A Performance Study of CPRI over Ethernet

Tao Wan and Peter Ashwood

Huawei Canada Research Center

{tao.wan, peter.ashwoodsmith}@huawei.com

January 30, 2015

Abstract

There has been a debate on whether or not Ethernet, a highly cost effective technology, could meet the stringent latency and jitter requirements imposed by CPRI. To facilitate the discussion, we conducted simulations to understand how Ethernet performs when carrying CPRI traffic, with and without newly proposed Ethernet enhancements, namely Ethernet with preemption and Ethernet with scheduled traffic.

Our simulation results led to three observations: 1) Ethernet alone, with or without newly proposed frame preemption enhancement, could not meet the jitter requirement of $65ns$; 2) the newly proposed Ethernet enhancement of scheduled traffic could lower or even completely remove jitter in most cases, but not always. Further, scheduled traffic appears intrusive, thus requires further careful examination. 3) Ethernet with frame preemption, combined with some jitter reduction methods such as edge based buffering, could potentially meet jitter requirements of CPRI.

Key words. CPRI, Ethernet, Frame Preemption, Scheduled Traffic, Simulation

1 Introduction

Common Public Radio Interface (CPRI) is a protocol used between a Remote Radio Header (RRH) and a Base Band Unit (BBU) for carrying digital signals. Traditionally, BBU is physically co-located with RRH inside a base station, significantly limiting the locations where base stations could be deployed. To improve deployment flexibility and operation efficiency, a new mobile network architecture, namely Cloud Radio Access Network (C-RAN) [9], has been proposed to have BBUs separated from RRHs and implemented in a centralized location such as a data center. Advantages of the C-RAN architecture include, but are not limited to, flexible deployment of the mobile network, significant economic benefits on both CAPEX and OPEX, improved energy efficiency, elasticity in meeting capacity demand, reduced power consumption, and improved physical security of BBUs [9].

In C-RAN architecture, there is a transport network (Fronthaul network) that carries signals from RRHs in remote sites to BBUs implemented in a central location. This transport network must meet the stringent performance requirements imposed by CPRI [1], including 1) 100ns of one way delay; 2) 65ns of maximum variation in delay (i.e., jitter); 3) up to 10Gbps of throughput per RRH; and 4) 10^{-12} of maximum bit error rate. Consequently, C-RAN architecture proposes the use of optical transport networks for carrying digitalized

radio in real time. Options for CPRI optical transport include *dedicated fiber*, *optical transport network (OTN)*, *passive optical network (PON)*, and *wavelength division multiplexing (WDM)* [8].

Recently, there has been a debate on whether or not, Ethernet networks, could be used to carry CPRI traffic. On one hand, Ethernet is highly attractive due to its ubiquitous deployment and low costs. On the other hand, it is widely believed that Ethernet, a shared and best effort based technology, could not meet the performance requirements, particularly the low jitter requirement. For those who are in favor of CPRI over Ethernet (CoE), they often suggested the use of Ethernet networks with dedicated links between RRH and BBU, enhanced with additional features such as integrated monitoring features to satisfy the stringent latency and jitter constraints [8].

On a separate activity in IEEE, two new enhancements, namely Frame Preemption [4] and Scheduled Traffic [5], have been proposed to allow Ethernet to carry time sensitive traffic. Frame Preemption (namely *Ethernet with preemption* thereafter) allows a frame with lower priority to be preempted by another frame with higher priority. A same frame could be preempted more than once. Scheduled Traffic (namely *Ethernet with scheduling* thereafter) allows frames to be transmitted at precise times according to a schedule. While these two Ethernet enhancements are not proposed to directly address the performance requirements of CPRI over Ethernet, they are certainly relevant to CoE. Thus a natural question would be, could Ethernet with those new enhancements meet CPRI performance requirements? In this paper, we studied the performance of CPRI over Ethernet to answer the following questions:

1. What are delays and jitters of CPRI traffic over standard Ethernet?
2. What are delays and jitters of CPRI traffic over Ethernet with frame preemption ?
3. What are delays and jitters of CPRI traffic over Ethernet with Scheduled Traffic ?
4. What other solutions could work with Ethernet to reduce latency and jitter ?

To answer these questions, we conduct extensive simulations of CPRI traffic over Ethernet using open source Network Simulator (NS3) [2]. We use UDP packets with fixed transmission intervals to represent CPRI traffic with constant rates of 2.45, 4.91, and 9.83Gbps respectively. The DropTail queue management mechanism is used to measure the delay and jitter for standard Ethernet without any newly proposed enhancement (**Question 1**).

To simulate the behavior of Ethernet with frame preemption (**Question 2**), we injected background packets with length of 64 or 128 bytes. The time taken to preempt a low priority packet is assumed to be capped by the time taken to process a packet of 128 bytes. In other words, the delays caused by packets of 64 and 128 bytes respectively represent the normal and worst cases in preempting low priority traffic by CPRI traffic.

