5G Fronthaul–Latency and Jitter Studies of CPRI Over Ethernet

Divya Chitimalla, Koteswararao Kondepu, Luca Valcarenghi, Massimo Tornatore, and Biswanath Mukherjee

***Abstract*—Common Public Radio Interface (CPRI) is a successful industry cooperation defining the publicly available specification for the key internal interface of radio base stations between the radio equipment control (REC) and the radio equipment (RE) in the fronthaul of mobile networks. However, CPRI is expensive to deploy, consumes large bandwidth, and currently is statically configured. On the other hand, an Ethernet-based mobile fronthaul will be cost-efficient and more easily reconfigurable. Encapsulating CPRI over Ethernet (CoE) is an attractive solution, but stringent CPRI requirements such as delay and jitter are major challenges that need to be met to make CoE a reality. This study investigates whether CoE can meet delay and jitter requirements by performing FPGA-based Verilog experiments and simulations. Verilog experiments show that CoE encapsulation with fixed Ethernet frame size requires about tens of microseconds. Numerical experiments show that the proposed scheduling policy of CoE flows on Ethernet can reduce jitter when redundant Ethernet capacity is provided. The reduction in jitter can be as large as 1 μs, hence making Ethernet-based mobile fronthaul a credible technology.**

***Index Terms*—5G; CPRI over Ethernet; Fronthaul; Jitter; Scheduling; Time-sensitive networking (TSN).**

I. Introduction

Extensive adoption of smartphones and smart devices has enormously increased bandwidth consumption in cellular networks [1], thus calling for effective ways to improve cellular capacity. For example, 5G bandwidth consumption is expected to be 1000× of 4G [1,2], which calls for novel radio access network (RAN) architectures that can support much higher bandwidths in a cost-effective manner. A popular approach is to split the functionalities of 4G evolved NodeB (eNB) into radio equipment (RE), consisting of antenna and basic radio frequency (RF) functionality, and a radio equipment controller (REC), which processes the signals from the physical layer and above. This solution was originally called centralized RAN (C-RAN), as multiple RECs could be consolidated in a single centralized location, and a single REC can be shared among many REs, depending on traffic load. C-RAN can significantly increase the cellular coverage density by deploying many REs, which are lightweight compared with full-fledged macro-base stations, and thereby reducing network cost by using fewer RECs. Recent proposals push the REC function into the "cloud" (where the REC is "virtualized"), thereby moving from centralized-RAN to cloud-RAN and virtualized-RAN (V-RAN) [3].

Several ongoing projects, such as the Institute of Electrical and Electronics Engineers (IEEE) Standards Association 1914.1 working group [4], are striving to define an interface (electrical, optical, or wireless) between REC and RE. The interface requirements depend on the functional split [5], which, as proposed by 3GPP, is the set of functionalities that exist in the RE and REC. The split can occur at several protocol layers, thus resulting in different bandwidth and delay requirements of the mobile fronthaul. Our study considers the split at the physical layer of eNB (i.e., Option 8 in TR 38.801), which includes the entire layer 1 and above functions in the REC, whereas RE is a lightweight antenna having only RF functionality. In this option in-phase quadrature (IQ) samples of the baseband signal must be transported between RE and REC. Common Public Radio Interface (CPRI) is a well-known radio interface developed by several leading telecom vendors to transport sampled RF data between the RE and the REC. CPRI is a constant-bit-rate (CBR) interface with line rate options ranging from 614.4 Mbps (option 1) up to 24.33 Gbps (option 10) [6]. CPRI is a product of industry cooperation, which is of a closed nature, while other interfaces of a more open nature exist (e.g., Open Base Station Architecture Initiative (OBSAI) and Open Radio Equipment Interface (ORI)) [7,8].

CPRI is manufactured in low volumes, thus making it expensive. It is also extremely difficult to design switching equipment for CPRI. Although CPRI mentions that it supports several topologies such as tree, ring, and chain [6], there is no mention of how these topologies can be controlled. CPRI has stringent delay and jitter requirements, which can be satisfied only with high-speed fronthaul solutions (e.g., optical links) as in [9]. All these issues make it imperative to design a cost-efficient and reconfigurable mobile fronthaul that supports emerging network paradigms.

Manuscript received September 6, 2016; revised December 1, 2016; accepted December 6, 2016; published February 1, 2017 (Doc. ID 275153).

D. Chitimalla (e-mail: dchitimalla@ucdavis.edu), M. Tornatore, and B. Mukherjee are with the University of California, Davis, California 95616, USA

K. Kondepu is with the Scoula Superiore Sant' Anna, Pisa, Italy. He is also with the High Performance Networks Group, University of Bristol, Bristol BS8 1TH, UK

L. Valcarenghi is with the Scoula Superiore Sant' Anna, Pisa, Italy

https://doi.org/10.1364/JOCN.9.000172

Encapsulating CPRI over Ethernet (CoE) is a cost-efficient solution that can leverage existing Ethernet interfaces and switching equipment for mobile fronthaul. Ethernet has many advantages such as easy upgradability to higher data rates, wide-scale availability, low-cost equipment, and ease of scalability. Moreover, Ethernet switches can be used to configure a fronthaul into any network topology, even on a large network scale. Another advantage of utilizing CoE is that current high-speed optical networks can also be utilized for mobile fronthaul. In particular, 10 Gigabit (10G) Ethernet is fast enough to carry high-data-rate sampled IQ signals from the REC to the RE (e.g., a 20-MHz single-antenna I/O sampled radio signal can be handled by a 10G Ethernet interface). Transport options such as dedicated fiber, an optical transport network (OTN), and a passive optical network (PON) [10] can support the fronthaul by deploying fibers and other optical components (e.g., switches, optical line terminals (OLTs) for eNB-to-eNB communication). However, whether Ethernet can support stringent CPRI requirements in terms of delay and jitter is under scrutiny as the Ethernet mobile fronthaul needs to support delay within 100 μs and jitter within 65 ns [6], among other strict requirements of the time-sensitive IQ data that is being transmitted.

An ongoing effort by industry [11,12] and academia [13–16] is investigating an Ethernet fronthaul solution. The IEEE Standard Association (SA) 1914 working group has been effective since 2015 to standardize radio over Ethernet (RoE) [4]. In particular, the IEEE 1914.3 task force is investigating ways of transferring IQ user-plane data, vendor-specific data, and control and management (C&M) information channels [6] over an Ethernet-based packet-switched network. This standard focuses on encapsulating data into the Ethernet frame payload field with an additional RoE header for timing and synchronization purposes. Two types of encapsulation are defined in RoE: structure-aware and structure-agnostic. Structure-aware encapsulation uses knowledge of the encapsulated and digitized radio transport format content, whereas structure-agnostic encapsulation is a container that encapsulates bits into Ethernet frames irrespective of the encapsulated protocol. The applicability of Ethernet to mobile fronthaul has been discussed in [11] by exploiting the buffers to reduce the jitter of Ethernet packets. However, there is no experimental or simulative study quantifying the jitter in the proposed Ethernet fronthaul implementation. Several time-sensitive networking (TSN) Ethernet techniques (e.g., 802.1Qbu frame preemption and 802.1Qbv with guard band) for carrying fronthaul data have been compared in [12]; however, there is no detailed study on under which conditions of Ethernet with scheduled traffic can achieve less than 65 ns. Moreover, to minimize jitter in Ethernet fronthaul, scheduling Ethernet frames with fixed timeslots to a specific flow has been proposed in [13]. A functional split between the REC and RE, which permits baseband signal transport instead of the transport of sampled radio streams to enable lower-rate fronthaul, has been proposed in [14]. Such a fronthaul can also make use of Ethernet switches and networking statistical multiplexing gains, as it transports relatively bursty data instead of continuous radio waveforms. Furthermore, [15] provides experimental realization of dynamically reconfigurable CoE and also provides delay analysis of dynamically reconfigurable Ethernet fronthaul. Reference [16] discusses the advantages of having Ethernet fronthaul in a reconfigurable scenario.

