CPRI over Ethernet: Towards fronthaul/backhaul multiplexing

Mahmoud Bahnasy
École de Technologie Supérieure
mahmoud-mohamed.bahnasy.1@etsmtl.net

Halima Elbiaze
Université du Québec à Montréal
elbiaze.halima@uqam.ca

Catherine Truchan
Ericsson Research
catherine.truchan@ericsson.com

***Abstract*—Ethernet has been proposed for the 5G fronthaul to transport the Common Public Radio Interface (CPRI) traffic between the radio equipment (RE) and the radio equipment control (REC). The advantages of adopting an Ethernet transport are threefold 1) the low cost of equipment, 2) the use of a shared infrastructure with statistical multiplexing, as well as 3) the ease of operations, administration and maintenance (OAM). In this paper, we introduce distributed timeslot scheduler for CPRI over Ethernet (DTSCoE) as a scheduling algorithm for IEEE 802.1Qbv to support CPRI traffic. DTSCoE is built upon the stream reservation protocol (SRP) IEEE 802.1Qcc to propagate timeslot information across the datapath without any centralized coordination. The simulation results demonstrate that DTSCoE reduces one-way delay to minimum and reduces the jitter to zero which satisfies the CPRI requirements.**

1. Introduction

The exponential increase in mobile network users and the enormous bandwidth required by new mobile applications lead to massive increase in mobile data traffic. It is expected that smartphone subscription will double to reach 6.4 billion while exchanging 1.6 zettabytes of data [1]. In addition, it anticipated that data traffic will grow annually by 45% until 2021. These characteristics require the envisioned 5G cellular system to provide very high rates (up to 10 Gbps per user) and sub-milliseconds latency, particularly for time-critical applications [2]. In order to achieve ultra-high user data rates, 5G networks require shorter radio transmission distance which could be achieved by distributing the remote radio heads RRHs into smaller cells.

A promising approach to reconcile these requirements with conservative investment is to split the functionality of the mobile network node into radio equipment (RE) and a radio equipment control (REC). RE is composed of a remote radio head (RRH) that provides basic radio functionality, while REC consists of a baseband unit (BBU) which processes baseband signals and is located in a central office. Originally, this solution was called centralized radio access network (C-RAN) since a single BBU can be shared by many RRHs. Based on traffic load, many lightweight RRHs could be deployed into smaller cells and connected to fewer BBUs in a centralized BBU pool as shown in Fig. 1. The emergence of virtualization and cloud computing with its cost efficiency, high performance, scalability, accessibility led to a novel proposal that virtualizes the BBU pool in the cloud. Therefore, the solution name changed from centralized RAN to cloud RAN (C-RAN) [3], [4].

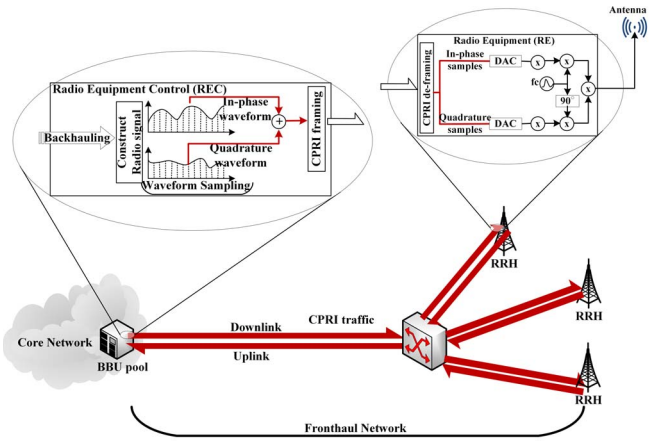

Figure 1: C-RAN architecture.

In traditional RAN, digitized baseband complex in-phase and quadrature I/Q quantized radio samples are transmitted between rooftop and basement. In C-RAN, the separation between BBU and RRH introduces the Fronthaul network. Fig. 1 depicts the separation between RRHs and BBU pools. It shows also the I/Q encoding/decoding process that occurs at the BBU pool and RRH in the downlink direction. The uplink encoding/decoding process is similar but in the reverse direction.

The Common Public Radio Interface (CPRI) [5] and the open base station architecture initiative (OBSAI) [6] were defined as transport protocols for Fronthaul network. CPRI is currently the most widely used protocol by RAN vendors for fronthaul transport. CPRI interface must satisfy fronthaul network requirements. For example, [4] depicts that base stations are required to prepare an acknowledgment (ACK)/non-acknowledgment (NACK) within 3 ms including RE and REC processing besides propagation delay, which leaves $100 - 200\ \mu s$ [4] for the fronthaul one-way delay. In addition, CPRI has a one-way jitter requirement of several nanoseconds.

