Time-Sensitive Networking (TSN): An Experimental Setup

Morteza Hashemi Farzaneh and Alois Knoll
Department of Robotics and Embedded Systems
Technical University of Munich
Boltzmannstr. 3, 85748 Garching bei München
Email:{hashemif,knoll}@in.tum.de

***Abstract*—Time-Sensitive Networking (TSN) is a set of upcoming standards supporting highly deterministic communication based on the Ethernet. As a candidate for in-vehicle communication infrastructure, it has recently raised significant attention of the automotive domain. A prototypical experimental setup is designed and developed for the purpose of benchmarking with focus on latency and jitter of time-triggered periodic frames described in IEEE 802.1Qbv.**

***Index Terms*—Time-Sensitive Networking, TSN, Real-Time, Ethernet, Time-triggered**

I. Introduction

Time-Sensitive Networking (TSN) [1] is a task group of the Institute of Electrical and Electronics Engineers (IEEE) targeting the development of standards to support hard real-time communication based on the Ethernet technology. These standards raised attention of automation and automotive. To guarantee highly synchronized communication with very low latency and jitter values, *IEEE 802.1Qbv* is proposed based on the time-triggered communication paradigm. Combining this with the priority classes defined in IEEE 802.1Q, frame queuing delays in the egress ports are minimized.

Considering the simulation-based approaches, performance of Audio Video Bridging (AVB) and 802.1Q are compared in [2] for an in-vehicle scenario using discrete event simulation tool OMNET++. The results show that AVB improves the real-time capabilities of the Ethernet but it still requires more advanced features to fulfill the tight timing requirements of e. g. critical motion control applications. The time-triggered Ethernet and AVB are analyzed and compared regarding their performance in [3], with focus on driver assistance systems. Extending AVB with time-triggered feature is simulated in [4] and [5]. The results show that event-based traffic can increase the end-to-end latency and jitter in the network. The worst-case analysis of TSN traffic shapers (time-aware, persitaltic and burst-limiting) are recently investigated in [6]. Based on the results, only the time-aware shaper 802.1Qbv can support very low latency and jitter guarantees. The peristaltic and burst-limiting shapers can be enhanced using frame preemption mechanisms. This approach has been improved in [7] using a compositional performance analysis [8] considering interference by same-priority frames. The results are compatible with [6] and the time-aware shaper achieves the lowest latency if nodes are synchronized. Formal methods may come to very pessimistic estimations regarding timing behavior.

However, simulation-based and formal methods do not deal with configuration challenges of TSN such as schedule synthesis. The schedule synthesis of time-triggered Ethernet is discussed in [9]–[12]. Based on a logic-based modeling approach proposed in our previous work [13], we developed a graphical network modeling tool [14] to automate the schedule synthesis of TSN and to support an iterative model correction by interpretation of logic-based unsatisfiable cores. This tool is used to distribute the synthesized time-triggered schedule among the network switches. Performance evaluation of the developed tool on real hardware requires to ensure that the basic underlaying time-triggered functionalities, are working according to the specification.

In this paper, we present a new experimental setup and investigate the timing behavior of the prototypical 802.1Qbv switches by generating competing unscheduled traffic with variations in load and priority.

The remainder of this paper is structured as follows. First, we describe the experiments' objectives and expectations in Section II, then we introduce our demonstrator and experiments in Section III. The experiment results are discussed in Section IV and finally we conclude in Section V.

II. Objectives

The subject of the investigations are *periodic real-time frames* in the network which will compete with unscheduled traffic. Because there is no time-triggered scheduling mechanism in 802.1Q, the expectation is that the frames with highest priority are transmitted faster in average which reduces their experienced average latency and jitter.

Hypothesis 1: Maximum and average latency of a real-time periodic frame when applying 802.1Q, depends on the priority and network load generated by other competing frames.
If there is no synchronization between the end-stations, the sender of a periodic frame will miss its slot with high probability and has to wait for its next slot. This phenomenon may increase the average latency and jitter compared to the situation when all nodes are synchronized but guaranties the next slot which is an upper bound for the worst-case latency.