There are investigations within IEEE 802.1 CM whether IEEE 802.1Qbu [17] and IEEE 802.1Qbv [18] using preemption and scheduling could be utilized to guarantee latency and jitter requirements for Ethernet fronthaul. IEEE 802.1Qbu is utilizing frame preemption policies where IEEE 802.3br provides the mechanism to implement preemption at the media access control (MAC) and below layers. IEEE 802.1Qbv is working on scheduled traffic with an edge buffer, which absorbs variation in packet delay with the added delay cost. The works in [19,20] provide enhancements to IEEE 802.1Qbu and IEEE 802.1Qbv standards. These studies have shown that (i) using 802.1Qbu preemption in Ethernet cannot meet jitter requirements of 65 ns and (ii) 802.1Qbv using Ethernet scheduling can remove jitter in some cases depending on the input flows but not always. 802.1Qbv utilizes guard bands to absorb fluctuations in the schedule of Ethernet packets. The size of the guard band determines the performance of 802.1Qbv Ethernet, where small guard band size increases packet collisions, and large guard band size decreases the effective throughput of Ethernet. IEEE 802.1Qbv and IEEE 802.1Qch address the synchronization problems such as latency and jitter in networks where time-sensitive data share capacity along with non-time-sensitive data. In particular, IEEE 802.1Qch describes the methods that can be adopted to schedule flows at strict time intervals using on–off gates for scheduled Ethernet. IEEE 802.1Qbv enhances the methods suggested in 802.1Qch to include VLAN tags to prioritize time-sensitive traffic such that delay/jitter are reduced. Our work assumes that the fronthaul network is capable of implementing the methods as described by Qch and Qbv. However, these standards do not explicitly describe any algorithm to minimize jitter in Ethernet fronthaul. In this work, we provide a scheduling algorithm for CoE data such that jitter remains within 65 ns for the given CoE data rates, which is not specified in Qbv/Qch. We also estimate the Ethernet capacity required to achieve tolerable jitter (65 ns) for a given set of CoE flows in the Ethernet fronthaul.

Our study provides a quantitative performance evaluation of CoE in terms of delay and jitter. An FPGA pre-synthesis evaluation is performed to verify the logical functionality of CoE design and encapsulation overhead. Moreover, we exploit advances in TSN such as scheduling Ethernet (IEEE 802.1Qbv) to devise an exhaustive-search algorithm that returns jitter-reduced frame scheduling.

The rest of the study is organized as follows. In Section II, we discuss the CoE-based mobile fronthaul architecture. In Section III, we give the mapping between CPRI and Ethernet frames, where we also evaluate its important parameters such as encapsulation delay, Ethernet overhead, and distance supported by Ethernet fronthaul. Section IV discusses jitter-minimization techniques for CoE.

We propose algorithms that can be programmed in the Ethernet switch that reduce jitter in Ethernet fronthaul. In Section V, we perform Verilog experiments and simulations to evaluate the delay and jitter of CoE-based fronthaul. Section VI concludes the study.

II. CoE-Based C-RAN Architecture

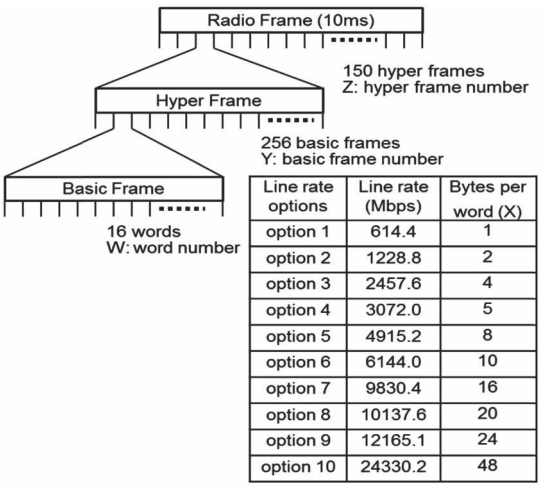
A. Frame Structure

CPRI sends sampled IQ data in a frame format, as shown in Fig. 1. It uses fixed-bandwidth connections between the REC and the RE with different line rates (options 1 to 10) [6]. CPRI supports 8B/10B and 64B/66B encoding options; without loss of generality, our study considers 8B/10B encoding. While CPRI supports topologies such as tree, ring, and chain, each link between RE and REC is a fixed-bandwidth time-division-multiplexed (TDM) connection. A single basic frame duration is 260 ns (1/3.84 MHz), which is compatible with a Universal Mobile Telecommunications System (UMTS) chip length. Each basic frame consists of 16 words, and the word length depends on the CPRI line rate [6]: 256 basic frames make a hyper frame, and 150 hyper frames make a radio frame.

The CPRI radio frame is 10 ms. CPRI line rate information is sent in Z.Y.W.X format between the RE and the REC, where Z is the hyper frame number, Y is a basic frame within a hyper frame, W is the word number within a basic frame, and X is the byte number within a word. CPRI provides auto-rate negotiation, which allows a dynamic reconfiguration of the CPRI line rate based on the antenna and, hence, user traffic characteristics [6].

B. Network Architecture

The considered architecture for Ethernet-based C-RAN and/or V-RAN is shown in Fig. 2, where there are three



Line rate options	Line rate (Mbps)	Bytes per word (X)
option 1	614.4	1
option 2	1228.8	2
option 3	2457.6	4
option 4	3072.0	5
option 5	4915.2	8
option 6	6144.0	10
option 7	9830.4	16
option 8	10137.6	20
option 9	12165.1	24
option 10	24330.2	48

Fig. 1. Frame structure of CPRI.

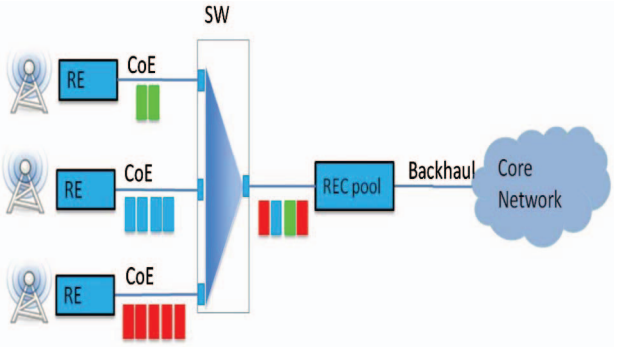


Fig. 2. CPRI-over-Ethernet fronthaul architecture for C-RAN.

links from the RE to the REC supporting CPRI flows packetized over Ethernet. The CoE flows from the RE to the REC pool are switched using an Ethernet switch (SW), where a scheduling policy can be programmed to provide access control to avoid collision.

This architecture can support network sharing between multiple vendors and operators as envisioned for 5G networks [21], which should integrate different wireless standards: 3G, 4G, LTE-advanced, and Wi–Fi. Also, several physical media can be used based on the demand and availability of resources such as fiber, cable, DSL, mm-wave, and free-space optics. The proposed architecture can jointly optimize resources of different media for fronthaul and backhaul (the connection between the REC pool and the core network), as Ethernet can be the underlying protocol for each of these platforms. This facilitates network sharing and common operation and maintenance functions. If large capacity is requested, an optical infrastructure can be utilized to support the RAN [22]. Moreover, thanks to virtualization, multiple operators can share a common physical infrastructure [23].