978-1-5386-4790-5/18/$31.00 ©2018 IEEE

Moreover, The separation between radio channels in mobile networks must be fulfilled with a minimum frequency accuracy of ± 50 parts-per-billion (ppb) [7]. This raises the need for stringent clock synchronization in the fronthaul network.

Motivated by the need to reduce capital expenses (CAPEX) and operating expenses (OPEX), network operators strive to find alternative networks that can handle the increasing bandwidth requirements for CPRI traffic. Reusing existing packet switched networks (e.g. Ethernet) is one of the best solutions considered because of its cost efficiency, flexibility and scalability. Furthermore, it provides the integration of fronthaul and backhaul into a unified network. Ethernet network is widely used in access and data-center networks. It has also shown huge capacity growth lately. Accordingly, by encapsulating CPRI traffic over Ethernet, significant savings in CAPEX and OPEX can be achieved. In addition, the operations, administration and maintenance (OAM) capabilities of Ethernet provide standard methods for network management and performance monitoring.

The time-sensitive networking (TSN) task group (IEEE 802.1) [8] is currently working on a set of standards aiming to provide synchronized, low latency, and high bandwidth services over Ethernet networks. However, IEEE 802.1 was not developed originally for fronthaul traffic. Time-sensitive networking for fronthaul (IEEE 802.1CM) [9] project was proposed as a TSN profile for Fronthaul network. TSN project includes tools such as frame preemption (IEEE 802.1Qbu), scheduled traffic (IEEE 802.1Qbv) and stream reservation protocol (IEEE 802.1Qcc) that have been introduced as enhancements to Ethernet network.

In IEEE 802.1Qbu frame preemption, when a time-critical frame (TCF) is received while transmitting non-time-critical frame (NTCF), the switch stops processing the NTCF and begins processing the TCF. IEEE 802.1Qbv is a time-aware traffic shaper that opens or closes a gate for a traffic class. High priority traffic can be transmitted as soon as its gate opens which can reduce jitter. The current draft of IEEE 802.1Qbv does not define any scheduling algorithm, therefore, a scheduling algorithm must be defined. IEEE 802.1Qcc Stream Reservation Protocol (SRP) is proposed to be implemented on top of the existing Multiple Registration Protocol (MRP) 802.1Qak. The MRP protocol allows streams to register their attributes (e.g. bitrate, maximum delay, etc) across the network.

To this end, we propose DTSCoE as a scheduling algorithm for IEEE 802.1Qbv that uses IEEE 802.1Qcc to propagate timeslot information along the datapath. Due to the constant bitrate (CBR) nature of CPRI traffic, it could be transmitted with fixed inter-frame time. Consequently, reserving periodic timeslot is feasible and efficient in delivering CPRI traffic.

The rest of this paper is organized as follows. Section 2 provides related work. DTSCoE is described in detail in section 3. Section 4 provides the simulation results of DTSCoE. Section 5 provides some insight discussion on DTSCoE. Finally, conclusion and future work are introduced in section 6.

2. Related Work

IEEE 802.1CM aims to enable the transport of Fronthaul streams in Ethernet network using TSN project (IEEE 802.1) tools such as frame preemption (IEEE 802.1Qbu), scheduled traffic (IEEE 802.1Qbv) and stream reservation protocol (IEEE 802.1Qcc).

Buffering at egress devices is proposed to address CPRI jitter requirement. [10] showed by simulation that buffering at the edge could eliminate jitter. Yet, implementing buffering at egress devices is a challenge because it must cope with the worst case delay. In addition, it introduces a fixed delay that is equal to the worst case delay regardless of the network path. While this is suitable for streaming traffic, it introduces an extra delay that reduces CPRI traffic performance.

It was shown by simulation in [11] that a CPRI over Ethernet network of 3 switches without TSN features causes jitter of 3 μs. But when frame preemption is enabled, jitter was limited to 100 ns. It shows also that when IEEE 802.1Qbv is enabled, jitter reduced to 50 ns, and when a 70 ns guard band was added on top of preemption the jitter was reduced to 0 ns. The delay for all of these cases was below 26 μs.

For a larger network typology, [10] showed that frame preemption with dedicated or shared Ethernet network failed to satisfy CPRI jitter requirement. On the contrary, they demonstrate, with simulation, promising results using IEEE 802.1Qbv and a local scheduling algorithm. Since the scheduling algorithm in [10] reserves timeslots for IEEE 802.1Qbv at each network node independently, contention resolution may generate extra delay and jitter. Such algorithm resolves contention by queuing CPRI frames and shifting its timeslot without coordination with other network nodes. Indeed, this lack of coordination causes high delay/jitter as it is explicitly shown by the simulation results in [10] where the jitter increases up to 1000 ns when frame size changes. Whereas, our proposed solution propagates timeslot information through network path without any centralized coordination. This distributed coordination guarantees that each registered CPRI frame will pass through the network without queuing which induces minimum latency and zero jitter.