To simulate the behavior of Ethernet with scheduled traffic (**Question 3**), we implemented a traffic scheduling algorithm in NS3. Our scheduling algorithm maintains a scheduling table for each output port. Each scheduling table represents a fixed time window, which

is further divided into a set of fixed time slots. Each slot is either associated with a flow (in open state) or not associated with any flow (closed state). A slot in open state is used to transmit packets in the associated flow, and a slot in closed state is not used to transmit any packet. This implementation allows us to study the behavior of the proposed scheduled traffic enhancement. We also implemented an edge based buffering mechanism to see if it could help reduce or remove jitter completely (**Question 4**). Our simulation results indicate buffering at the edge appears promising.

Here is a summary of the simulation results based on a network topology of 15 Ethernet switches (cf 3.1), representing a Fronthaul mobile network. For standard Ethernet, both delay and jitter are relatively stable in all testing cases. For example, the delay is about $90us$, and the jitter is $400ns$ in the worst case. Ethernet with preemption also performs consistently in all testing cases, with the delay of $91us$ and the jitter of $410ns$ in the worst case. However, Ethernet with scheduling does not seem to perform consistently. In most cases, the jitter is completely removed , while in a few cases, it is increased to $1000ns$. Delay appears to be consistent, with the amount of $95us$ in the worst case. Based on those results, we suggest that Ethernet could meet the one-way delay requirement of $100us$, but could not meet the jitter requirement of $65ns$, with or without any proposed new enhancement.

To this end, we proposed the use of buffering at the edge to further reduce jitter. Based on preliminary testing results, this approach appears promising since the jitter could be completely removed, and it performs consistently. To summarize, we made three observations based on our simulation results :

1. Ethernet alone, with or without newly proposed preemption, could not meet the jitter requirement of $65ns$ in maximum.

2. Ethernet with scheduling appears intrusive and require advanced scheduling algorithms to become practical for carrying CPRI traffic.

3. Ethernet with preemption, combined with buffering at the edge, could potentially meet the jitter requirement, thus is recommended.

The rest of the paper is organized as follows. Section 2 describes a scheduling algorithm used for measuring the behavior of Ethernet with scheduling. In Section 3, we discuss the setup of our simulations, including the network topologies used in the study. Simulation results are presented in Section 4. We conclude the paper in Section 5.

2 A Scheduling Algorithm

Scheduling algorithms could be classified into *global scheduling* and *local scheduling*. In global scheduling, a centralized computing engine (e.g., an SDN controller) computes a queue schedule for each network node, by taking into the consideration of a number of parameters, including but not limited to, network topology, queue schedule for predecessor nodes, and transmission delays. A local scheduling is computed locally and independently

f_i	flow i
Rt_i	the data rate of flow i
Pl_i	packet length of flow i
Pv_i	packet interval of flow i
L_k^j	link connecting switches s_j and s_k.
Sc_k^j	scheduling cycle of egress port k of switch j
Ls_k^j	speed of link L_k^j
$Pt_k^j(f_i)$	time required to transmit a packet from f_i over link L_k^j
$lcm(i_1, ..., i_n)$	the least common multiplier of n integers from i_1 to i_n

Table 2.1: Notations

for each egress port, without coordination with the schedule of any other node (i.e., no global optimization). For simplicity, we implemented a local scheduling algorithm.

In our implementation, each egress port is assigned multiple queues, one per data flow (i.e., no queue sharing among flows). Each queue is processed according to a schedule. A schedule has a fixed cycle of constant duration time and repeats itself until a new schedule is calculated and activated. A schedule cycle is divided into a number of time slots. Each slot is at least equal to the packet transmission time of an outgoing link. Each time slot is either assigned to a queue or not used. Next we define the notations and equations used by the scheduling algorithm.

$$Pv_i = Pl_i * 8/Rt_i \tag{2.1}$$

We use UDP packets to simulate CPRI traffic of a certain data rate. We use Equation 2.1 to calculate the interval of UDP packets. Since the under-layers (e.g., IP and Ethernet layers) will add additional information to the packets (e.g., IP headers and Ethernet headers along with some padding bits), the actual traffic rate will be higher.

$$Sc_k^j = lcm(Pv_0, ..., Pv_n) \tag{2.2}$$

Given the packet intervals of all data flows, we use Equation 2.2 to compute the schedule cycle. This is to ensure that a schedule cycle is large enough to transmit packets from any data flow at least once within the cycle. This cycle repeats itself until simulation ends, thus all packets from all flows are accommodated, and no packet gets lost due to mis-scheduling.

$$Pt_k^j(f_i) = Pl_i * 8/Ls_k^j \tag{2.3}$$

A schedule cycle is divided into a number of time slots, each of which is allocated to a flow or not used. If a slot is allocated to a flow, the size of the slot must be large enough to allow the full transmission of a packet from that flow without causing any collision on the wire. Thus, the size of a time slot allocated for a given flow is a linear function of the transmission time of the packet from that flow, as given in Equation 2.4. The slot size has

the lower bound of packet transmission time, and the upper bound of the schedule cycle. Note $(a-1)/a$ percent of bandwidth is wasted if $a > 1$.