III. CPRI Over Ethernet (CoE) Mapping

CoE encapsulation requires a mapping between CPRI and Ethernet frames. In this study, we describe a structure-agnostic mapping of CoE, where CPRI flows are sequentially packetized onto an Ethernet frame without the knowledge of CPRI data. Several CoE flows can share a common Ethernet link. Table I shows the notations utilized to describe the mapping between CPRI and Ethernet frames. Figure 3 shows the encapsulation of CPRI flows in the payload of Ethernet frames, considering that the input is at the CPRI line rate and the output is at the Ethernet link rate. The CPRI data is framed into the Ethernet with an additional MAC and RoE header: preamble (7 bytes), start of frame delimiter (1 byte), source address (6 bytes), destination address (6 bytes), Ethernet type (2 bytes), RoE header (6 bytes), frame check sequence (4 bytes), and interpacket gap (12 bytes). As in [4], 6 bytes of the RoE header further contain different subfields such as version, packet type, start of the frame, flow id, timestamp select field, timestamp, and optional extended RoE header space. Note that the optional 802.1Q tag field is not considered

TABLE I
NOTATIONS

Length of Basic CPRI Frame[s]	T_B
Length of Ethernet Frame [bit]	L_E
Encapsulation Delay[s]	T_{encap}
Ethernet Payload Size [bit]	L_P
CPRI Line Bit Rate [bit per second]	R_{CPRI}
Header Overhead per Ethernet Frame[s]	T_{EOH}
Total Ethernet Header Overhead[s]	T_{totHOH}
Ethernet Header Size [bit]	L_{EH}
Number of CPRI Basic Frames	N_B
Number of Ethernet Frames in a Radio Frame	N_E
Total CoE Overhead[s]	T_{totEOH}
Ethernet Rate [bit per second]	R_E
Hop Delay[s]	T_{hop}

in Ethernet overhead calculations. The CPRI data in an Ethernet frame is always a multiple of a CPRI basic frame. The CoE mapping parameters such as encapsulation delay, hop delay, and Ethernet overhead are discussed below.

CoE encapsulation/de-capsulation is assumed to be performed at both the RE and the REC, and the minimum CPRI data to be encapsulated into the Ethernet are one CPRI basic frame of duration $T_B \approx 260$ ns. Thus, Ethernet payload size L_P is computed as

$$L_P = N_B \cdot R_{\text{CPRI}} \cdot T_B. \tag{1}$$

The value of N_B for different line rates is chosen such that the payload value remains close to 1250 or 1500 bytes. L_P is made of multiple N_B, so the number of CPRI basic frames in an Ethernet frame remains an integer value, thus basic frame fragmentation is avoided.

Using L_P, the encapsulation delay T_{encap} is defined as the time taken to frame the CPRI data at a specific line rate into an Ethernet payload (i.e., time to receive the CPRI payload):

$$T_{\text{encap}} = \frac{L_P}{R_{\text{CPRI}}} = N_B \cdot T_B. \tag{2}$$

Total Ethernet header overhead (T_{totHOH}) is the additional delay to transmit Ethernet header (L_{EH}) bytes due to Ethernet encapsulation, and it depends on the total number of Ethernet frames (N_E) utilized to encapsulate the CPRI data, which depend on R_{CPRI} and L_P:

$$T_{\text{totHOH}} = N_E.L_{\text{EH}}/R_E = N_E.T_{\text{EOH}}, \tag{3}$$

where L_{EH} is set to a fixed value (i.e., 44 bytes), R_E is the Ethernet line rate (e.g., line rate of 10G Ethernet: 10 Gbps), T_{EOH} is the header overhead per Ethernet frame.

Hop (i.e., switch, router) delay (T_{hop}) is the delay introduced by the Ethernet switch to process the packet using a store-and-forward mechanism, when the RE and the REC are in a multihop configuration [6]. Hop delay can be estimated based on the switch forwarding functionality, which could be a store-and-forward or cut-through mechanism. This paper considers a worst-case hop delay utilizing a store-and-forward switch. The cut-through switch reduces the hop delay (to 6.4 ns compared with 1 μs for store-and-forward), as only the first 8 bytes are needed to be processed before the switch forwards the Ethernet packet to the respective output port:

$$T_{\text{hop}} = \frac{L_E}{R_E}, \tag{4}$$

where L_E is length of the Ethernet frame ($L_P + L_{\text{EH}}$), expressed in multiples of CPRI basic frame length T_B (see Fig. 3).

From Eqs. (3) and (4), the total CoE overhead (T_{totEOH}) caused by encapsulation of CPRI data on the Ethernet is computed as

$$T_{\text{totEOH}} = T_{\text{totHOH}} + T_{\text{hop}}. \tag{5}$$

Combining Eqs. (3)–(5), we get

$$T_{\text{totEOH}} = N_E.L_{\text{EH}}/R_E + L_E/R_E,$$

where the total Ethernet header overhead (T_{totHOH}) is the additional delay to transmit Ethernet header (L_{EH}) bytes due to Ethernet encapsulation, and it depends on the total number Ethernet frames (N_E) utilized to encapsulate the CPRI data. Hop (i.e., switch, router) delay (T_{hop}) is the delay introduced by the Ethernet switch to process the packet using a store-and-forward mechanism.

Table II shows the computed values of CoE parameters based on Eqs. (1)–(5) when 10G Ethernet is used to send CPRI line rates from option 1 (614.4 Mb/s) to option 6 (6144.0 Mb/s) for two Ethernet payloads sizes, L_P, of 1250 bytes and 1500 bytes. T_{hop} values in Table II are

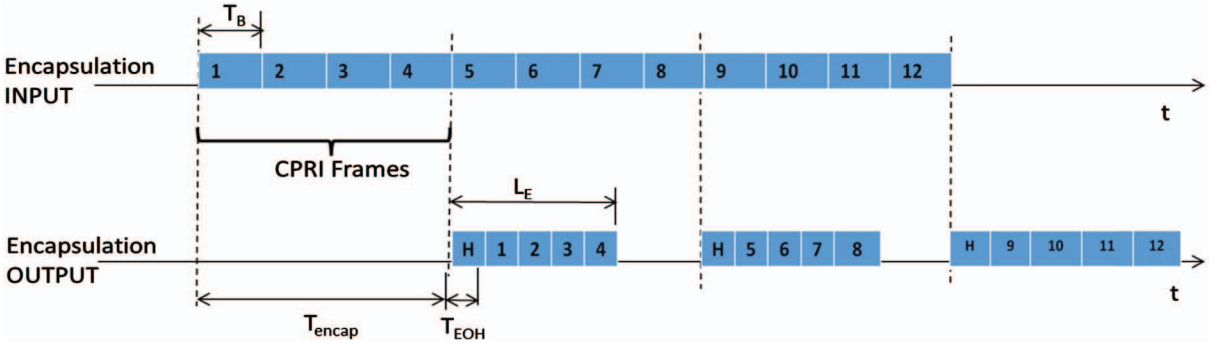


Fig. 3. CPRI encapsulation over Ethernet.