Hypothesis 2: By applying 802.1Qbv without synchronization of the end-stations, a deterministic latency upper bound

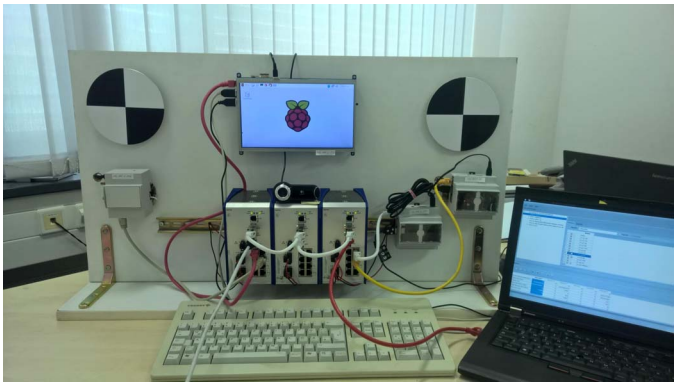

Fig. 1. Three industrial RSP35 switches with each two prototypical FPGA-based 802.1Qbv ports. Automotive exemplary applications (drowsiness detection, step motor control and infotainment video streams) are implemented on end-stations based on RaspberryPi 3 and Odroid C1 embedded hardware. The notebook generates the desired unscheduled network traffic required for the experiments.

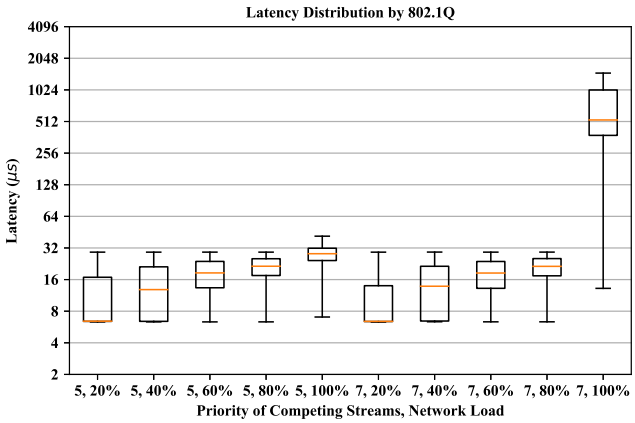

Fig. 2. Higher traffic load and priority of the unscheduled traffic leads to higher latency and jitter for the time-triggered frame.

for a periodic time-triggered frame is guaranteed if and only if there are no other *unscheduled* frames with the same priority (higher or lower priorities are allowed).

Synchronization in TSN is described in IEEE 802.1AS-Rev which exploits Precision Time Protocol (PTP) described in IEEE 1588-2008. The PTP functionality is implemented either software-based on the top of an operating system or hardware-based. A hardware-based time stamping leads to synchronization with higher precision. In spite of this advantage, it narrows down the choice of network designers regarding the embedded hardware and may cause additional hardware costs. This costs may be significant when network components are used in serial productions such as in the automotive domain. We intend to investigate the capabilities of the cheaper software-based PTP implementation.

Hypothesis 3: 802.1Qbv with software-based PTP synchronization achieves the lowest latency and jitter values for a periodic frame compared to Hypothesis 1 and Hypothesis 2 if and only if all other competing frames with the same priority are scheduled and synchronized.

III. Experiments

The setup is motivated by typical mixed-critical automotive applications. Two synchronized step motors communicating through the network are representative for hard real-time applications with periodic behavior. It requires guaranteed lowest latency and jitter values. A drowsiness detection application uses video stream of a camera to analyze the drivers' eyes and triggers alarm if it estimates that the driver has fallen asleep. In contrast to the step motors, this application requires maximum latency guarantees and jitter is less important. An infotainment video streaming application represents those applications which do not have tight timing requirements but requires high communication bandwidth.

The subject of investigation is the latency and jitter behavior of the periodic frames generated by the step motor application. The frame has a period of $500\mu s$, a size of 200 Bytes, and an exclusive QoS priority 6.

