Synchronization Challenges in Packet-based Cloud-RAN Fronthaul for Mobile Networks

Aleksandra Checko∗†, Anders Christian Juul†, Henrik L. Christiansen†, Michael S. Berger†

∗†MTI Radiocomp, Hillerød, Denmark
†DTU Fotonik, Department of Photonics Engineering, Technical University of Denmark, Kgs. Lyngby, Denmark

Email: aleksandra.checko@mtigroup.com, s104000@student.dtu.dk, hlch@fotonik.dtu.dk, msbe@fotonik.dtu.dk

I. INTRODUCTION

One option for mobile service providers to continue increasing capacity and coverage to their customers while remaining competitive, is to minimize costs of deploying new, expensive equipment in the mobile access network. Cloud Radio Access Network (C-RAN) offers such a solution. Cells share the computing power of each BBU, thereby increasing its utilization benefiting from multiplexing gain, lowering the overall amount of Baseband Unit (BBU) resources in the network. C-RAN is seen as one of the key technologies for 5G mobile networks [1].

The traditional view on the C-RAN architecture is to connect multiple Remote Radio Heads (RRHs) to the BBU Pool, each using a dedicated fiber connection to carry In-phase/Quadrature (IQ) data, as shown in Figure 1a. Synchronization and timely delivery of traffic are ensured by using a synchronous protocol - Common Public Radio Interface (CPRI), or less popular Open Base Station Architecture Initiative (OBSAI).

Following up on our earlier work [2], in order to decrease the cost of BBU pools, we propose packet-based fronthaul, where cells from residential or office areas can be dynamically associated to different pools to maximize multiplexing gain. The proposed architecture is shown in Figure 1b. In order to further optimize the cost, fronthaul network could utilize the existing Ethernet deployments so that IQ data shares resources with other types of traffic.

Today’s mobile technologies, especially LTE-Advanced (LTE-A), require high accuracy in terms of frequency and phase for proper transmission. By using C-RAN, these requirements are extended to the link connecting the RRH and the BBU Pool, and thereby to the Ethernet. It is not the nature of Ethernet to be synchronous, which conflicts with CPRI. The main challenge is then to find a method of providing synchronization across Ethernet in order to meet the demands of LTE-A, which this article focuses on.

Recently, several standardization activities have been started to redefine fronthaul network, showing a high interest in discussed architecture among industry. A proposal for a study item focusing on a variable rate multi-point to multi-point packet-based fronthaul interface supporting load balancing has been submitted to 3GPP. In October 2014 IEEE started 1904.3 Task Force [3] which focuses on encapsulation and mapping of IQ data over Ethernet which is applicable for fronthaul links. Synchronization aspect was considered, however, it is left for a new group to tackle, possibly within ITU-T. Moreover, IEEE 802.1 Time-Sensitive Networking Task Group is having discussions on CPRI [4] fronthaul ([5], [6]). Therefore, inputs on synchronization aspects are needed.

This paper is organized as follows. In Section II we present requirements of synchronization in mobile network. In Section III we discuss solutions for achieving phase and frequency synchronization in current mobile networks. In Section IV we analyze factors that contribute to inaccuracies in packet-based fronthaul and solutions to deliver synchronization. In Section V we present a proof of concept of a 1588 delivering synchronization information taking into account factors that introduce inaccuracies in 1588 operation. Finally, section VI concludes the paper.

II. LTE SYNCHRONIZATION REQUIREMENTS

Proper synchronization is essential for mobile network operation. In order for an RRH to send the data on a particular frequency it needs to know the precise definition of 1 s/1 Hz. It is important to keep the carrier frequency sharp in order for signal coming from base stations operating in different frequency bands not to overlap. For successful TDD network operation RRH needs to follow time frames precisely in order for DL and UL frames to not overlap. We can differentiate two types of synchronization: frequency and phase (time) synchronization. We say that clocks are synchronized in frequency if
the time between two rising edges of the clock match. For phase synchronization rising edges must happen in the same time.