TABLE II
CoE Parameters

Line Rate [Mb/s]	Ethernet Packets per Radio Frame (L_P = 1250 bytes)	T_{encap} [μs]	T_{hop} [μs]	T_{totHOH} (for Radio Frame) [μs]	T_{totHOH} (for Four Subframes) [μs]	$T_{totEOH} * 2$ (Round Trip) [μs]	Distance Supported [km]
614.4 (option 1)	615	16.27	1.00	21.65	8.66	18.32	22.77
1228.8 (option 2)	1229	8.13	1.00	43.26	17.30	35.61	21.04
2457.6 (option 3)	2458	4.06	1.00	86.52	34.61	70.22	17.58
3072.0 (option 4)	3073	3.25	1.00	108.17	43.27	87.54	15.85
4915.2 (option 5)	4916	2.03	1.00	173.04	69.22	139.43	10.66
6144.0 (option 6)	6144	1.62	1.00	216.27	86.51	174.02	7.20
Line Rate [Mb/s]	Ethernet Packets per Radio Frame (L_P = 1500 bytes)	T_{encap} [μs]	T_{hop} [μs]	T_{totHOH} (for Radio Frame) [μs]	T_{totHOH} (for Four Subframes) [μs]	$T_{totEOH} * 2$ (Round Trip) [μs]	Distance Supported [km]
614.4 (option 1)	512	19.53	1.20	18.02	7.21	15.62	23.04
1228.8 (option 2)	1024	9.76	1.20	36.04	14.42	30.04	21.60
2457.6 (option 3)	2048	4.88	1.20	72.09	28.84	58.87	18.71
3072.0 (option 4)	2560	3.90	1.20	90.11	36.04	73.29	17.27
4915.2 (option 5)	4096	2.44	1.20	144.18	57.67	116.54	12.95
6144.0 (option 6)	5120	1.95	1.20	180.22	72.09	145.38	10.06

for a single hop. They are critical to analyze the delay performance of CoE. A LTE radio frame of 10 ms is divided into 10 subframes, each of 1 ms. A LTE eNB should complete eNB processing (uplink CPRI processing, uplink frame decoding, ACK/NACK creation, downlink frame creation, downlink CPRI processing) within 3 ms after receiving uplink data from user equipment (UE) as the HARQ protocol needs an ACK/NACK to be sent in 3 ms for every four LTE subframes. Hence, T_{totHOH} for transmitting four subframes is also shown in Table II. Note that T_{totEOH} is obtained by adding T_{totHOH} for four subframes (4 ms) with single T_{hop}. From [24], the maximum allowed fiber round-trip time is 246 μs after removing RF processing time (40 μs), CPRI processing time (10 μs), REC processing time (2700 μs), and fronthaul equipment processing (4 μs) from 3 ms delay requirement. Thus, the maximum distance supported (km) between the REC and the RE by CoE is given by

$$\text{Distance} = (246\ \mu\text{s} - T_{totEOH})/10\ \mu\text{s/km}, \quad (6)$$

where 10 μs/km is the round-trip fiber propagation delay as the speed of light in fiber is 200,000 km/s. Virtualized RECs can move across different REC pools (hotel of RECs that share cooling and housing resources to save energy) according to traffic/network requirements. This can lead to a situation where fronthaul data traverses different Ethernet switches, leading to a multihop scenario as explained in CPRI [6], where each hop corresponds to an Ethernet switch. Experiments conducted in the next section investigate the scheduling policies to reduce jitter in Ethernet fronthaul.

IV. Jitter Study of CoE

Proper scheduling that minimizes jitter is crucial to achieve acceptable jitter performance on Ethernet fronthaul. An attractive solution to minimize jitter in Ethernet fronthaul is scheduling Ethernet frames by assigning fixed timeslots to send packets of a specific CoE flow [13]. Figure 4(a) shows an example where three CoE flows (1, 2, 3) of rates 5000, 2500, and 1250 Mbps (each of L_E = 1000 bytes), respectively, are multiplexed on an Ethernet interface at 10 Gbps. Scheduling length is defined as the shortest time interval where CoE packets are multiplexed whose pattern repeats periodically; in Fig. 4, scheduling length is denoted by L_S.

The difference in the inter-arrival time between packets is measured as the packet-to-packet jitter [25,26]. The CoE input packets are isochronous, meaning packets arrive at

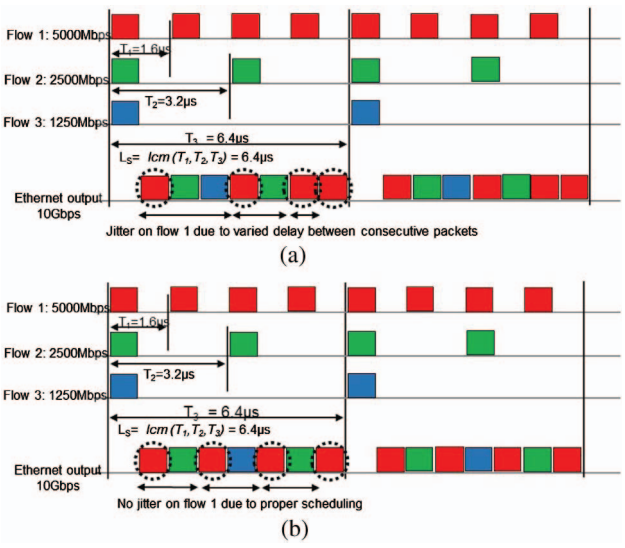


Fig. 4. (a) Example shows jitter on flow 1. (b) Example shows how proper scheduling can eliminate jitter.

the input of the Ethernet switch at regular intervals. Inter-arrival jitter is usually taken as the absolute value of the deviation from its regular state. For evaluating jitter characteristics of fronthaul, we take the worst-case jitter value for all the CoE flows multiplexed, as follows:

$$\begin{aligned} \text{delay}_{i,j} &= \text{arrival time}_{i+1,j} - \text{arrival time}_{i,j}, \\ \text{Jitter}_j &= \max_{\forall\, i}\text{delay}_{i,j} - \min_{\forall\, i}\text{delay}_{i,j}, \\ \text{Jitter} &= \max_{\forall\, j}\text{Jitter}_j, \end{aligned} \tag{7}$$

where $\text{delay}_{i,j}$ denotes delay at the receiver of the REC for packet number i in flow j. From Eq. (7) the worst-case jitter for all flows (i.e., max of max) is taken as a quality metric of the schedule. For this example, jitter on flow 1 is the difference between the highest inter-packet delay, i.e., 2.4 μs, and the lowest inter-packet delay, i.e., 0.8 μs, which is 1.6 μs. However, better scheduling can be done that completely removes jitter, as shown in Fig. 4(b), where the jitter is zero because there is no variance in inter-packet delay for packets of the same flow. In this section, we propose a scheduling policy to multiplex several CoE flows on Ethernet such that the jitter of CoE remains within an acceptable level [see Eq. (7)].

This scheduling policy can be programmed in the Ethernet switch, as shown in Fig. 2, where multiplexing occurs. CoE flows from several REs need to be scheduled at precise times to provide least-delay variance and, hence, tolerable jitter. Scheduling Ethernet requires strict (and periodic) time schedules (on/off slots) where each CPRI flow's packets can be transmitted. The schedule is formed using parameters such as the queue schedule of nodes, transmission delays, packet lengths, and CoE rates.

A conflicting schedule is defined as one that schedules more than one packet of different or the same flows at the same time. Finding a non-conflicting schedule of packets is proven to be NP-complete. References [27] and [28] prove that the problem of producing a non-conflicting schedule to multiplex multiple flows can be reduced to the classical graph-coloring problem, which is known to be NP-complete. There are several algorithms proposed in other network problems that strive to produce a non-conflicting schedule of multiplexed packets [27,28]; node-based scheduling and level-based scheduling are popular ones. But for fronthaul, where topologies are not as complex as other networks and jitter is much more stringent, a greedy approach that exhaustively searches the minimum jitter sequence can be a good choice. Below, we propose a greedy scheduling algorithm that minimizes jitter by proper scheduling, and then we compare it with a benchmark algorithm. We assume that all the flows in the proposed fronthaul network are CoE flows whose characteristics such as packet lengths and CPRI rates are well-determined. We also assume that the network is not oversubscribed, and there is only one switch that is multiplexing multiple CPRI flows onto Ethernet output using a tree topology. If there are multiple switches aggregating flows in the network, a combined schedule needs to be formed using global information with the help of a software defined network controller and pushed into each of the switches.