We use the two other applications as unscheduled competitors generating ~ 80 Mbit/s traffic load. To generate arbitrary higher traffic loads, the traffic generator software Ostinato [15] is applied. The critical frame is sent out from the most right end-station to the most left one. Two Odroid-C1 (CPU: QuadCore Amlogic S805 Cortex A5 1.5 GHz, 1GB RAM) embedded computers (with Ubuntu 14.04) are used as end-stations executing the step motor application. Between these end-stations, there are three RSP35 industrial switches with prototypical FPGA-based Ethernet ports (1Gbit/s) supporting the basic functions of 802.1Qbv.

Important note: Using hardware-based time stamping features of the switches (in range of few nanoseconds), the latency and jitter of a periodic frame is measured with a high precision. Because this feature is only available in the switches and not in the end-stations, we measure the latency of a frame as $t_{latency} = t_{out} - t_{in}$ where t_{in} is the arriving time of the frame in the first ingress switch port and t_{out} is the time when the frame leaves the last egress switch port towards the receiver end-station. This approach is used in all experiments to avoid measuring disturbance caused by the end-stations. The components of the setup are described in Figure 1.

A. Experiment 1: 802.1Q

In this experiment, we measure the timing impact of the priority and load of the unscheduled traffic on the critical frame to validate Hypothesis 1. The latency is measured by increasing the unscheduled network load with data rates of $200, 400, 600, 800$ and 1080 Mbit/s (to create network congestion) once with lower priority 5 and once with higher priority 7.

The results as depicted in Figure 2 show, that increasing the network load leads to higher maximum and average latency values. The impact is maximum if the unscheduled traffic has a higher priority than the critical frame and the network is congested (cf. $7, 100\%$). If all unscheduled frames have

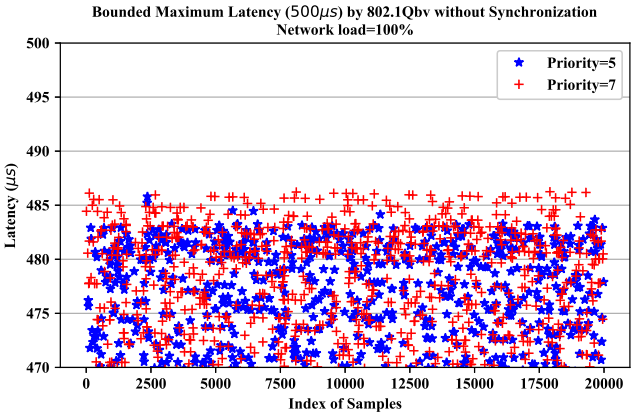

Fig. 3. Independent from the priority of unscheduled traffic, the maximum latency of the time-triggered frame is guaranteed even when the network is congested.

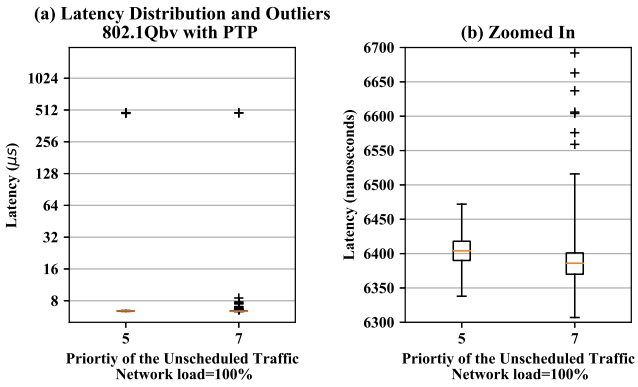

Fig. 4. PTP-synchronized 802.1Qbv minimizes the latency and jitter values for the time-triggered frame. There are two outliers from 20000 samples caused by not deterministic network stack of the sending end-station.

a priority less than the priority of the critical frame, lower average latencies can be achieved even if the network is congested (cf. $5, 100\%$). As expected, it can be observed that the achieved worst-case and average latency of periodic frames when using 802.1Q, depends on the shape of unscheduled network traffic.

B. Experiment 2: 802.1Qbv without synchronization

In this experiment, the gate controlling feature of 802.1Qbv is examined. Gate Control List (GCL) is used to configure time slots defining when and how long the gates of an egress port are open for a given QoS priority class. We use priority 5 and 7 and assign them to the unscheduled traffic and measure the worst-case latency.

