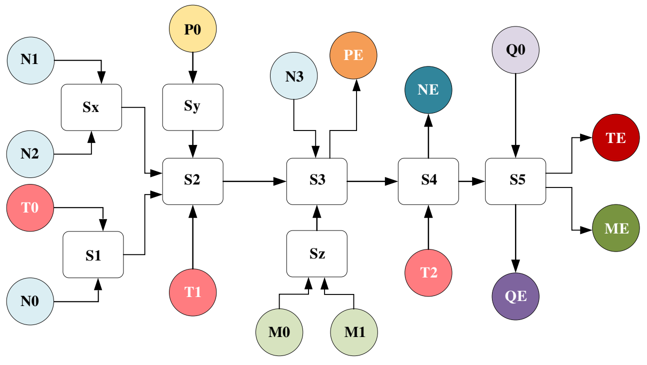


Fig. 2: Simulation topology.

The circles in Fig. 2 designates the nodes in the topology that either generates or receives the flows, which have installed the DTT-BAS, Frame Preemption and C-FIT algorithms. All the other nodes depicted as rectangles are switches supporting the DTT-BAS, Frame Preemption and C-FIT algorithms. The overheads, such as switch and transmit delays, are not considered.

The setting of all flows running in the topology is shown in Table I. There are two types of flows running in the network: CBR-type IQ-data flow and Variable Bit Rate-type BG flow. Each sending node generates two types of streams in the same direction. The link between each node is a 10Gbps full-duplex supported fronthaul. The IQ-data flow is 1Gbps, the frame sizes are 1500bytes, and the transmission interval is 12000ns.

The simulations are run under two scenarios by taking the results of IQ-data analysis as the basis. The first scenario is run without the clock skews, where the second one however, uses the clock skews in $\pm$ 16ppm [4]. Note that, both scenarios have the same topology and flow settings.

TABLE I: Information on the flows.

Flow	**SRC**	**DST**	***Flow***	**SRC**	**DST**
1	N0->	NE	*7*	T0->	TE
2	N1->		*8*	T1->	
3	N2->		*9*	T2->	
4	N3->		*10*	M0->	ME
5	P0->	PE	*11*	M1->	
6	Q0->	QE			

A. Simulation Results with Non-Clock Skews

Fig. 3 depicts the maximum delay for the first scenario, i.e., the maximum end-to-end (e2e) delay of all IQ-data flows in the network. Compared to the C-FIT, the DTT-BAS has similar maximum e2e delay of the transmission with frame preemption. The DTT-BAS usually requires more resources and might get blocked longer at one hop, but it is more advantageous in decentralized network environment with more stream aggregation. The DTT-BAS shows that some transmissions perform better than the frame preemption due to the unpredictable preempt motion, which may block a large amount of IQ-data. However, the C-FIT, which guarantees low jitter, leads to longer delays.

Fig. 4 shows the jitter of each flow for the first scenario. It is seen that both DTT-BAS and C-FIT can reach zero jitter under the non-clock skews situation. The figure also shows that the frame preemption is more unstable, and the jitter of each flow is different because of the unstable transmission and unpredictable schedule motion. Furthermore, from both Fig. 3 and Fig. 4, we observe that the DTT-BAS not only provides relatively low latency but also achieves lower jitter.

As the previous discussion focuses on the comparison of delay and jitter performances at the same flow rate, the following section considers the overall performance of the algorithms. The maximum link utilization is defined as the link with the highest traffic aggregation in the simulation topology. Following this, more than 2000 random combinations of input rates are generated, and random IQ-data flow rates are derived from 0.5Gbps to 2Gbps. Fig. 5 illustrates the jitter value versus increasing maximum link utilization for the DTT-BAS

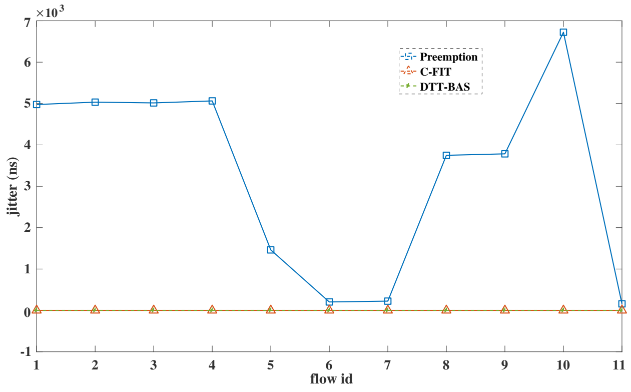


Fig. 4: Jitter with non-clock skews.

compared to C-FIT and frame preemption . It can be seen that the sum of jitter for the DTT-BAS remains lower and more concentrated than the C-FIT, which is affected by inter-stream interference and coordination between the schedules when the rates of IQ-data streams are different.