$$Ss_k^j(f_i) = a * Pt_k^j(f_i) \quad \text{where} \quad 1 \leq a \leq \frac{Sc_k^j(f_i)}{Pt_k^j(f_i)} \tag{2.4}$$

Algorithm 1 Computing a Schedule

```
for i = 1; i ≤ N; i++ do
    NumOfSlots[i] = Sc/Pv[i];
    for j = 1; j ≤ NumOfSlots[i]; j++ do
        Schedule[i][j] += Pv[i];
for i = 2; i ≤ N; i++ do
    for j = 1; j ≤ NumOfSlots[i]; j++ do
        Schedule[i][j] = ResolveCollision(Schedule[i][j], Schedule, i − 1);
```

Algorithm 2 Resolving Schedule Collision

```
procedure ResolveCollision(t, Schedule[][], M)
    for i = 1; i ≤ M; i++ do
        for j = 1; j ≤ NumOfSlots[i]; j++ do
            if |t − Schedule[i][j]| < Pt[i] then t = Schedule[i][j] + Pt[i];
    Return t
```

We use a simple network topology (Figure 2.1) to illustrate how to calculate a schedule for a given port. This network has three flows of $2.5Gbps$, $5Gbps$, and $10Gbps$ respectively. Assuming a packet length of 1000 bytes, these three flows have packet intervals of $3200ns$, $1600ns$, and $800ns$ respectively (Equation 2.1). Thus, the scheduling cycle is $3200ns$ (Equation 2.2). The transmission time of a packet of 1000 bytes over a $40Gbps$ link is $200ns$ (Equation 2.3). If we choose $a = 1.2$, the slot size would be $240ns$ (Equation 2.4).

The initial schedules for these three flows are $\{0\}, \{0, 1600\}$, and $\{0, 800, 1600, 2400\}$ respectively. After collisions are resolved, the schedules become $\{0\}, \{0, 240, 1600\}$, and $\{480, 800, 1840, 2400\}$. Figure 2.1 shows the combined schedule. Note this example assumes that the first packets from all flows arrive all at the same time 0 (i.e., the start time of the simulation).

3 Simulation Setup

In this section, we gives an overview of NS-3, describes our simulation implementation, and presents the network topologies used in our simulation.

