IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) | 978-1-6654-9427-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/INFOCOMWKSHPS57453.2023.10226161

Towards a Network Aware Model of the Time Uncertainty Bound in Precision Time Protocol

Yash Deshpande, Philip Diederich, Wolfgang Kellerer

Chair of Communication Networks, Technical University of Munich, Germany
Email: {yash.deshpande, philip.diederich, wolfgang.kellerer}@tum.de

***Poster Abstract*—Synchronizing the system time between devices by exchanging timestamped messages over the network is a popular method to achieve time-consistency in distributed applications. Accurate time synchronization is essential in applications such as cellular communication, industrial control, and transactional databases. These applications consider the maximum possible time offset or the Time Uncertainty Bound (TUB) in the network while configuring their guard bands and waiting times. Choosing the right value for the TUB poses a fundamental challenge to the system designer - a conservatively high value of the TUB decreases the chances of time-based byzantine faults but increases latency due to larger guard bands and waiting times. The TUB is affected by packet delay variation (PDV) of time synchronization messages due to congestion from background network traffic. In this work, we use Network Calculus (NC) to derive the relation between network traffic and the TUB for a network built with commercial off-the-shelf (COTS) hardware. For centrally deployed and monitored local area networks (LAN)s such as in cellular networks and datacenters this relation could be useful for system designers to plug a better-informed value of the TUB.**

I. Introduction

A consistent understanding of time is required between all devices in a network to realise applications such as network telemetry, centralized log collection, transactional databases, cellular communication and real-time communication using Time Sensitive Networking (TSN) methods etc.

The designer of the applications needs to consider the maximum possible time error in the network to specify the accuracy of delay measurements or set the read-write waiting time in databases. Thus, the Time Uncertainty Bound (TUB) in a network ϵ was introduced in Spanner [1], which is defined as the maximum clock error between any two distributed devices in a transactional database. Furthermore, Farsite [2], a distributed file server, uses a similar bound to specify a guard time on top of its content lease time. Moreover, this uncertainty bound plays an important role in deciding time based guard bands in cellular communication using Time Division Duplex (TDD) mode [3].

The Precision Time Protocol (PTP) is one of the most popular methods of time synchronization over the network. It is designed for local and more centrally planned networks such as telecom fronthaul, datacenter, and industrial networks. The packet delay variation (PDV) of PTP messages due to congestion from other traffic in the network is one of the main reasons that degrades the synchronization accuracy of PTP.

In this work, we dive deeper into characterizing the relationship between the TUB and network conditions. Here, we consider networks that contain commercial off-the-shelf (COTS) hardware and common PTP configurations. To the best of our knowledge, this is the first such attempt to model the TUB.

II. Modelling and Evaluation

A. Modelling

A typical PTP message exchange starts with a *Sync* message sent out periodically from the clock source. This message contains the timestamp T_1 that describes the point in time when the message is sent on the network. Once the sink receives this message, it records the timestamp of arrival T_2 and then sends a *Delay Request* message back to the source. Then, the source immediately replies to the sink with a *Delay Response* message. The sink records the arrival time T_3 of the *Delay Response* message. With the three timestamps, the sink calculates its offset to the source clock as follows:

$$\textit{Offset} = T_2 - T_1 + D_{sync} = T_2 - T_1 + (T_3 - T_2)/2. \quad (1)$$

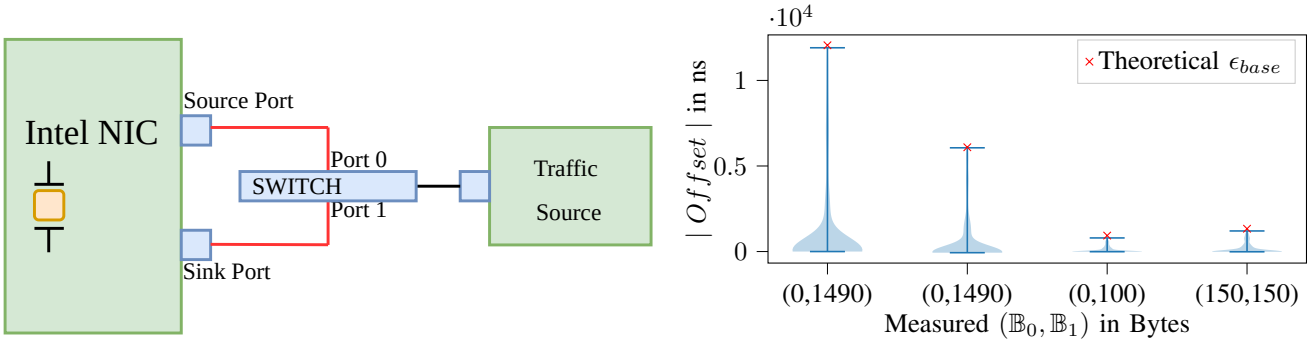
$(T_3 - T_2)/2$ is essentially half of the Round trip time (RTT) between source and sink which is assumed to be equal D_{sync}, the propagation time of the *Sync* message. The sink subtracts this value of estimated offset to its system time to synchronize with the source. Note that the offset value can be positive or negative depending on whether the sink is ahead or behind the source clock.