In addition, [12] introduces ZeroJitter as a centralized solution for CPRI over Ethernet that uses a software-defined network (SDN) controller to define a scheduling algorithm form IEEE 802.1Qbv. However, in this work, we avoid centralized solutions for two reasons. First, centralized solutions face scalability issue which rises as the network expands and the number of flows increases. Second, in centralized solutions, transmission time and propagation delay must be accounted in the reservation algorithm. Therefore, the centralized solution requires, in addition to the network topology, network dimensions in terms of cables' length and capacity. While in our proposal, the same transmission/propagation delay is applied for reservation requests as well as CPRI frames. Therefore, no need to explicitly consider it in the calculation (review section 3.2 for more details).

C-FIT scheduling algorithm is proposed in [13] to minimize CPRI jitter in Ethernet network. However, this proposal is built on the assumption that all CPRI traffic is multiplexed into the network using one switch which is not realistic. In case of aggregating CPRI traffic on multiple switches, a centralized schedule needs to be formed using an SDN controller. In addition, the C-FIT algorithm complexity increases exponentially with the number of flows, which increases the required computational power and raises a scalability issue.

3. Distributed Timeslot Scheduler for CPRI over Ethernet (DTSCoE)

Two major challenges that face CPRI over Ethernet are packet delay and jitter of every packet depending on queue occupancy. To address these issues, we propose DTSCoE as a distributed timeslot scheduling algorithm for IEEE 802.1Qbv. DTSCoE mechanism is divided into three key processes i) timeslot declaration-to-registration process, ii) contention resolution, iii) per-node synchronization.

3.1. Timeslot declaration-to-registration process

Similar to IEEE802.1cc SRP, DTSCoE declares and registers network resources for certain flows where timeslot is a resource. The DTSCoE notations are summarized in table 1.

Ts	CPRI sending time
TW	Time window size (in ns)
L	CPRI frame size (in bit)
t_L	Timeslot duration (in ns)
C	Link capacity
R	CPRI flow rate
$RTab$	Registration table
$TsVec$	Timeslot vector
T_{abs}	Absolute value of Ts
T_{arr}	Registration request arrival time
T_{tr}	Registration request transmission time
Ts_{alloc}	The newly allocated Ts
ΔTs	Ts change that applies after allocating a new timeslot
T_p	Processing time
d_p	propagation/transmission delay

TABLE 1: DTSCoE notations

In DTSCoE, each CPRI source propagates its intention of sending information to a destination. CPRI sources convey their sending intention by defining sending time Ts within a periodic time cycle, which we call time window TW. CPRI sources do not propagate a time-defined timeslot width, instead, they define a maximum frame size L. Thus, each network node defines the timeslot duration t_L based on its link capacity C as follow $t_L = L/C$.

In DTSCoE, the registration process is initialized, similar to SRP, by the source. Each source propagates a registration request that includes its stream attributes (in this case: Flow ID, TW, Ts, and L) toward the destination. This process is repeated until it reaches the destination. The declaration-to-registration reaction always occurs in a single direction from the declaring participant to its adjacent node. Therefore, a bidirectional connection between RE and REC occurs in two separate processes. Each network node registers its information in a local table called registration table $RTab$. This table is used to define a timeslot vector $TsVec$ with a size equal to the least common multiple LCM of all declared TWs. Each entry i in $RTab$ table must be represented by several entries in $TsVec$ vector of width $t_L = L/C$ and start at Ts_j defined by (1).

$$Ts_j = j \times TW_i + Ts_i \tag{1}$$

Where Ts_i and TW_i represent the request sending time and time window respectively, and $j = \{0, 1, \ldots, LCM/TW_i\}$. If the requested timeslot (defined by Ts, L & TW) is not available, the network node allocates a new timeslot by searching for a new timeslot that has all Ts_j represented by (1) available.

Hence, it updates the registration request with the new Ts and it sends back an update acknowledge that carries the sending time change $\Delta Ts = Ts_{alloc} - Ts$, where Ts_{alloc} is the newly allocated Ts. Additionally, each node adds its currently occupied timeslots to the forwarded request. By adding the occupied timeslots, in the registration request, each node can avoid allocating a timeslot that is already allocated by other nodes while searching for a new timeslot. This facilitates the reservation process and ensures that it occurs in a single step. Once the registration request reaches the destination, it replies with one of the following attributes (i) ready, (ii) ready failed (at least one node is unable to reserve resources), (iii) asking failed (there are no destinations that succeeded in reserving resources), (iv) update or (v) ignore. Finally, RRHs can start sending their CPRI flows within the reserved timeslot by using the final Ts.