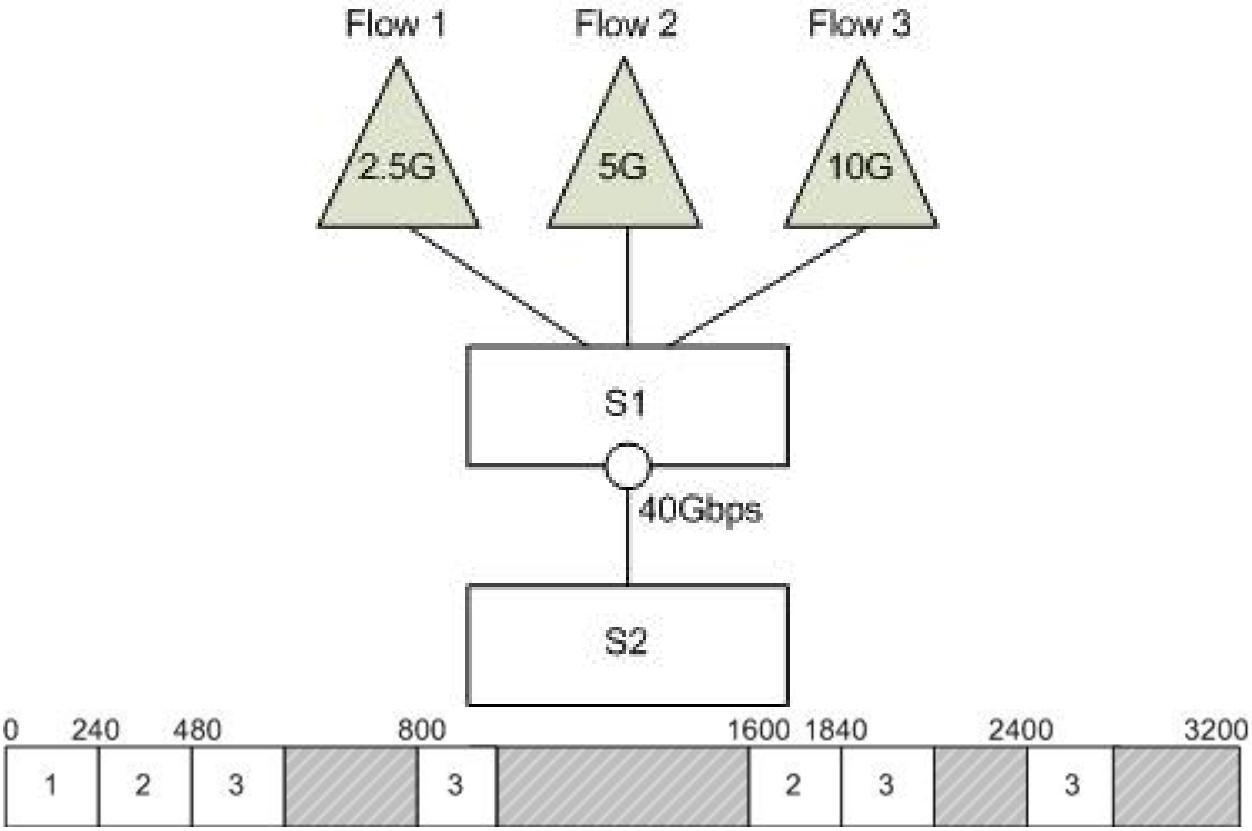


Figure 2.1: A Simple Network with 3 Flows and its Schedule

3.1 Network Simulator 3

NS-3 is a discrete-event network simulator, targeted primarily for research and educational use. It is open source software, licensed under the GNU GPLv2 license, and publicly available at http://www.nsnam.org/.

NS-3 is based on a clean slate design (not based on NS-2), aiming for easy use and easy extension. It is written entirely in c++, but Python wrappers exist for user codes. NS-3 includes many models. Those used in our simulation include, but not limited to: core, network, Internet, UDP Echo Application, and Flow Monitor.

3.2 Simulation Implementation

We implemented the schedule algorithm (see Section 2) in NS-3 by creating a class named **ScheduledQueue**. It inherits the abstract base class **Queue**. Beside implementing common queue operations such as *Enqueue* and *Dequeue*, we also implemented several operations specific to scheduling queue, such as **SetSchedule**, and **SetClassifier**. A schedule is computed prior to simulation, and **SetSchedule** is called to set the schedule for a network device port. **SetClassifer** sets a packet classifier for a network device, which is used to assign queues for packets based on their IP addresses or flow identifiers.

We also implemented a new network topology reader, based on Inet Topology Generator [7] from University of Michigan. Our topology reader allows to specify a traffic generator for a network node by providing traffic rate, packet size, packet priority, and packet destination. Based on those attributes, our simulation program automatically installs a UDP application on a network node, thus eliminating the need of code changes when testing with different traffic patterns.

To integrate the scheduled queue into NS-3 and to simplify simulation and measurement, we also made changes to several modules in NS-3, such as PointToPointNetDevice, FlowMonitor, UDP Echo Application, among others.

3.3 Network Topologies

The network topology (Figure 3.1) used in our simulation represents a Fronthaul network, serving three areas, each of which has 12 CPRI traffic flows. These three areas are connected via each other to a C-RAN. Traffic from all flows terminated at C-RAN (node 4). Since traffic aggregation occurs mostly from CPRI to C-RAN, we only study the behavior of this one-way traffic. The Ethernet port with a circle in Figure 3.1 is where traffic aggregation occurs, thus is configured with a scheduling algorithm when simulating Ethernet with scheduled traffic.

We also use a second network topology (Figure 3.2) to verify the results obtained from the first topology (Figure 3.1). This second topology represents an Ethernet network connecting two 3-floor buildings with a C-RAN. Note a triangle in Figure 3.2 represents a CPRI flow, and an oval represents a background traffic flow. Each of the 6 CPRI flows terminates at one of the C-RAN functions (e.g., B101), and a background traffic flow terminates at another node generating background traffic.

4 Simulation Results

Here we present the simulation results based on a network topology (Figure 3.1) representing a fronthaul mobile network.

Figure 4.1.(a) shows the jitter for standard Ethernet with and without background traffic. The former represents the jitter for Ethernet with preemption enhancement, and the latter represents the jitter for standard Ethernet (carrying only CPRI traffic). The results imply that standard Ethernet cannot meet the $65ns$ jitter requirements even when carrying purely CPRI traffic. Further, Ethernet with preemption could minimize the impact on jitter from background traffic, thus useful albeit insufficient.

Figure 4.1.(b) shows the jitters for Ethernet with scheduled traffic, where packets are of 1000 bytes and slot size is of 1.2 times of packet transmission time. When there is no background traffic, all jitter is completely removed. When there is background traffic, a small amount of jitter is incurred if preemption enhancement is also implemented. While this result might appear encouraging at first, further tests show scheduled traffic performs inconsistently when some of the parameters change. For example, when packet size is set to 1250 bytes, a few flows have a jitter of $1000ns$. Jitter also varies when slot size changes.