In Table I we summarize the requirements that need to be observed for various network features, like Time-Division Duplex (TDD), Frequency-Division Duplex (FDD), Multiple Input Multiple Output (MIMO), enhanced Inter-cell Interference Coordination (eICIC) and Coordinated Multi-Point (CoMP). For latter three the requirement is expressed relatively to common reference between cells/streams involved, otherwise the maximum deviation from ideal source is listed.

Taking the example of ±50ppb frequency synchronization requirement it is worth noticing that only one-third of this budget, i.e. ±16ppb is available for frequency reference used for the radio frequency synthesis [7]. The rest of this budget is consumed within RRH.

III. SYNCHRONIZATION TODAY

In current networks typically BBU is equipped with a high precision clock having the time source in form of Global Positioning System (GPS), SyncE or IEEE 1588 [12], as shown in Figure 1c. GPS and IEEE 1588 can deliver both frequency and phase synchronization, while SyncE can only deliver frequency synchronization. Therefore SyncE is often used as a complementary solution to GPS or IEEE 1588 to enhance frequency accuracy. RRH gets precise time information via CPRI link, where timing information is included together with a data stream.

IV. SYNCHRONIZATION IN THE FUTURE

In the future a fronthaul link might be multipoint-to-multipoint with packet-based transport which will disturb synchronization. Comparison between synchronization solution used today and scenario for the future is illustrated in Figure 1. Especially challenging in Ethernet networks is the fact that packets will experience variable delays passing through the switches. Moreover, Ethernet switches themselves

<table>
<thead>
<tr>
<th>Feature</th>
<th>Frequency</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTE-A FDD</td>
<td>±50ppb (wide area BS)</td>
<td>±5 μs (cell with radius &gt; 3km)</td>
</tr>
<tr>
<td></td>
<td>±100ppb (local area BS)</td>
<td>±1.5 μs (cell with radius ≤ 3km)</td>
</tr>
<tr>
<td>LTE-A TDD</td>
<td>±250ppb (home BS)</td>
<td>[8], Clause 6.5.1.1</td>
</tr>
<tr>
<td>MIMO</td>
<td>≤ 65ns [8], Clause 6.5.3.1</td>
<td></td>
</tr>
<tr>
<td>eICIC</td>
<td>±1 μs [10]</td>
<td></td>
</tr>
<tr>
<td>CoMP</td>
<td>±1.5 μs [10]</td>
<td></td>
</tr>
</tbody>
</table>
Frequency error requirement: $\pm 16$ppb (Wide Area BS)
Cell phase synchronisation requirement: $\pm 5$μs between to overlapping cells

Performance affected by:
- Variable queuing delay
- Timestamping error: $+1$ns, $+4$ns

Ethernet clock accuracy defined as $\pm 100$ppm.
Realistic values are $\pm 1$ppm.

R-18: Maximum jitter contribution from CPRI link to RRH: $\pm 2$ppb.
R-20: Round trip absolute accuracy: $\pm 16.276$ns.

To use legacy RRH CPRI2Ethernet (CPRI2Eth) gateway is needed to bridge Ethernet and CPRI domains. For future RRH (RRH++) Ethernet link could terminate at RRH, omitting CPRI. A possible solution for delivering synchronization is to equip CPRI2Eth gateways or RRH++ with GPS. That solution assures both frequency and phase delivery. However, it increases the cost of solution (spend not only on GPS equipment but also on an oven-controlled oscillator). Moreover, coverage indoors and in metropolitan valleys (small cell on a lamp post in between high buildings) will be problematic. For some operators it is also important not to depend on a third-party solution for their network. Another solution could be to implement 1588 slave in CPRI2Eth gateways or RRH++. This solution assures lower equipment cost, however, it will be affected by variable network delay present in Ethernet networks. Ashwood [14] shows that such a jitter can be in order of μs per Ethernet switch.