Fig. 2 depicts a declaration-to-registration use case. The process takes place in the following order:

i) A source (RRH in this case) sends a registration request with frame size L and $TW = L/R$, where R is the CPRI traffic rate.
ii) When switch 1 receives this registration request, it adds this request in its $RTab$ table and searches for overlap in its local $TsVec$ vector.
iii) In this case, switch 1 does not find any overlaps, consequently, it forwards the registration request to switch 2.
iv) At switch 2, an overlap is found, therefore, switch 2 allocates a new timeslot and updates Ts.
v) Switch 2 forwards the registration request to the BBU pool and sends an update acknowledge backward to inform the previous network nodes about the timeslot modification $\Delta Ts = Ts_{alloc} - Ts$.
vi) Each switch/host, that receives the update acknowledge, updates its $RTab$ table and $TsVec$ vector accordingly.
vii) At the BBU pool, a ready acknowledge is created and sent back to the source.

Hence, Declaration-to-registration process spreads timeslot reservation information along the datapath without any centralized coordination which avoids scalability issue that faces centralized solutions.

Background traffic might experience long queuing delay due to CPRI traffic timeslot reservation. Therefore, a proper flow control mechanism must be used to control its transmission such as Priority-based Flow Control (PFC) [14].

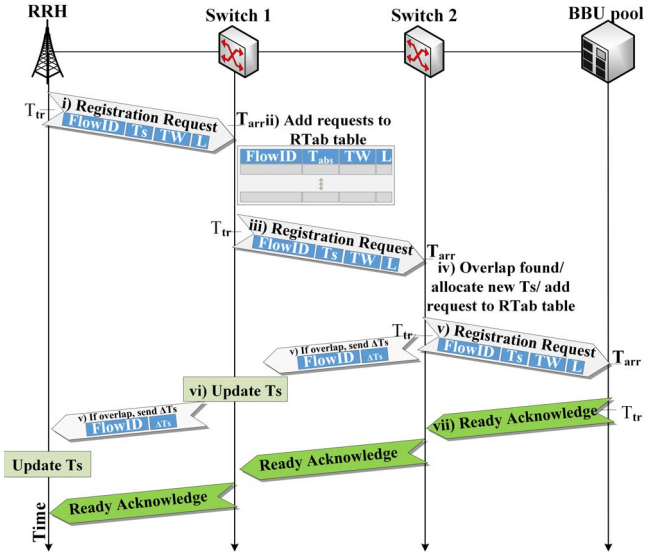

Figure 2: Declaration-to-registration reaction

3.2. Per-node synchronization

DTSCoE requires strict clock synchronization, among network nodes, in order to perform timeslot reservation. Several protocols were proposed to perform RE-to-REC synchronization in the fronthaul network such as IEEE 1588 precision time protocol PTP [15] that provides synchronization at layer 2 and higher. Also, ITU-T has defined synchronous Ethernet (Sync-E) in G.8261 standard [16] that requires all network nodes to participate in clock synchronization. Other research such as network time protocol (NTP) [17] provides synchronization over Layer 3. It is expected that fronthaul network must use one of the aforementioned clock synchronization protocols. However, we propose a simpler algorithm that does not require per-node synchronization while the synchronization between RE and REC could be provided by a higher layer protocol. DTSCoE uses the registration request transmission/arrival time at each node as a reference time. We can think of each registration request as a description of each host sending behavior in the future TW time as shown in Fig. 3. Thus, Ts represents the duration between the request transmission time T_{tr} and CPRI transmission time. Therefore, each network node maps Ts to T_{arr} once the request is received and calculates its absolute sending time T_{abs} using (2). In addition, it updates Ts before forwarding by subtracting the processing time T_p in order to remap it to the request transmission time T_{tr} using (3).

$$T_{abs} = Ts + T_{arr} \tag{2}$$

$$Ts \leftarrow (Ts - T_p) \pmod{TW} \tag{3}$$

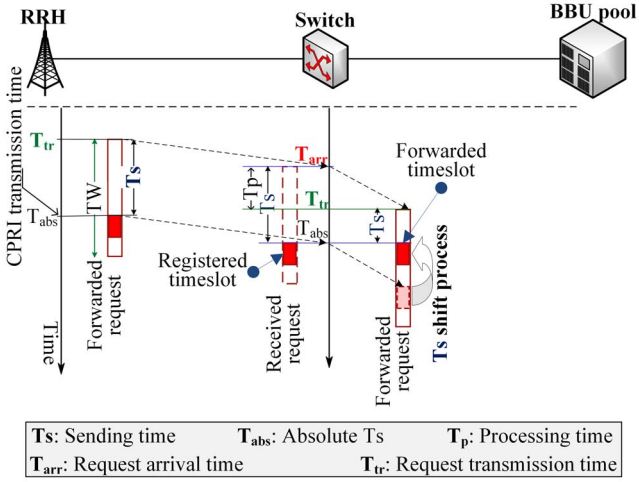

Figure 3: Ts shifting process: each network node updates Ts in order to remap it to T_{tr}