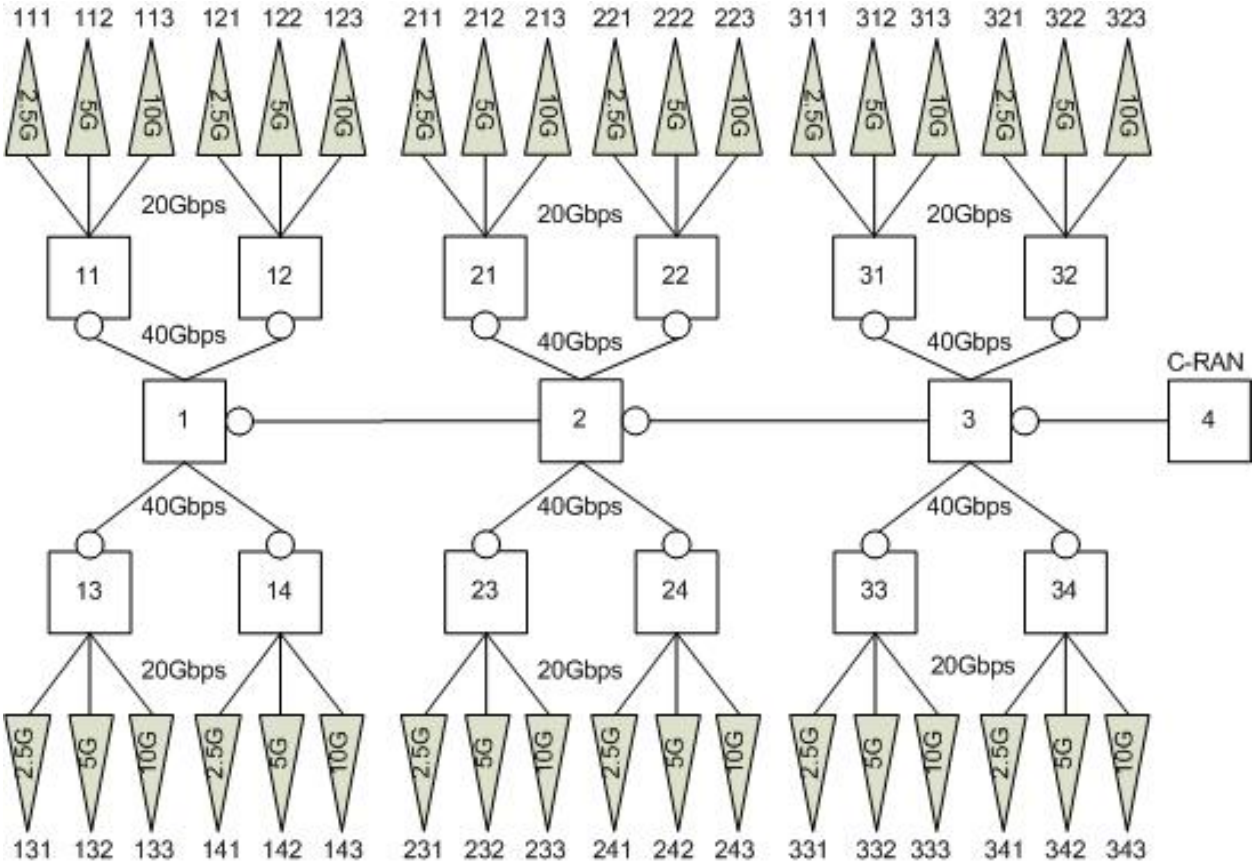


Figure 3.1: A Network Topology Representing a Front-Haul Mobile Network

Figure 4.3.(a) and Figure 4.3.(b) show the delays for Ethernet with preemption and Ethernet with scheduled traffic respectively. The worst case delays of preemption and scheduled traffic are $91us$ and $96us$ respectively, both of which are below the one-way delay requirement of $100us$ by CPRI.

We also use a different network topology (Figure 3.2), albeit smaller, to validate some of the results obtained from the larger topology. Particularly, we would like to verify if Ethernet with preemption performs consistently with different topologies and different background traffic patterns.

Figure 4.4.(a) shows the worst jitter is $168ns$ and Figure 4.4.(b) shows the worst delay is about $8us$. This demonstrates that Ethernet with preemption alone cannot meet the jitter requirement even in a small network with a few CPRI flows. Further, it confirms that Ethernet is capable of meeting delay requirements by CPRI.

Based on the above observations, we suggest that preemption enhancement to Ethernet is necessary but not sufficient in enabling CPRI over Ethernet. Other enhancements directly into Ethernet or outside of Ethernet are needed in order for Ethernet to meet the jitter requirement by CPRI.

To this end, we suggest an edge based buffering method to further reduce jitter for