We consider a system presented in Figure 2 with 1588 master present in BBU Pool and 1588 present in CPRI2Eth gateway. Figure 2 summarizes requirements on different layers: LTE, CPRI and Ethernet as well as factors introducing inaccuracies in 1588 and Ethernet layers. In section below we perform a feasibility study of using 1588 for timing delivery.

In modeling work we take into the account factors influencing 1588 performance mentioned in the Figure 2. However, we take into account a dedicated Ethernet network, leaving for future work the case of sharing Ethernet infrastructure with other types of traffic.

V. Feasibility Study of Using 1588 for Packet-Based Fronthaul

The 1588 standard defines a set of messages (Sync, DelayReq (delay request) and DelayResp (delay response)) for an end-to-end operation in order to exchange a set of timestamps between master and slave clocks as shown in Figure 3. Master clock needs a precise time definition e.g. from GPS and then propagates this timing information to slaves. Timestamp $t_1$ is inserted to Sync message when it leaves master node, $t_2$ is noted when message arrives to the slave. Each time Sync and DelayReq messages pass through an Ethernet switch (working as 1588 Transparent Clock (TC)) the value of CorrectionField ($CF_S$ and $CF_D$, respectively) is updated for residence time as presented in (1). It is very important as variable traffic in packet-based networks will fill-in the queues leading to variable network delay for 1588 packets. In response to Sync message slave sends DelayReq noting down when does the message leave the node - timestamp $t_3$. Master node notes the time the when it receives the message - $t_4$ and sends it to the slave via DelayResp message. Based on those timestamps, delay and offset between the clocks can be computed as shown in (2) and (3), respectively. More information of 1588 operation can
be found in [15].

\[ CF = EgressTimestamp - IngressTimestamp \] (1)

\[ Delay = \frac{(t_2 - t_1) + (t_4 - t_3) - CF_S - CF_D}{2} \] (2)

\[ Offset = \frac{(t_2 - t_1) - (t_4 - t_3) - CF_S + CF_D}{2} \] (3)

A protocol stack of the suggested solution is shown in Figure 4. Here the relevant protocols when CPRI traffic is transmitted over Ethernet using 1588 are shown.

The crucial aspect in implementing 1588 functionality is to execute timestamps generation as close as possible to the moment when each packet enters/leaves each node. It is important that the timestamp \( t_1 \) is taken exactly when Sync message leaves Master node, \( t_2 \) when Sync message enters Slave node etc. Otherwise, variable internal processing time of packets will affect the measurements. It is especially important in case of Ethernet switches, as variable residence time is expected depending on variable traffic queuing up in the switches. Therefore for the correction fields ingress and egress timestamps should be taken as soon as \( Sync \) or \( DelayReq \) packets enter and leave the node, respectively.

\[ DriftStd = \frac{t_2(N) - t_2(0)}{(t_1 + Delay + CF_S)N - (t_1 + Delay + CF_S)(0)} \] (4)

\[ DriftImplemented = \frac{t_2(N) - t_2(N-3)}{(t_1 + Delay + CF_S)N - (t_1 + Delay + CF_S)(N-3)} \] (5)

We have built a network model in OPNET modeler checking the performance of this algorithm. OPNET is a event-driven simulation software, where user can build his scenario from self-defined nodes and processes. We have built a network consisting of a 1588 Master, a 1588 Slave and variable number of Ethernet switches working as 1588 transparent clocks. \( Sync \) and \( DelayReq \) packet rate is 64 packets per second (pps). The novelty of our work is a network view where we test the protocol against various errors that can occur in the network. Slave node has an initial frequency drift of 1 ppm or 100 ppm (maximum that an Ethernet switch can have). Each of the Ethernet switches has a frequency error of 1 ppm or 100 ppm (maximum that an Ethernet switch can have). Each of the Ethernet switches has a frequency error of 1 ppm or 100 ppm (maximum that an Ethernet switch can have). Each of the Ethernet switches has a frequency error of 1 ppm or 100 ppm (maximum that an Ethernet switch can have). Each of the Ethernet switches has a frequency error of 1 ppm or 100 ppm (maximum that an Ethernet switch can have).
For each exchange of 1588 messages, after all timestamps are gathered, delay, drift and offset are computed. Drift correction is applied to slave clock frequency $f_S$ every third exchange of timestamps. That affects both synchronous and syntonous time scales at slave having impact on $t_{2(Syntonous)}$ and $t_{3(Syntonous)}$. Synchronous time scale of slave is updated for offset after each exchange of timestamps. That affects local time at slave $t_S$. We have measured a relative frequency error between master clock frequency $f_M$ and slave clock frequency $f_S$ as presented in (6) and an absolute phase error between time in master $t_M$ and time at slave $t_S$ as presented in (7).