Due to the reserved timeslot, CPRI traffic passes through the network with no processing time, including queuing and processing delays, unlike the reservation request. Therefore, each node shifts Ts forward in the timeslot registration by the processing time ($T_p = T_{arr} - T_{tr}$) (Fig. 3). This process is called Ts shifting and it forces T_{abs} to be identical for both the received request and the forwarded request as shown in Fig. 3 at the switch. Performing the same process at each network node forces the request timeslot to be delayed only by the propagation/transmission delay which is identical for both CPRI frame and the registration request. Hence, shifting the reservation inside TW by the processing time resolves the synchronization issue while considering implicitly the propagation/transmission delay. We propose to add padding to the registration request to match the length of CPRI frames. Therefore, transmission time will be identical for both CPRI frames and registration request frame.

Timeslot registration process in DTSCoE is shown in algorithm 1. For each new request, parameter initialization occurs in lines 1 - 7. Then, $TsVec$ is initialized in line 8. The LCM is calculated and used to update $TsVec$ in lines 9 and 10. After that, in line 11 - 13, network node verifies if the requested timeslot is available, otherwise it allocates a new one. In line 14 - 16, the registration request is updated with the occupied timeslots in the requested path. In line 17, the requested timeslot is registered in $RTab$. Finally, in line 18 - 22, the Ts shifting process takes place, and the request is forwarded to the adjacent node.

Algorithm 1: Timeslot registration algorithm

Input : reservation request RES_REQ & registration table $RTab$
Output: timeslot vector $TsVec$

foreach RES_REQ **do**
  $T_{arr} \leftarrow read\_frame\_arrival\_time$ $TW \leftarrow$ time window size;
  $L \leftarrow$ frame size;
  $Ts \leftarrow$ read Ts;
  $T_{abs} \leftarrow Ts + T_{arr}$;
  $t_L \leftarrow L/C$ $TsVec \leftarrow RTab.getRTab()$ **if** $(LCM(TW, TsVec.size) != TsVec.size)$ **then**
    $TsVec.size \leftarrow LCM(TW, TsVec.size)$;
  if $(TsVec(T_{abs} : t_L)\ is\ NOTIDLE)$ **then**
    $Ts_{alloc} \leftarrow TsVec.nextAvailableTimeSlot(t_L)$ - T_{arr}
    $RES\_REQ.Ts = Ts_{alloc}$
  for i=0;i<$RTab.size()$; i++ **do**
    RES_REQ.add($RTab(i)$)
  end
  $RTab.addTimeSlot(T_{abs} : t_L)$;
  $T_{tr} \leftarrow current\_time$ **for** j=0;j<RES_REQ.size(); j++ **do**
    $Ts \leftarrow \big(Ts - (T_{arr} - T_{tr})\big)$
  end
  Forward RES_REQ;
end

3.3. Contention resolution

DTSCoE uses the fact that multiple CPRI basic frames are encapsulated in one single Ethernet frame and we can slide the encapsulation process within a predefined time window. Therefore, in order to avoid long negotiation between network nodes, if the requested timeslot is occupied, we propose that each CPRI flow registers a timeslot defined by a flexible start time Ts within a time window TW. Thus, network nodes can reserve a new timeslot within the requested TW in case of overlap (step (iv) in Fig. 2). We believe this relaxed constraint will reduce the registration phase to one step rather than renegotiating new timeslot.

By sliding the encapsulation process within the window size, we give the switch enough time to finish transmitting each flow frame in its dedicated timeslot. Therefore, CPRI frames of all flows are transmitted sequentially with zero queuing delay. In addition, one queue is required to support all CPRI traffic of all flows, contrary to the the proposed solutions in [10], [12] that need one queue per CPRI flow.

4. Simulation

To evaluate the performance of the proposed mechanism, we use OMNEST [18] to build a simulation model for IEEE 802.1Qbv with DTSCoE. Two experiments are conducted in this simulation environment to study the effect of different frame sizes on DTSCoE and to evaluate the performance of DTSCoE when different CPRI traffic are aggregated at multiple switches. In this simulation, we have a shared Ethernet network that provides a transport network for CPRI and data traffic. We compare the results of our proposed scheduling algorithm against Ethernet network with and without frame preemption (IEEE 802.1Qbu). Similar to [10] and [19], we estimate the performance of Ethernet with frame preemption by setting data traffic frame size to 128 bytes. Throughout all simulations, we use one strict priority queue per output port for CPRI traffic.