B. Simulation Results with Clock Skews

Since the frame preemption algorithm has no relation to clock or time of the device, all results of the frame preemption are maintained in the similar state for both scenes. However, the C-FIT uses a standard time algorithm for the control, where the DTT-BAS is based on a timestamp mechanism.Both algorithms are subject to clock skews. The results in Fig. 6 are for the upstream switching device Sx and S1 with varying clock skews, while other devices are configured randomly. Average of all flows is selected for the observation by s-tarting the send time from 0sec to 3sec. By comparing the statistical distribution of the frames, we find that all clock skew becomes larger, the delay and jitter become larger, and a clock calibration is required every 1 second.we find that as the clock skew becomes larger, the delay and jitter become larger,and all clock calibrations need to be performed every 1 second. we conclude that the DTT-BAS is less affected by the clock skews. In other words, the clock skew has a more negative effect on the delay of the C-FIT, because the clock offset and its cumulative impact lead to a large number of

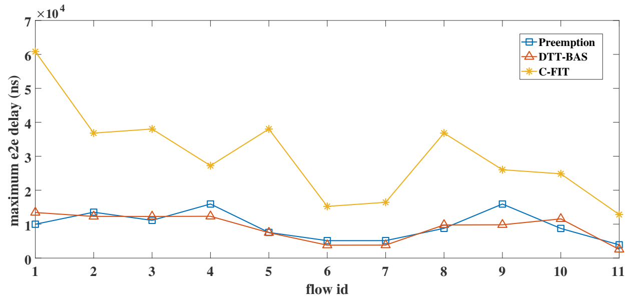


Fig. 3: Maximum e2e delay with non-clock skews.

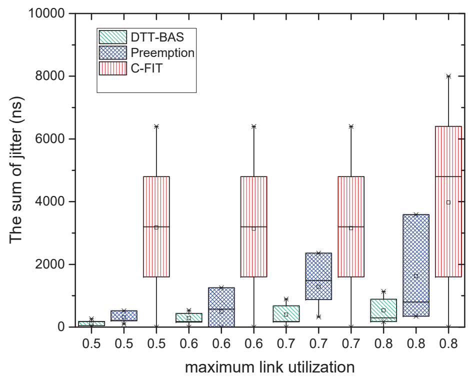
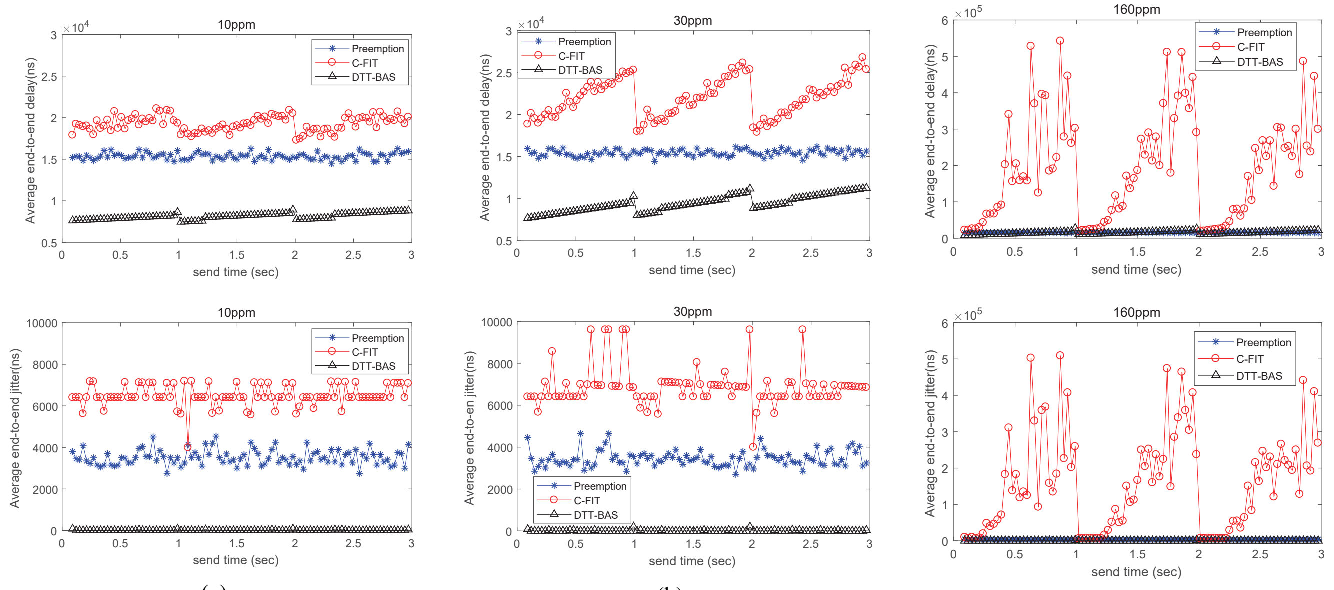


Fig. 5: The sum of jitter distribution with non-clock skews.