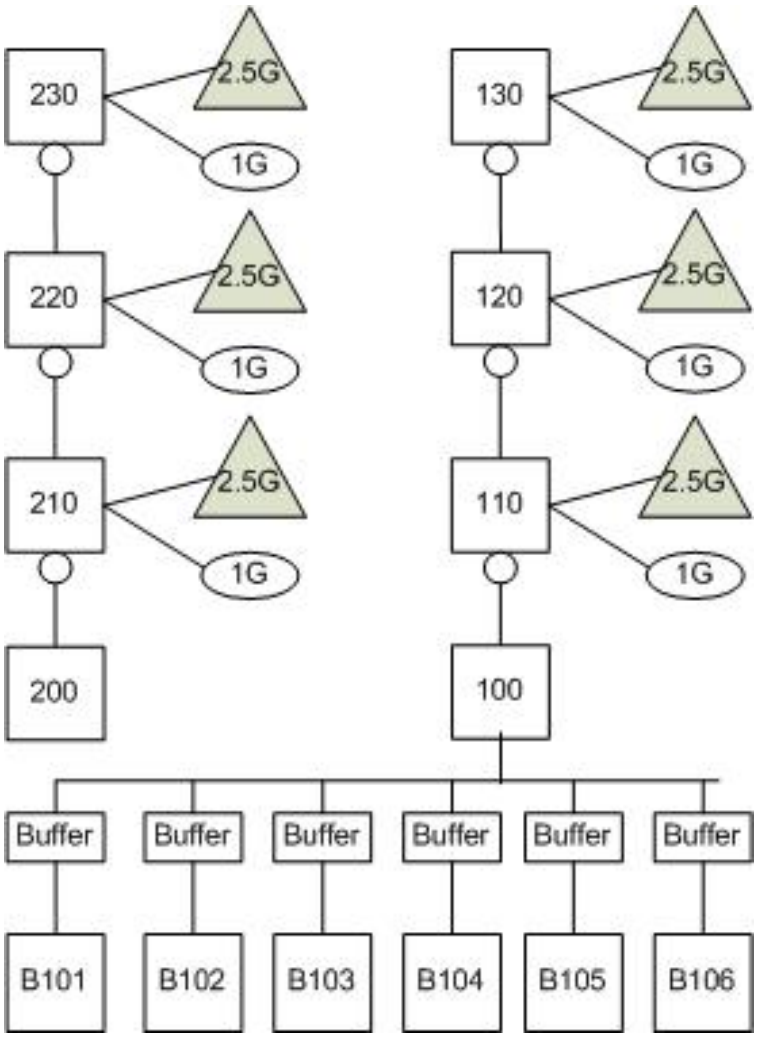
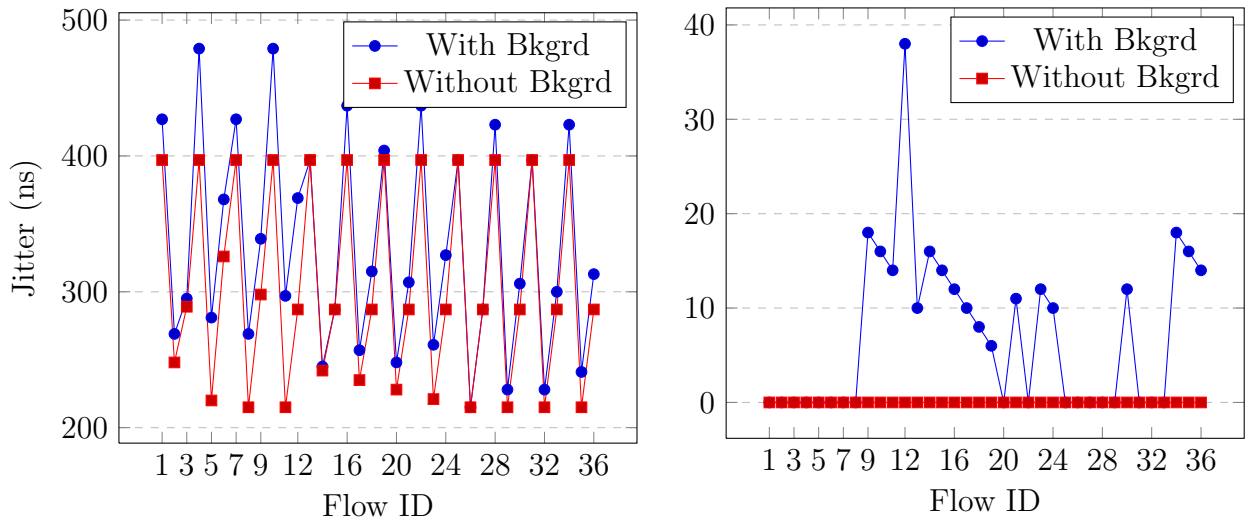