When multiple input ports get aggregated into an output port, there is an internal serialization delay in the switch known as the M:1 delay. The objective of the proposed scheduling policies is to decrease the maximum jitter among all the flows, as defined in the paper, thus providing the frames with a fair amount of serialization delay.

A. CoE Scheduling Policies

This section introduces the proposed comb-fitting (C-FIT) algorithm that schedules flows in Ethernet to reduce jitter. The basic-offset algorithm provides an initial configuration to be used by C-FIT, and the first available timeslot (FAT) serves as a benchmark algorithm. Table III shows several parameters that are utilized in the pseudo code of these algorithms. The basic-offset algorithm (Algorithm 1) schedules CoE packets such that jitter is temporarily zero (ideal case) without taking into consideration that the obtained solution can contain scheduling conflicts (i.e., multiple packets can be scheduled at the same time). CoE packets are offset by multiples of Ethernet timeslot sizes T_{ETS} for each flow.

C-FIT takes the schedule produced by the basic-offset algorithm as input and resolves conflicts. All possible permutations of input flow orders are formed because the order in which any two flows are combined using the matcombine subroutine affects the final jitter. For example, if three flows are considered, then the possible flow orders are $\{1 > 2 > 3\}$, $\{1 > 3 > 2\}$, $\{2 > 1 > 3\}$, $\{2 > 3 > 1\}$, $\{3 > 1 > 2\}$, and $\{3 > 2 > 1\}$. For each order, the flows are sequentially combined using matcombine, which takes two flows' schedules as input and produces a non-conflicting schedule according to this procedure: the flow with the higher number of packets is kept intact, and the other flow is offset by a multiple of L_{ETS} to produce a non-conflicting schedule (this is called a sliding approach). If such a non-conflicting sequence is not achieved by using the sliding approach, conflicting packets are moved to the nearest timeslot that is unoccupied. This approach is followed for all possible flow orders, and the schedule with the lowest amount of jitter is selected as the final schedule.

The proposed C-FIT algorithm is compared with the FAT algorithm, namely, benchmark FAT, which resolves the conflicts produced with the basic-offset algorithm by moving the conflicting packets to the first available timeslot that can accommodate the packet without using a sliding

TABLE III
PARAMETERS FOR COE PACKET SCHEDULING

Input CoE rate for flow i [bit per second]	R^i_{COE}
Ethernet rate [bit per second]	R_E
Transmission time of flow i packet on Ethernet link [s]	T^i_P
Ethernet timeslot size [s]	T_{ETS}
# of slots in a schedule length for flow i	L^i_{SF}
Schedule of CoE flow i on Ethernet link [time vector]	comb^i
Number of flows	N_F
Total slots in scheduling length	N_S
Schedule length [s]	L_S

approach and flow ordering. Algorithm 3 shows the pseudo code for benchmark FAT.

For the input flows in Fig. 4, we provide a non-conflicting scheduling sequence as an example using a benchmark FAT algorithm here. Let us assume that packets arrive at an Ethernet switch by time 0. The basic-offset algorithm assigns the periodic timeslots to the flows 1, 2, and 3: specifically, it assigns to flow 1 timeslots starting at {0, 1.6, 3.2, 4.8} μs, for flow 2 at {0, 3.2} μs, and for flow 3 at {0} μs. However, this solution results in two conflicts, i.e., at times {0, 3.2} μs. These conflicts are resolved by the benchmark FAT algorithm, by allocating available unallocated slots to flows involved in each of the conflicts. Flow 1 has a schedule of {0, 1.6, 3.2, 4.8} μs; however, by the end of the first timeslot, packets from flows 2 and 3 also arrive at the Ethernet switch, which results in a conflict. Hence, packets from flow 1 remain unaffected, whereas the other packets get the next unallocated timeslots. The same procedure is also applied at timeslot 3.2 μs, thus resulting in the sequence {0, 1.6, 3.2, 4.8} μs, {0.8, 4.0} μs, and {2.4} μs for the three flows, respectively.

For the same input scenario, C-FIT produces several schedules using different flow orders. For the flow order {1 > 2 > 3}, the schedule is {0, 1.6, 3.2, 4.8} μs for flow 1, {0.8, 4.0} μs for flow 2, and {2.4} μs for flow 3, the same as the FAT algorithm.

For flow order {1 > 3 > 2}, the schedule is {0, 1.6, 3.2, 4.8} μs for flow 1, {2.4, 5.6} μs for flow 2, and {0.8} μs for flow 3.

For flow order {2 > 3 > 1}, the schedule is {0, 1.6, 3.2, 4.8} μs for flow 1, {0.8, 4.0} μs for flow 2, and {2.4} μs for flow 3.

For flow order {2 > 1 > 3}, the schedule is {0, 1.6, 3.2, 4.8} μs for flow 1, {0.8, 4.0} μs for flow 2, and {2.4} μs for flow 3.

For flow order {3 > 1 > 2}, the schedule is {0, 1.6, 3.2, 4.8} μs for flow 1, {2.4, 5.6} μs for flow 2, and {0.8} μs for flow 3.

For flow order {3 > 2 > 1}, the schedule is {0, 1.6, 3.2, 4.8} μs for flow 1, {0.8, 4.0} μs for flow 2, and {2.4} μs for flow 3.

All the flow orders in this scenario produced zero jitter, so any schedule can be selected. Because the C-FIT approach considers all possible flow orders, the complexity gets exponential. One way to reduce complexity is to stop running the algorithm as soon as a flow order produces zero jitter. Because the fronthaul consists of a limited number of flows multiplexed at the Ethernet switch, the complexity is not a big concern.

Algorithm 1 Basic-Offset Algorithm

Input: R^i_{COE}, R_E, L_E (assume flows are synchronized at input)
Output: Schedule of CoE flows on Ethernet output
Step 1: Calculate transmission time for flow i packet on incoming Ethernet link as $T^i_P = \frac{L_E}{R^i_{COE}}$
Calculate scheduling length L_s as lowest common multiple of T^i_P, i.e., $L_S = lcm(T^i_P)$
Calculate outgoing Ethernet timeslot size $T_{ETS} = \frac{L_E}{R_E}$
Calculate number of timeslots in L_S for flow i, $L^i_{SF} = \frac{L_S}{T^i_P}$
Step 2: initialize: offset_nf = 0;
//starting offset value of next flow is set to zero
for i = 1 to N_F
mat = comb^i;
/* mat is 2D temporary matrix that holds the contents of comb^i */
offset = 0;
for j = 1 to L^i_{SF}
in a certain flow
mat(j, 1) = offset + offset_nf;
// mat(j, 1) represents start time of packet j
mat(j, 2) = mat(j, 1) + T_{ETS};
// mat(j, 2) represents end time of packet j
offset = offset + T^i_P;
end
offset_nf = offset_nf + T_{ETS};
comb^i = mat;
/* comb^i is 2D matrix that holds start and end time of each packet in flow i*/
end

Algorithm 2 Comb Fitting (C-FIT)

Input: Schedule of CoE flows given by basic-offset algorithm
Output: Non-conflicting schedule of CoE packets
Step 1: Form all possible permutations flow orders from 1 to N_F (N_F! different sequences) denoted by SEQ^m_{PF}, where m = 1 to N_F!
Step 2: for each sequence $SEQ^m_{PF} \in \{SEQ^m_{PF}\}$
for j in SEQ^m_{PF}
initialize: matcomb = first element in$\{SEQ^m_{PF}\}$
matcomb = **matcombine** (comb^j, matcomb)
end
Calculate jitter matcomb
end
Pick matcomb with least amount of jitter