(a) (b) (c)

Fig. 6: Average e2e delay versus sending time for DTT-BAS, C-FIT and Preemtion algorithms with different clock skews.

frames missing their reserved time slots in the C-FIT. These missing reservations generate much longer waiting times for the queue. On the other hand, the DTT-BAS does not have a strict time slot allocation mechanism, which accordingly reduces the waiting time.

Simulation results reveal that both DTT-BAS and C-FIT are affected by the clock skews. In a topology environment where multiple upstream switches are provided, compared to frame preemption and C-FIT scheduling algorithms, the DTT-BAS greatly outperforms the C-FIT in terms of jitter while keeping the maximum delay close to the frame preemption. In conclusion, the DTT-BAS is proved to be able to achieve much lower jitter. Thus, the stability of the DTT-BAS is superior to the frame preemption and the C-FIT algorithms.

V. CONCLUSION

In this paper, a deterministic transmittable time-based asynchronous scheduler, namely DTT-BAS, is proposed. It numerically compares the timestamps to optimize the flow scheduling in the network. In theory, the DTT-BAS provides bounded delay and good jitter performance in regard to clock synchronization errors, fan-in phenomenon of the bridges and network fluctuations. In addition, it avoids the interference of different flows by separating the flows from different directions during the flow aggregation. Simulation results disclose that when the DTT-BAS scheduling algorithm runs in the fronthaul carried by a TSN network, it can provide unique jitter-free performance for the asynchronous scheduling scheme in a stable, clock-free scenario. Furthermore, the system is able to provide lower delays and achieve higher stability in scheduling for an environment where the clock skews are considered.

REFERENCES

[1] T. Pfeiffer. "Next Generation Mobile Fronthaul Architectures," in *Optical Fiber Communication Conference*, 2015, pp. M2J-7.

[2] T. Wan and P. Ashwood-Smith. "A Performance Study of CPRI over Ethernet with IEEE 802.1 Qbu and 802.1 Qbv Enhancements," in *Proc. IEEE Global Communications Conference*, 2015, pp. 1-6.

[3] A. Gupta and R. K. Jha. "A Survey of 5G network: Architecture and Emerging Technologies," *IEEE Access*, vol.3, pp. 1206-1232, July 2015.

[4] "Time-Sensitive Networking for Fronthaul," IEEE Standard P802.1CM, [Online]. Available: http://www.ieee802.org/1/pages/802.1cm.html.

[5] "Frame Preemption," IEEE Standard 802.1Qbu, [Online]. Available: http://www.ieee802.org/1/pages/802.1bu.html

[6] J. Bartelt *et al.* "Fronthaul and Backhaul Requirements of Flexibly Centralized Radio Access Networks," *IEEE Wireless Communications*, vol. 22, no. 5 pp. 105-111, Oct. 2015.

[7] N. J. Gomes "Fronthaul Evolution: From CPRI to Ethernet." in *Optical Fiber Technology*, 2015, pp. 50-58.

[8] T. T. Wang. "Consideration of Burstiness in Bridged Fronthaul Network," [Online]. Available: http://www.ieee802.org/1/files/public/ docs2017/cm-wangtt-burstiness-in-network-0117.pdf.

[9] "Enhancements for Scheduled Traffic," IEEE Standard 802.1Qbv, [Online]. Available: http://www.ieee802.org/1/pages/802.1bv.html.

[10] "CPRI Specification V7.0, Interface Specification, [Online]. Available: http://www.cpri.info/spec.html.

[11] D. Thiele and R. Ernst. "Formal Worst-case Performance Analysis of Time-sensitive Ethernet with Frame Preemption," in *Proc IEEE International Conference on Emerging Technologies and Factory Automation*, 2016, pp. 1-9.

[12] "Cyclic Queuing and Forwarding," IEEE Standard 802.1Qch, [Online]. Available:https://1.ieee802.org/tsn/802-1qch/.

[13] D. Thiele, R. Ernst and Jonas Diemer. "Formal Worst-case Timing Analysis of Ethernet TSN's Time-aware and Peristaltic shapers," in *Proc. IEEE Vehicular Networking Conference*, 2015, pp. 251-258.

[14] D. Chitimalla *et al.* "5G Fronthaul-latency and Jitter Studies of CPRI over Ethernet," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 9, no. 2, pp. 172-182, Feb. 2017.

[15] "Specification and Management Parameters for Interspersing Express Traffic," IEEE standard 802.1802.3br, [Online]. Available: https://ieeexplore.ieee.org/document/7592835/.

[16] J. Specht and S. Samii. "Urgency-based Scheduler for Time-sensitive Switched Ethernet Networks," in *Proc. IEEE Euromicro Conference on Real-Time Systems*, 2016, pp. 75-85.

[17] "Asynchronous Traffic Shaping," IEEE Standard 802.1Qcr, [Online]. Available: https://1.ieee802.org/tsn/802-1qcr/.

[18] J. Specht and S. Samii. "Synthesis of Queue and Priority Assignment for Asynchronous Traffic Shaping in Switched Ethernet," in *Proc. IEEE Real-Time Systems Symposium*, 2017, pp. 178-197.