Figure 3.2: A Network Topology Representing a Fronthaul Mobile Network

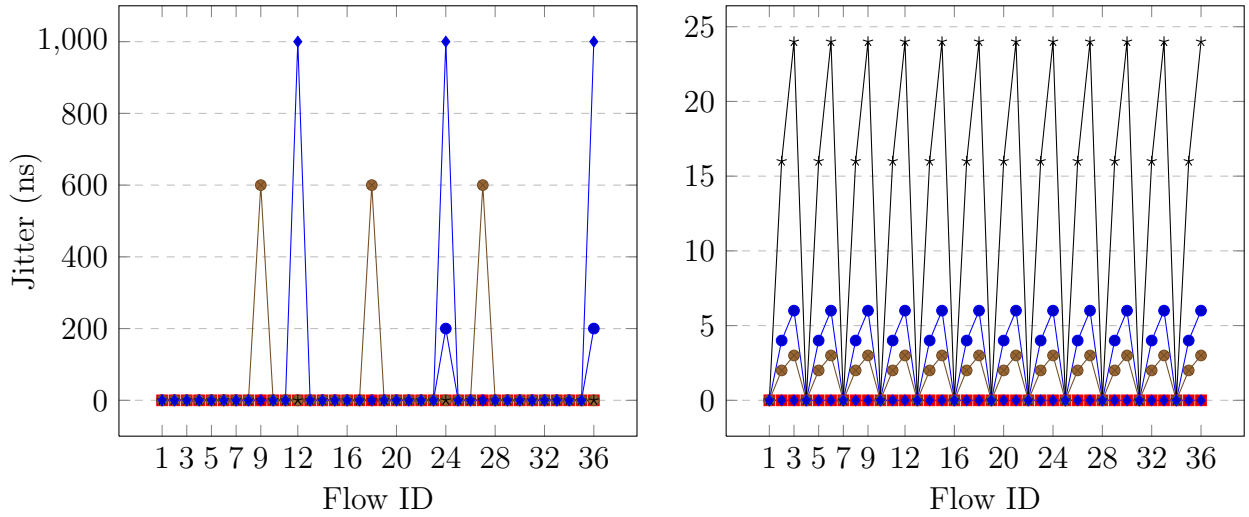
Ethernet. In this approach, a buffering function is inserted in front of each and every C-RAN function. Assuming that each C-RAN function serves a single CPRI flow, the corresponding buffering function could simply replay packets according to the frequencies of CPRI frames. In this way, jitter could be completely removed in theory, with a drawback of increased delay. Since delay is not a problem for Ethernet when carrying CPRI traffic, we believe this edge based buffering is worth of pursuing. Our simulation results also demonstrate that jitter could be all reduced to 0 while increased delay is still tolerable (Figure 4.4).