$$\text{Frequency Error} = \frac{f_M - f_S}{f_M}$$

(6)

$$\text{Phase Error} = t_M - t_S$$

(7)

Figure 6 presents maximum observed phase (time) error during stable operation (after initial time discovery) for different number of Ethernet switches present in the network and two different timestamping error values. Phase error stays in the order of nanoseconds and is highly dependent on timestamping errors. The dependency is close to linear. It is not perfectly linear as model was run only for one seed value. The results are shown for the worst-case scenario of 100 ppm drift for both Ethernet switches and slave, as drift value had marginal effect whether it was 1 ppm or 100 ppm in both cases.

Figure 7 shows the frequency error for the above mentioned scenarios. Frequency error falls way above required values (16 ppm or below, depending on implementation). It is also highly dependent on timestamping errors. That is the reason why we apply improvements to this method.

For improvement, we apply averaging to both offset and drift correction. The method is presented in (8) for drift and applies also for offset. Frequency is adjusted by averaged drift computed taking only a fractional value of currently computed drift - drift - and previously computed drift - driftPrev. We checked the performance of the system for different values of alpha, and we concluded that the higher it gets, the lower frequency error is observed. However, for higher values of alpha it takes more time for the system to converge to stable operation. For simulation we used alpha = 0.99. After 180 s system reached stable operation. Frequency error got significantly smaller for all the cases (20 times smaller), while phase error got slightly smaller (2 times) as presented in Figures 8 and 9.

$$\text{driftAvg} = \alpha \cdot \text{driftPrev} + (1 - \alpha) \cdot \text{drift}$$

(8)

$$\text{freq} = \text{freq} - \text{driftAvg}$$

(9)

### B. Presented results vs mobile network requirements

It should be noted what is the dependency between synchronization inaccuracies of a local oscillator and inaccuracies of an RF signal. As presented in Figure 10, Ethernet signal entering an RRH carries both the IQ baseband signal and 1588 packets carrying timestamps. The 1588 module processes
We presented a feasibility study showing the performance for frequency and phase synchronization using 1588 in Ethernet networks under various inaccuracies that can be present in the network. The one that has highest impact is the timestamping error associated with the way timestamps are generated in Ethernet switches. Whether this performance will meet the requirements of future mobile network depends on PLL and local oscillator implementation, based on 1588 feedback to clock offset and drift.

The most challenging aspect of packet-based networks that will influence 1588 performance is background traffic. Our future work will cover testing the solution against networks with variable traffic. Moreover, in our future work we will study the delay that a shared Ethernet fronthaul introduces.

**VI. CONCLUSIONS AND FUTURE WORK**

C-RAN is seen as a key technology for 5G networks. A deal breaker for an operator whether to employ it is the fiber availability. Moreover, they want to optimize the overall cost of the solution. Therefore reusing existing Ethernet networks for C-RAN fronthaul is a promising solution and an important matter to study. However, synchronization is the main challenge.

In this work we analyzed factors that are challenging for achieving synchronization in packet-based C-RAN fronthaul.

---

**REFERENCES**


[14] P. A. Smith. CPRI FrontHall requirements discussion with TSN.