4.1. Experiment I - The effect of frame sizes

In this experiment, we study the effect of frame size on the performance of DTSCoE using the topology that is depicted in Fig. 4. Eight CPRI flows with different frame sizes (as shown in Fig. 4) and fixed inter-packet gap, that is equal to 1.920 μs, send data to a BBU pool. In addition, eight data flows transmit data to one data center. Data flows vary their frame sizes using normal distribution with average frame sizes as shown in Fig. 4 and a standard deviation = 150. Moreover, all data flows vary their inter-packet gaps using exponential distribution with mean time = 1.920 μs. DTSCoE is built on the assumption that Ethernet network is not oversubscribed. Therefore, we set all links capacity to 60 $Gbps$ in order to accommodate the highest possible data transmission.

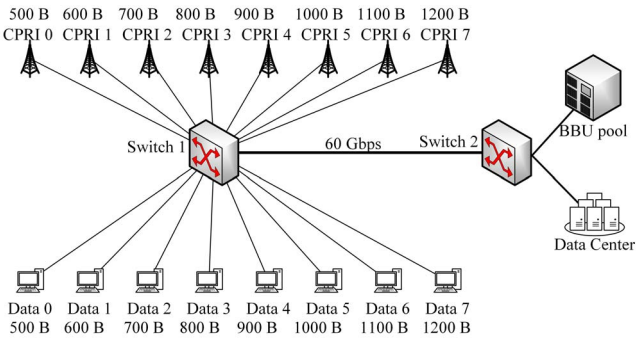

Figure 4: Experiment I topology

Fig. 5 depicts the delay and its standard deviation for Ethernet with strict priority, Ethernet with frame preemption and IEEE 802.1Qbv with DTSCoE. It shows that as the frame size increases, the delay increases dramatically for Ethernet with or without frame preemption. On the other hand, IEEE 802.1Qbv with DTSCoE experience the increase that is caused by the transmission time due to frame size increase. In addition, the figure shows that delay standard deviation for Ethernet with strict priority is higher than

Ethernet with frame preemption. Whereas, IEEE 802.1Qbv with DTSCoE experience no delay variation.

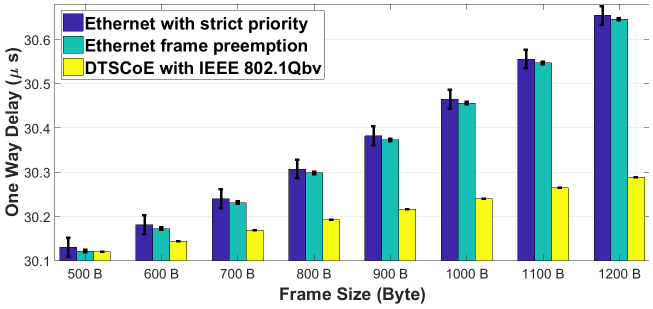

Figure 5: Delay and delay standard deviation

Fig. 6 depicts the average jitter for the CPRI flows of different frame sizes. It shows that Ethernet with strict priority (without frame preemption) experiences high jitter (on average 16 ns and up to 120 ns). It depicts also that Ethernet with frame preemption failed to achieve zero jitter. On the other hand, IEEE 802.1Qbv with DTSCoE succeeded in achieving zero jitter as DTSCoE successfully reserved timeslot for each CPRI flow. Therefore, CPRI traffic is served instantly as it arrives at the switch.

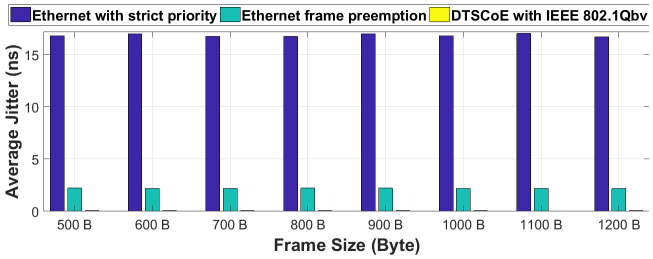

Figure 6: Jitter for flows of different packet sizes

4.2. Experiment II - The performance of DTSCoE

In this experiment, we aim to study the effect of using multiple aggregation points for CPRI traffic. Therefore, in this experiment, we evaluate the performance of DTSCoE using the topology depicted in Fig. 7 where we have 2 layers of aggregation. First, at the access switches (switches 0 - 3) where two CPRI flows and two data flows are connected directly to each switch. Second aggregation layer at switches 4 and 5 where the multiplexed traffic groups join each other. The sum of eight CPRI flows and eight data flows transmit data to one BBU pool and one data center respectively.