MATCOMBINE SUBROUTINE

Input: comb^i, comb^j (any two schedules)
Output: Combined non-conflicting schedule
Step 1: Initialize: matcomb as a matrix with length as sum lengths of comb^i, comb^j
Step 2: Take longest sequence out of comb^i, comb^j, and add its contents to matcomb, call the other matrix mattemp
Step 3: Shift mattemp by multiples of T_{ETS} to form a perfect non-conflicting schedule with matcomb
Step 4: if success in this procedure
Copy mattemp to matcomb and **return** matcomb
Step 5: else
Copy non-conflicting packets of mattemp to matcomb
for all conflicting packets in mattemp
Find nearest open timeslot that can fit the packet and update matcomb
end
return matcomb
end

Algorithm 3 First Available Timeslot (FAT) for Benchmark

Input: Schedule of flows given by basic-offset algorithm
Output: Non-conflicting schedule of CoE packets
Step 1: for i = 1 to N_F
initialize: fatcomb = $comb^i$;
// fatcomb is temporary matrix that holds contents of $comb^j$
fatcomb = **fatcombine** ($comb^i$, fatcomb)
end
Calculate jitter for fatcomb

FATCOMBINE SUBROUTINE

Input: $comb^i$, $comb^j$ (any two schedules)
Output: Combined non-conflicting schedule
Step 1: Initialize: fatcomb as a matrix with length as sum lengths of $comb^i$, $comb^j$
Step 2: Take longest sequence out of $comb^i$, $comb^j$, and add its contents to fatcomb, call the other matrix mattemp
Step 3: Copy non-conflicting packets of fattemp to fatcomb
for all conflicting elements in fattemp
Find nearest open timeslot that can fit the packet and update fatcomb
end
return fatcomb

V. Performance Evaluation and Results

This section presents evaluation of CoE performance metrics such as delay and jitter in the Ethernet fronthaul obtained through Verilog pre-synthesis experiments and simulations.

A. Ethernet Encapsulation Delay

We present results on the impact of CPRI line rates and payload size on the Ethernet encapsulation delay of CoE flows. We also show how the delay affects the fronthaul distance. An FPGA pre-synthesis verification is performed to map CoE and analyze delay performance of the multihop mobile fronthaul [6]. Pseudo random binary sequence data is generated and encapsulated in the Ethernet frame using Verilog hardware description language (HDL) and evaluated utilizing ModelSim as a HDL simulator.

The Ethernet header consists of 24 bytes, which contains layer-2 Ethernet MAC header fields such as source address (6 bytes), destination address (6 bytes), Ethernet type (2 bytes), RoE header (6 bytes), frame check sequence (4 bytes). The generated data is framed at a clock rate (i.e., a clock cycle takes 6.4 ns) of 10 Gbps. Thus, generating Ethernet header overhead of 24 bytes requires three clock cycles, which is 19.2 ns (as shown in Fig. 5). The experiments are conducted for three payload sizes—500, 1000, and 1500 bytes—to study the effect of payload size on the encapsulation delay.

Figure 5 verifies that Ethernet encapsulation is successfully designed and implemented in Verilog HDL. The left

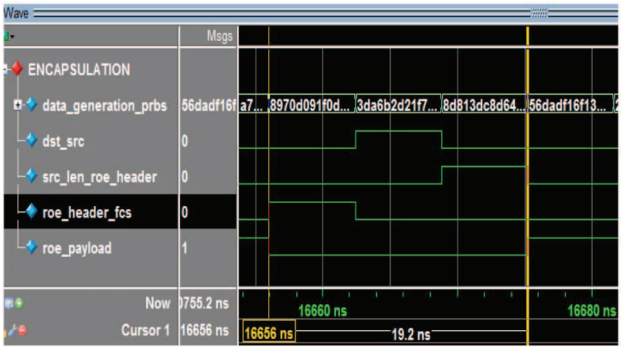


Fig. 5. FPGA pre-synthesis simulation of CoE encapsulation.

side of the waveform shows the labels of generated data, header fields, Ethernet frame (encapsulating the generated data), and inter-frame gap fields, while the right side shows the corresponding timing information. *data_generation_prbs* shows the generated data in pseudo random form. *dst_src* in the pre-synthesis evaluation shows the first part of the header containing the destination and part of the source fields. The *src_len_roe_header* field shows the remaining part of the source field, the Ethernet type, and the first part of the *roe* header. The *roe_header_fcs* field shows the remaining part of the *roe* header and the frame check sequence (FCS). The *roe_payload* field indicates IQ samples encapsulated in the Ethernet payload. Note that the data is generated continually at all times including header generation time. The markers show layer-2 Ethernet MAC overhead for a single Ethernet frame as 19.2 ns. For 615 Ethernet frames, the delay would be 19.2 × 615 ns = 11.8 μs, as shown in Table II (for CPRI option 1).

Figure 6 shows the encapsulation delay (T_{encap}) as a function of CPRI line rate (R_{CPRI}) options with different Ethernet packet payload P_E sizes (500, 1000, and 1500 bytes). As expected, the encapsulation delay decreases as R_{CPRI} increases because higher R_{CPRI} flow takes shorter time to fill up the Ethernet packet payload size. Moreover, the encapsulation delay decreases as L_E size decreases at the given R_{CPRI} option, as lower payload gets filled in lesser time. Figure 7 shows the distance of fronthaul (based on latency constraints) calculated using Eq. (6), for different

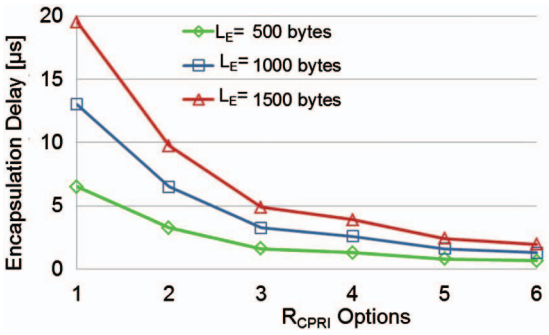


Fig. 6. Encapsulation delay with different R_{CPRI} options and different L_E sizes.

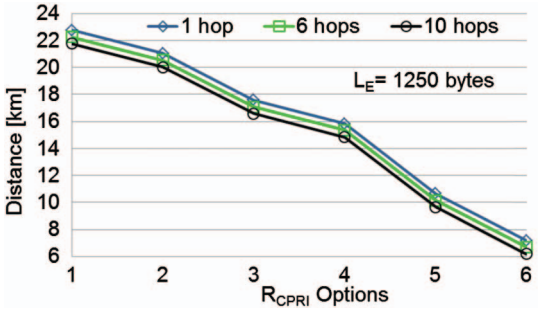


Fig. 7. Fronthaul distance supported in multihop scenario with $L_E = 1250$ bytes.

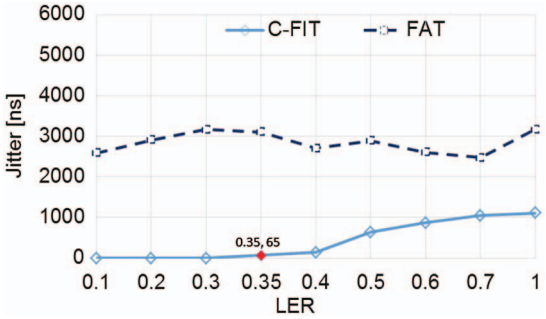


Fig. 9. Jitter versus load to Ethernet ratio for number of flows 2 to 5 and line rates 1 to 9.