(a) Ethernet with Preemption

(b) Ethernet with Scheduled Traffic

Figure 4.1: Jitters for Ethernet with IEEE enhancements



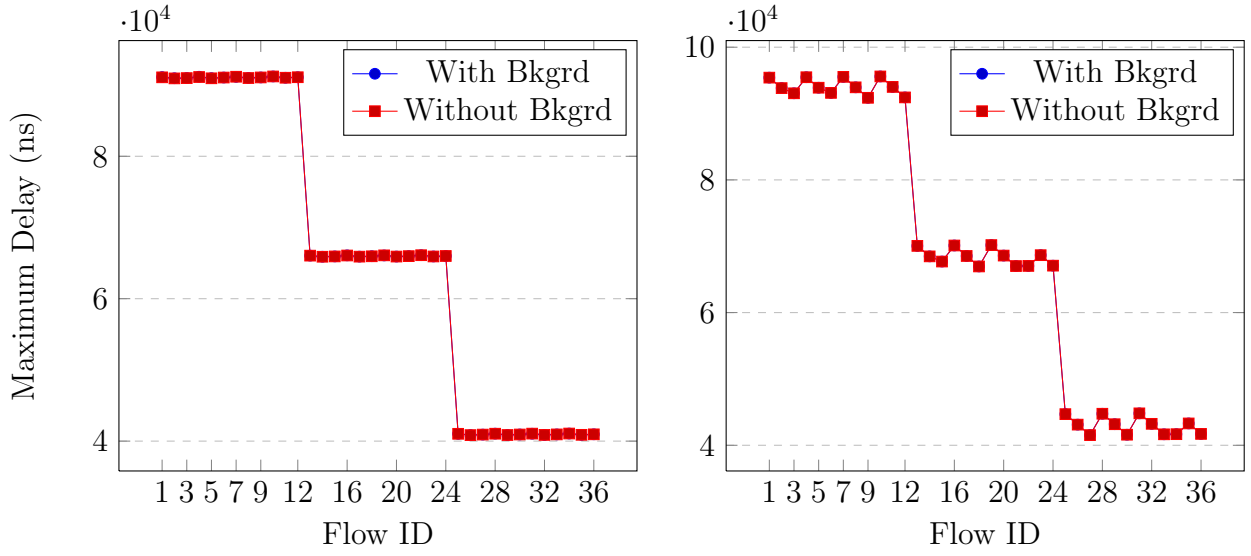
(a) Different Packet Sizes

(b) Different Slot Sizes

Figure 4.2: Jitters for Scheduled Traffic

5 Conclusions

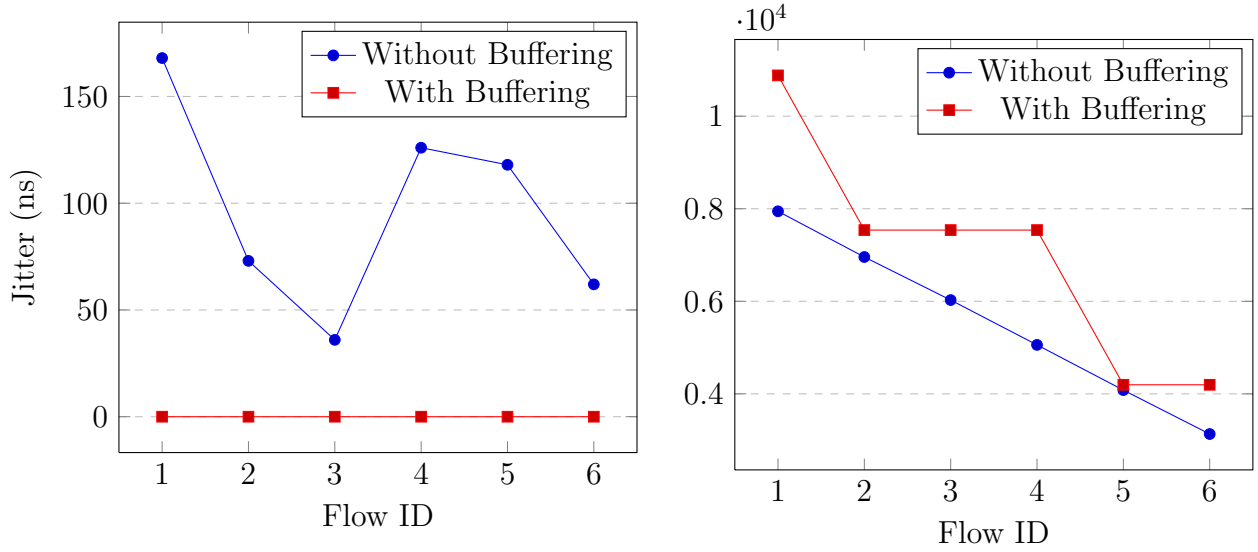
Ethernet is universally deployed and offers attractive cost benefits comparing with other types of networks, such as optical networks. However, the best-effort nature of Ethernet makes it unsuitable for carrying highly time-sensitive traffic, such as CPRI traffic. In this paper, we studied how CPRI over Ethernet behaves using simulations, and made several interesting observations.



(a) Ethernet with Preemption

(b) Ethernet with Scheduled Traffic

Figure 4.3: Maximum Delays for Ethernet with IEEE Enhancements



(a) Jitters with and without Buffering

(b) Delays with and without Buffering

Figure 4.4: Jitters and Delays for a Small Topology

First, standard Ethernet with a basic dropTail queue seems to incur only small amounts of jitter, and performs consistently with different packet sizes. With the enhancement of frame preemption to Ethernet, the amount of jitter incurred continues to be reasonable even in the presence of background traffic. *Second*, Scheduled traffic have the potential to completely eliminate jitter, but require more effort in developing advanced scheduling algorithms, such as SDN based global scheduling algorithms. Further, scheduled traffic seems intrusive and risk

incurring higher jitter if something unexpected happens, i.e., in the presence of packets not according to pre-computed and pre-configured schedules. *Third*, Ethernet with preemption, combined with other jitter reduction mechanisms, such as edge-based buffering, has the potential to meet the stringent jitter requirements of CPRI.

Acknowledgment

We thank Wen Tong for his guidance on this study, and Bill McCormick for his constructive comments on the design of the scheduling algorithm. Preliminary results of this study [6] were presented to IEEE 802.1 Time-Sensitive Networking (TSN) Task Group [3] in November 2014, and we thank the members of that group for their feedback, particularly on the amount of jitters that could incur from the preemption of low priority packets.

References

[1] Cpri specification v6.0. "http://www.cpri.info/downloads/CPRI_v_6_0_2013-08-30.pdf".

[2] Network simulator 3 (ns-3). "http://www.nsnam.org".

[3] Time-sensitive networking task group. "http://www.ieee802.org/1/pages/tsn.html".

[4] Ieee p802.1qbu - bridges and bridged networks - amendment: Enhancements for frame preemption. "http://www.ieee802.org/1/files/private/bu-drafts/d2/802-1Qbu-d2-0.pdf", 12 2014.

[5] Ieee p802.1qbv - bridges and bridged networks - amendment: Enhancements for scheduled traffic. "http://www.ieee802.org/1/files/private/bu-drafts/d2/802-1Qbu-d2-0.pdf", 10 2014.

[6] Peter Ashwood and Tao Wan. Cpri fronthaul reqirements - continuing discussion with tsn. "http://www.ieee802.org/1/files/public/docs2014/new-ashwood-tsn-cpri-fronthaul-1101-v01.pdf".

[7] Qian Chen Cheng Jin and Sugih Jamin. Inet: Internet topology generator. "http://topology.eecs.umich.edu/inet/".

[8] Fujitsu. The benefits of cloud-ran architecture in mobile network expansion. "http://www.fujitsu.com/downloads/TEL/fnc/whitepapers/CloudRANwp.pdf", 2014.

[9] China Mobile. C-ran: the road towards green ran. "http://labs.chinamobile.com/cran/wp-content/uploads/CRAN_white_paper_v2_5_EN.pdf", 10 2011.