In addition, we study the effect of multiplexing CPRI flow of different packet sizes, inter-packet gaps, and rates by setting transmission parameters of each flow as depicted in Table 2. Each CPRI flow sends with constant frame size and fixed inter-packet gap. Data flows vary their frame sizes using normal distribution (Average = 600 Bytes, standard deviation = 150) and inter-packet gap using exponential distribution. All links in this topology have a capacity of 100 $Gbps$.

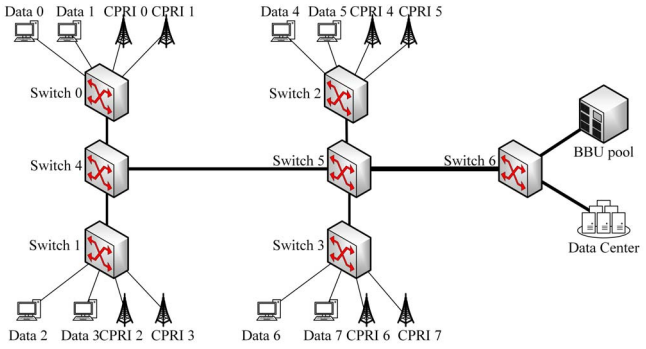

Figure 7: Experiment II topology

FID	CPRI			Data
	Frame size	Inter-packet gap	Rate	Average Rate
0	600 B	640 ns	7.5 Gbps	7.5 Gbps
1	600 B	1920 ns	2.5 Gbps	2.5 Gbps
2	750 B	960 ns	6.25 Gbps	6.25 Gbps
3	600 B	1920 ns	2.5 Gbps	2.5 Gbps
4	600 B	640 ns	7.5 Gbps	7.5 Gbps
5	600 B	1920 ns	2.5 Gbps	2.5 Gbps
6	750 B	960 ns	6.25 Gbps	6.25 Gbps
7	600 B	1920 ns	2.5 Gbps	2.5 Gbps

TABLE 2: CPRI/Data transmission parameters

Fig. 8 depicts the average delay for Ethernet network with and without frame preemption and DTSCoE. It shows that IEEE 802.1Qbv with DTSCoE achieves the minimum delay for all flows. One can notice that flows that cross 4 switches (flow 0, 1, 2 & 3) have higher delay than these that cross 3 switches (flow 4, 5, 6 & 7). Moreover, Fig. 9 shows the average jitter measured during the experiment. It also depicts the maximum and minimum jitter. The results demonstrate that Ethernet without frame preemption induces jitter up to 310 μs, while Ethernet with frame preemption introduces jitter as high as 250 ns. In contrast, DTSCoE achieves zero jitter regardless of frame size or inter-packet gap. One can notice that network complexity affects the performance of Ethernet network with and without frame preemption since in this topology jitter is higher than experiment I. Whereas, the proposed solution is not affected by network complexity and achieves zero jitter in both experiments.

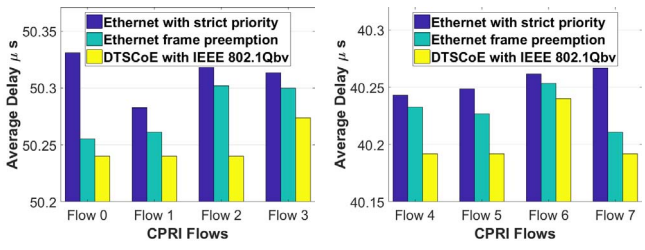

Figure 8: Average delay for CPRI flows

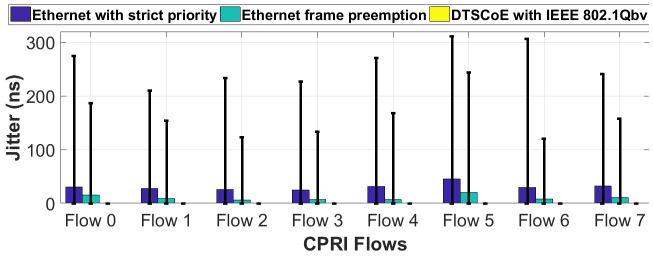

Figure 9: Average, minimum and maximum jitter for CPRI flows

5. Discussion

In this section, we present some remarks regarding DTSCoE implementation. First, clock drifting between network nodes could cause, in long-term, timeslot overlap. To overcome this issue, each CPRI node can update the registration process after a predefined time T. Further, we propose using a guard band before the timeslot start time that is equal to the largest possible time drift per T.