CPRI line rates (options 1 to 6) and for the number of Ethernet switches the packets cross for the Ethernet payload of 1250 bytes. It shows that, as the number of hops increases (more Ethernet switches crossed), the distance of fronthaul decreases due to added delay from the Ethernet switch using a store-and-forward mechanism.

The measured values of switch delay are in concurrence with the calculated values (hop delay) as in Table I. Higher line rates also support lower fronthaul distances due to a larger number of Ethernet packets that need to be generated for a single 10 ms radio frame. The distances supported using CoE are good for access and metro network coverage distances. Figure 8 shows that a larger Ethernet payload leads to lower CoE overhead, thus supporting longer fronthaul distance.

Hence, CoE can be implemented for different CPRI line rates by compromising a few kilometers in the fronthaul (i.e., from a minimum of 1 km to a maximum of 10 km as a function of the CPRI line rate, and the Ethernet packet payload size, as shown in Figs. 7 and 8).

B. Jitter for CoE on Scheduled Ethernet

This section evaluates the performance of the C-FIT algorithm and compares it with the benchmark FAT algorithm. An event-driven simulator built in-house in MATLAB is used to evaluate both algorithms. The following indexes are considered to evaluate the performance of the algorithms: The load to Ethernet ratio (LER) is defined as the ratio between the sum of input CoE flow rates and the Ethernet rate and jitter measured at the REC. Over 5000 random combinations of input rates are generated and scheduled over Ethernet, and the experiment is repeated 10 times. The value of jitter for particular LER is averaged and plotted. Each plot (Figs. 9–12) simulates random CoE rates derived by encapsulating CPRI flows from line rates uniformly selected from 1 to x ($x \leq 9$). Uniformly distributed random numbers of flows are multiplexed on an Ethernet link.

Figures 9 and 10 show the effect of increasing line rates while keeping the number of flows in a constant range, whereas Figs. 11 and 12 show the effect of number of flows multiplexed while keeping the range of line rates constant. Figure 9 shows the value of jitter versus increasing LER for the C-FIT algorithm compared with the benchmark FAT algorithm. The CoE rates are uniformly picked from option 1 to option 9, and the number of flows multiplexed are randomly picked from 2 to 5. It can be seen that jitter for C-FIT remains zero until a load ratio of 0.3 and then increases. We call the LER until jitter remains zero and then increases to a nonzero value as the inflection point; hence, 0.3 is the inflection point in this case. The marked 0.35 LER value shows the maximum allowed jitter of 65 ns. The trend of jitter versus LER is not only dependent on the LER but also on the periodicity of the flows because more flows that are multiples of each other can form jitter-free schedules without resulting in conflicts. However, if the LER is low, there is enough room for the flows to fit in perfectly without conflicts, hence leading to zero jitter.

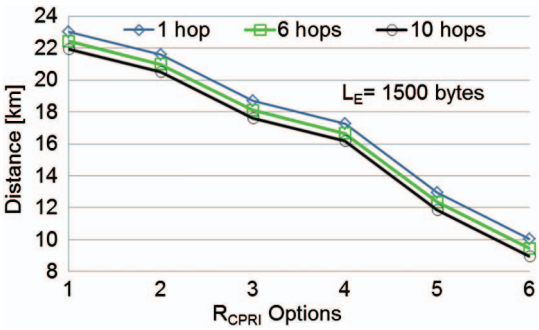


Fig. 8. Fronthaul distance supported in multihop scenario with $L_E = 1500$ bytes.

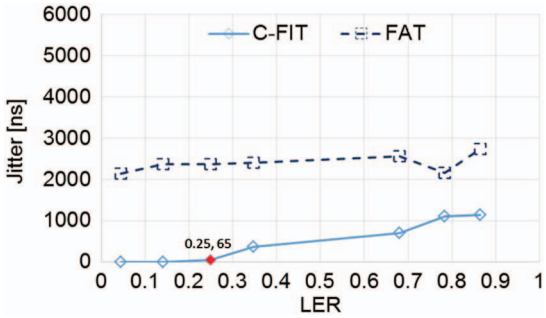


Fig. 10. Jitter versus load to Ethernet ratio for number of flows 2 to 5 and line rates 5 to 9.

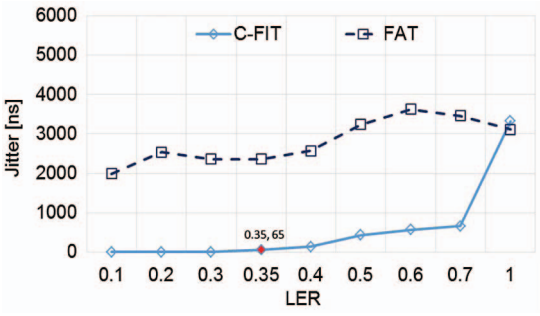


Fig. 11. Jitter versus load to Ethernet ratio for number of flows 1 to 3 and line rates 1 to 9.

Figure 10 shows jitter versus load ratio for CoE rates with a range from option 5 to option 9 and number of flows from 2 to 5. We see that the trend is similar until a LER of 0.2 (inflection point) and then jitter increases. Jitter is below 65 ns until a LER of 0.25 for C-FIT. Red dots in Figs. 9–12 represent a 65 ns inflection point. C-FIT shows a monotonically increasing jitter behavior, but the benchmark FAT does not. This is because C-FIT explores all possible flow orders and takes the least jitter schedule, but the benchmark FAT considers only the given input sequence, which can largely affect the jitter values, making the relation with the LER not as pronounced as expected.

Figures 11 and 12 show the effect of number of flows multiplexed on the Ethernet switch. Figure 11 has one to three flows multiplexed on Ethernet. We see that the inflection point is 0.3 and jitter is acceptable until a LER of 0.35. Figure 12 shows four to six flows multiplexed on Ethernet. The inflection point is 0.3 with acceptable jitter until a LER of 0.36, and there is a monotonic increase in the jitter with higher load ratios. We see from Figs. 11 and 12 that jitter using the FAT algorithm is higher when a higher number of flows (i.e., four to six) is multiplexed. This is because, as the fronthaul topology gets larger (more number of flows), the number of conflicts increases.

Although results indicate that only 30% of the average amount of Ethernet bandwidth can be used if we want to satisfy the jitter requirement to multiplex CoE flows, there are certain flow combinations that can achieve very low jitter with much higher Ethernet utilization. In fact, input flows that are multiples of each other are found to achieve higher Ethernet utilization while guaranteeing tolerable jitter of 65 ns. Moreover, it is worth noting that the redundant Ethernet capacity could be utilized to send other non-time-sensitive data, as in [29] where fronthaul, midhaul, and backhaul can share Ethernet capacity.

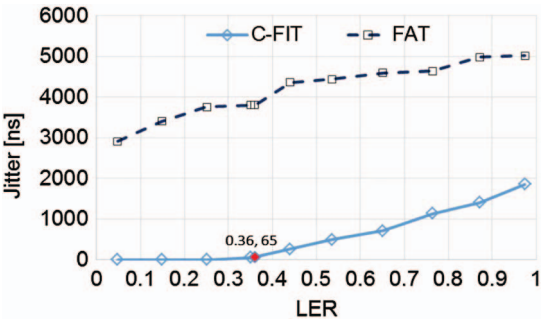


Fig. 12. Jitter versus load to Ethernet ratio for number of flows 4 to 6 and line rates 1 to 9.

The encapsulation delay of CPRI packets decreases as the line rate increases, as shown in Table II; however, jitter increases as the LER increases. Hence, there is a need to make a careful choice of Ethernet rate for a given CoE topology (rates).