The TUB between the sink and source depends on the maximum offset estimation error ϵ_{base}, synchronization rate, and the maximum clock drift rate between the source and sink [4]. If there are no packet drops in the network, the update rate and maximum clock drift rate can be considered as constants, therefore the only variable in TUB is ϵ_{base} which contains two components: the maximum error in the estimation of D_{sync} due to PDV and the maximum timestamping error when the hardware timestamps T_1, T_2 and T_3 the arrival and sending of messages. Assuming that the source and sink have the same hardware, the maximum timestamping error at the source and sink will also be the same. Hence, we get ϵ_{base} from equation 1,

$$\epsilon_{base} = max(|\ \ D_{sync} - (D_{req} + D_{resp})/2\ |) + 3T_E, \quad (2)$$

where the first term represents the maximum error due to PDV, and T_E is the maximum timestamping error. The equation

978-1-6654-9427-4/23/$31.00 ©2023 IEEE



(a) (b)

Fig. 1: (a) Measurement setup for the evaluation of our model. Two ports of a generic Intel NIC driven by the same crystal oscillator are used. The true offset between the source and sink is zero. Hence, any clock offset reported the offset estimation errror. We use the standard *linuxptp* program for our evaluation. Another traffic source congests the PTP messages. (b) The results of the measured offset estimation error over one day (∼80,000 measurements) for background traffic with different packet sizes. The maximum offset estimation error closely agrees with the theoretical ϵ_{base} from our model.

shows the existence of two extreme cases. The two cases occur when PDV maximizes the first term in equation 2. The first scenario where this happens is when *Sync* experiences maximum delay due to congestion and *Delay request* and *Delay response* experience no congestion. In the second extreme scenario, *Sync* does not experience any congestion delay, but *Delay request* and *Delay response* experience maximum delay due to congestion. We represent the end-to-end delay of message m as a sum of three delay components: time spent on wire, processing time in switches, and queueing delays in switches, $D_m = D_{wire} + D_{proc} + D_{queue}$. D_{proc} is the sum of all the processing times of the message m over every switch that the message m traverses. Processing time of a message m usually depends on the packet size and can be found experimentally [5] or is sometimes provided by the manufacturer of the switch. Furthermore, one needs to only find the maximum and minimum processing time per switch for the packet size of m without any other traffic on the switch. D_{wire} faces very little jitter and is also provided by cable manufacturers, e.g., 5ns/m for CAT5 copper cables. Hence, D_{wire} only requires the knowledge the total cable length that m has to traverse. Maximum queuing delay $max(D_{queue})$ can be found out using Network Calculus (NC) if the network traffic conditions are known. For our model, we consider a 802.1p (priority scheduling at the end-hosts and switches) enabled network. In this case, all PTP profiles recommend that synchronization messages are sent with the second-highest priority. We assume all other background traffic to have lower priority than PTP traffic. Using NC rules, we obtain $max(D_{queue}) = \sum_{p \in P_m} \mathbb{B}_p/R$, where P_m is the set of all switch egress ports that m traverses and $\mathbb{B}_p$ is the largest packet size in bits out of all flows that exist on port p. We also assume a link capacity R of 1Gbit/s on all links and that this is never exceeded by network traffic.

B. Evaluation

The measurement setup for the evaluation of our model is shown in figure 1a. We consider a network with one *store-and-forward* switch between the source and sink. Minimum and maximum of D_{proc} were determined using the method in [5] to be 2255 ns and 2365 ns respectively. All three PTP messages have the same packet size and hence have the same processing time distribution at the switch. Figure 1a shows the cable that PTP messages traverse in red. The total length of both sections sums up to 4m resulting in D_{wire} of $20ns$.

T_E was found to be $8ns$ independent of the traffic load, which is common for COTS Network Interface Card (NIC)s with 125MHz clock sources [6]. Thus, we obtain $max(D_{sync}) = max(D_{resp}) = 2385ns + \mathbb{B}_1/Rns$ and $\max(D_{req}) = 2385ns + \mathbb{B}_0/Rns$ and $min(D_{sync}) = min(D_{resp}) = min(D_{req}) = 2275ns$. Putting these values in equation 2 we get ϵ_{base} of either $(134 + \mathbb{B}_1/R)ns$ if $\mathbb{B}_1 \geq \mathbb{B}_0$ or $(134 + \frac{(\mathbb{B}_0+\mathbb{B}_1)}{2R})ns$ otherwise. Figure 1b shows that the measurement results agree closely with our theoretical derivation.

References

[1] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford, "Spanner: Google's globally distributed database," *ACM Trans. Comput. Syst.*, 2013.

[2] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer, "FARSITE: Federated, available, and reliable storage for an incompletely trusted environment," in *5th Symposium on Operating Systems Design and Implementation (OSDI 02)*, 2002.

[3] "Packet over Transport aspects – Synchronization, quality and availability targets," *ITU-T G.8271, International Telecommunications Union*, 2020.

[4] Y. Li, G. Kumar, H. Hariharan, H. Wassel, P. Hochschild, D. Platt, S. Sabato, M. Yu, N. Dukkipati, P. Chandra, and A. Vahdat, "Sundial: Fault-tolerant clock synchronization for datacenters," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020.

[5] A. van Bemten, N. Deric, A. Varasteh, A. Blenk, S. Schmid, and W. Kellerer, "Empirical predictability study of sdn switches," in *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2019.

[6] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "Moongen: A scriptable high-speed packet generator," in *Proceedings of the 2015 Internet Measurement Conference*, 2015.