Second, because DTSCoE is a distributed mechanism, it is possible that a timeslot overlaps after applying ΔTs with a recently registered request. In DTSCoE current implementation, each switch delays all new registration requests until the pending request are fulfilled. This issue could be resolved differently by sending a fail acknowledge to the source and the source could send a new registration request.

6. Conclusion

In this paper, we presented DTSCoE as a distributed scheduling algorithm for IEEE 802.1Qbv that uses IEEE 802.1Qcc. DTSCoE nodes declare their intention of transmitting data by sending a registration request, that includes sending time Ts, time window size TW and maximum frame length L, through the datapath. The overall feasibility and performance of DTSCoE are assessed and evaluated through simulation. Our results show that DTSCoE with IEEE 802.1Qbv outperforms Ethernet with or without frame preemption and achieves minimum network latency and zero jitter.

For future work, further study is required to understand the impact on delay and jitter due to frame length variation caused by CPRI compression. In addition, a thorough study of the behavior of DTSCoE with flows of different time window sizes TW and different frame sizes is required.

Acknowledgments

This work is supported by Ericsson Research, the Fonds de Recherche Nature et Technologies (FRQNT) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] Ericsson, “Ericsson Mobility Report,” 2016. [Online]. Available: https://www.ericsson.com/mobility-report

[2] ——. (2016) 5G radio access capabilities and technologies.

[3] Y. Lin, L. Shao, Z. Zhu, Q. Wang, and R. K. Sabhikhi, “Wireless network cloud: Architecture and system requirements,” IBM Journal of Research and Development, vol. 54, no. 1, pp. 4:1–4:12, January 2010.

[4] C. Mobile, “C-RAN: the road towards green RAN,” White Paper, ver, vol. 3, 2013.

[5] Ericsson AB, Huawei Technologies Co. Ltd, NEC Corporation, Alcatel Lucent, and Nokia Networks. (2015) Common Public Radio Interface (CPRI) specification V7.0. [Online]. Available: http://www.cpri.info/downloads/CPRI_v_7_0_2015-10-09.pdf

[6] (2013) OBSAI specification.

[7] LTE, ETSI, “Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) radio transmission and reception (3GPP TS 36.104 version 8.6. 0 Release 8),” ETSI TS, vol. 136, no. 104, p. V8, July 2009.

[8] Institute of Electrical and Electronics Engineers. (2017) IEEE 802.1 Time-Sensitive Networking Task Group. [Online]. Available: http://www.ieee802.org/1/pages/tsn.html

[9] Institute of Electrical and Electronic Engineers. (2017) IEEE 802.1: 802.1CM - Time-Sensitive Networking for Fronthaul. [Online]. Available: http://www.ieee802.org/1/pages/802.1cm.html

[10] T. Wan and P. Ashwood, “A performance study of CPRI over Ethernet,” IEEE 1904.3 Task Force, 2015.

[11] J. Farkas and B. Varga. (2015) Applicability of 802.1Qbu and 802.1Qbv to fronthaul.

[12] T. Wan, B. McCormick, Y. Wang, and P. Ashwood-Smith, “ZeroJitter: An SDN Based Scheduling for CPRI over Ethernet,” in 2016 IEEE Global Communications Conference (GLOBECOM), Dec 2016, pp. 1–7.

[13] D. Chitimalla, K. Kondepu, L. Valcarenghi, M. Tornatore, and B. Mukherjee, “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.

[14] “IEEE Standard for Local and metropolitan area networks–Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks–Amendment 17: Priority-based Flow Control,” IEEE Std 802.1Qbb-2011 (Amendment to IEEE Std 802.1Q-2011 as amended by IEEE Std 802.1Qbe-2011 and IEEE Std 802.1Qbc-2011), pp. 1–40, Sept 2011.

[15] “IEEE Standard for a precision clock synchronization protocol for networked measurement and control systems,” IEC 61588:2009(E), pp. C1–274, Feb 2009.

[16] ITU-T, Recommendation G.8261: Timing and synchronization aspects in packet networks, ITU-T G.8261/Y.1361, 2013.

[17] D. Mills, J. Martin, J. Burbank, and W. Kasch, “Network time protocol version 4: Protocol and algorithms specification,” Tech. Rep., 2010.

[18] OMNEST Community. (2014) OMNEST - High-Performance Simulation. [Online]. Available: http://www.omnest.com/

[19] T. Wan and P. Ashwood-Smith, “A Performance Study of CPRI over Ethernet with IEEE 802.1Qbu and 802.1Qbv Enhancements,” in 2015 IEEE Global Communications Conference (GLOBECOM), Dec 2015, pp. 1–6.