VI. Conclusion

Ethernet fronthaul is expected to provide many benefits to mobile networks such as 5G. CPRI over Ethernet (CoE) can be a cost-efficient alternative to CPRI fronthaul, as Ethernet is easily available and can be a stepping stone to many useful applications. In this study, we showed that CoE encapsulation and switching introduces a slight delay that can compromise a few kilometers in the multihop mobile fronthaul. Moreover, jitter was studied in terms of LER, number of flows, and flow combination. For a given topology, a scheduling method that completely eliminates jitter can be provided by using a certain Ethernet rate with the proposed comb-fitting scheduling. Proposed C-FIT scheduling performs considerably well compared with the benchmark first-available-timeslot algorithm. In particular, jitter reduction as big as units of microseconds can be achieved.

References

[1] Cisco Visual Networking Index, 2015 [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html. Accessed Oct. 31, 2016.

[2] V. Jungnickel, K. Habel, M. Parker, S. Walker, C. Bock, J. Ferrer Riera, V. Marques, and D. Levi, "Software-defined open architecture for front-and-backhaul in 5G mobile networks," in *Proc. 16th Int. Conf. on Transparent Optical Networks (ICTON)*, July 2014.

[3] A. Checko, H. Christiansen, H. Yan, Y. Scolari, G. Kardaras, M. Berger, and L. Dittmann, "Cloud RAN for mobile networks—A technology overview," *IEEE Commun. Surv. Tutorials*, vol. 17, pp. 405–426, 2015.

[4] Next Generation Fronthaul Interface (1914) Working Group [Online]. Available: http://sites.ieee.org/sagroups-1914/. Accessed Oct. 31, 2016.

[5] "Next Generation Fronthaul Interface," China Mobile White Paper, 2015 [Online]. Available: http://labs.chinamobile.com/cran/wp-content/uploads/White Paper of Next Generation Fronthaul Interface.PDF. Accessed Oct. 31, 2016.

[6] "Common Public Radio Interface (CPRI) Specification V7," 2015 [Online]. Available: http://www.cpri.info/downloads/CPRI_v_7_0_2015-10-09.pdf. Accessed Oct. 31, 2016.

[7] Open Base Station Architecture Initiative (OBSAI), 2013 [Online]. Available: http://www.obsai.com/specifications.htm. Accessed Oct. 31, 2016.

[8] Open Radio equipment Interface (ORI), 2014 [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/ori. Accessed Oct. 31, 2016.

[9] G. Britton, B. Kubert, and J. Chapin, "RF over Ethernet for wireless infrastructure," in *Proc. Software Defined Radio Technical Conf.*, Nov. 2005.

[10] Fujitsu Network Communications, "The benefits of cloud-RAN architecture in mobile network expansion," 2014 [Online]. Available: http://www.fujitsu.com/downloads/TEL/fnc/whitepapers/CloudRANwp.pdf. Accessed Oct. 31, 2016.

[11] J. Huang, Ed., "Integrated mobile fronthaul and backhaul," 2016 [Online]. Available: https://tools.ietf.org/pdf/draft-huang-detnet-xhaul-00.pdf. Accessed Oct. 31, 2016.

[12] J. Farkas and B. Varga, "Applicability of Qbu and Qbv to fronthaul," Ericson White Paper, 2015 [Online]. Available: http://www.ieee802.org/1/files/public/docs2015/cm-farkas-applicability-of-bu-and-bv-1115-v02.pdf. Accessed Oct. 31, 2016.

[13] T. Wan and P. Ashwood, "A performance study of CPRI over Ethernet," IEEE 1904.3, 2015 [Online]. Available: http://www.ieee1904.org/3/meeting_archive/2015/02/tf3_1502_ashwood_1a.pdf. Accessed Oct. 31, 2016.

[14] N. Gomes, P. Chanclou, P. Turnbull, A. Magee, and V. Jungnickel, "Fronthaul evolution: From CPRI to Ethernet," *Opt. Fiber Technol.*, vol. 26, pp. 50–58, 2015.

[15] L. Valcarenghi, K. Kondepu, and P. Castoldi, "Analytical and experimental evaluation of CPRI over Ethernet dynamic rate reconfiguration," in *Proc. IEEE Int. Conf. on Communications (ICC)*, May 2016.

[16] D. Chitimalla, K. Kondepu, L. Valcarenghi, and B. Mukherjee, "Reconfigurable and efficient fronthaul of 5G systems," in *IEEE Int. Conf. on Advanced Networks and Telecommuncations Systems (ANTS)*, Dec. 2015.

[17] "IEEE P802.1Qbu—Bridges and bridged networks—Amendment: Enhancements for frame preemption," July 2015 [Online login required]. Available: http://www.ieee802.org/1/files/private/bu-drafts/d3/802-1Qbu-d3-0.pdf. Accessed Oct. 31, 2016.

[18] "IEEE P802.1Qbv—Bridges and bridged networks—Amendment: Enhancements for scheduled traffic, Sept. 2015 [Online login required]. Available: http://www.ieee802.org/1/files/private/bv-drafts/d3/802-1Qbv-d3-1.pdf. Accessed Oct. 31, 2016.

[19] "A performance study of CPRI over Ethernet," *IEEE 1904.3 Working Group Meeting*, Jan. 2015.

[20] T. Wan and P. Ashwood-Smith, "A performance study of CPRI over Ethernet with IEEE 802.1Qbu and 802.1Qbv enhancements," in *IEEE GLOBECOM*, Dec. 2015.

[21] M. Fiorani, B. Skubic, J. Mårtensson, L. Valcarenghi, P. Castoldi, L. Wosinska, and P. Monti, "On the design of 5G transport networks," *Photon. Netw. Commun.*, vol. 30, pp. 403–415, 2015.

[22] A. Asensio, P. Saengudomlert, M. Ruiz, and L. Velasco, "Study of the centralization level of optical network-supported cloud RAN," in *Optical Network Design and Modeling (ONDM)*, May 2016.

[23] H. Ali-Ahmad, C. Cicconetti, A. Oliva, V. Mancuso, M. Sama, P. Seite, and S. Shanmugalingam, "An SDN-based network architecture for extremely dense wireless networks," in *IEEE SDN for Future Networks and Services*, Nov. 2013, pp. 105–108.

[24] H. Son and S. Shin, "Fronthaul size: Calculation of maximum distance between RRH and BBU," Apr. 2014 [Online]. Available: http://www.netmanias.com/en/post/blog/6276/c-ran-fronthaul-lte/fronthaul-size-calculation-of-maximum-distance-between-rrh-and-bbu. Accessed Oct. 31, 2016.

[25] "Jitter measurement for voice over IP," Voipfuture White Paper, 2009 [Online]. Available: http://www.voipfuture.com/wordpress/wp-content/uploads/2015/07/Voipfuture_WP_Jitter-Measurement-VoIP-Quality-Monitoring-Basics.pdf. Accessed Oct. 31, 2016.

[26] J. Anuskiewicz, "Measuring jitter accuratley," 2008 [Online]. Available: http://www.lightwaveonline.com/articles/2008/04/measuring-jitter-accurately-54886317.html. Accessed Oct. 31, 2016.

[27] E. Sinem and V. Pravin, "TDMA scheduling algorithms for wireless sensor networks," *Wireless Netw.*, vol. 16, pp. 985–997, 2010.

[28] J. Mao, Z. Wu, and X. Wu, "A TDMA scheduling scheme for many-to-one communications in wireless sensor networks," *Comput. Commun.*, vol. 30, pp. 863–872, 2007.

[29] J. Korhonen, "Practical approach to converged FH/BH network architecture and functional partitioning," IEEE 1914.1 TF, 2016 [Online]. Available: http://sites.ieee.org/sagroups-1914/files/2016/08/tf1_1608_korhonen_practical_approach_2.pdf. Accessed Oct. 31, 2016.