The cycle time of the GCL is set to $500\mu s$ equal to the period of the critical frame. We reserve the first $25\mu s$ of the cycle for the transmission of the critical frame exclusively and use the remaining slot for the unscheduled traffic. In this experiment, the end-stations are not synchronized. Thus, they measure the current time based on their own CPU clock. In this experiment, 20000 samples of the critical frame are fired once competing with priority 5 and once competing with priortiy 7 traffic. The network is congested and there is a random waiting time between two consequent samples. The waiting time is required to assure varying sending times for the samples causing different latencies. Independent from the random sending time, the maximum measured latency has to be less than $500\mu s$ according to the specification.

The results are presented in Figure 3 and show that as expected in Hypothesis 2, the maximum latency is bounded by $500\mu s$. The latencies in this experiment are in average higher than the latencies from last experiment when only 802.1Q is applied. This is because of the missing synchronization in the network. The sender does not have the accurate time to hit its reserved slot and will miss it and has to wait for maximum one GCL cycle. However, in contrast to 802.1Q, the latency is bounded independent from the computational capacities of the end-station, load and priority of the unscheduled traffic. Additional to the guaranteed worst-case latency, very low and deterministic jitter can be achieved which only depends on the accuracy of the end-station's CPU clock accuracy.

Such a guarantee can be given if and only if there are no unscheduled frames with the priority 6 (a priority equal to the priority of the critical frame) in the network. Unscheduled frames with the same priority will occupy the reserved time slot and increase the latency. Such frames are allowed in the network only if they have their own guaranteed time slots to avoid disturbing the critical frames.

Considering the measurements, 802.1Qbv is able to fulfill the requirements of latency-sensitive applications e. g. the before-mentioned drowsiness detection application.

C. Experiment 3: 802.1Qbv with with PTP

The only difference between this experiment and experiment 2 is that the end-stations are synchronized using a software-based implementation of PTP. Similar to experiment 2, the assumption is that unscheduled traffic does not have the same priority as the critical frames. Based on the 802.1Qbv specification, an end-station has to calculate its periodic sending times t_{send} to transmit frame instances correctly. It requires information about the *OperBaseTime* (t_{base}) which is the same value for all network nodes to calculate the starting time of the next GCL cycle, *OperCycleTime* (t_{cycle}) which is the hyper period of all periodic frames traveling through an egress port, and its own *Offset* (t_{offset}), which describes the exact position within a cycle reserved for the frame.

Additionally, an end-station has to know its one-way latency Δ to the first switch port. Based on Δ value, the steam is sent out earlier from the end-station such that it arrives on time. One way to find the correct value is to observe the PTP output which exactly delivers the desired average one-way latency to the master clock (which is the first switch connected to the end-station). In this experiment we measured $\Delta = 27\mu s$. To calculate t_{send}, an end-station reads its own synchronized

clock time $t_{current}$. Using the current time, it calculates the next sending time t_{send}:

$$\alpha = \lceil \frac{t_{current} - t_{base}}{t_{cycle}} \rceil$$
$$t_{send} = (\alpha * t_{cycle} + t_{offset}) - \Delta$$

The results as depicted in Figure 4 show that the average latency and jitter is very low even if the network is overloaded and unscheduled frames have higher priority. One of the interesting findings is that even though the end-stations are synchronized, in very rare cases (two *outliers* in 20000 cycles), the sender misses one GCL cycle. This lead to a worst-case latency equal to the cycle length ($500\mu s$). The latency outliers are caused by the inaccuracy of the network software stack of the end-station. We observed the Δ continuously and found out that its value is increased to $44\mu s$ for a moment and is returned instantly back to $27\mu s$. This anomaly causes the outliers.

To avoid this problem, we propose two solutions. A first solution is to assign a higher value to Δ ($\geq 44\mu s$) in the calculations and guarantee that the frame will be sent out early enough to hit its slot. This approach has the disadvantage that the frame experiences in average $44\mu s - 27\mu s = 17\mu s$ additional latency because it arrives too early. The second solution is to arm the end-station with a real-time operating system to have full control the network stack and the hardware interrupts. Both solutions definitely remove the outliers such that very low latency and jitter values are guaranteed.

IV. Discussion

Considering the results of the experiments, the TSN switches conform to the 802.1Qbv specification and can be used for further advanced network benchmarking scenarios. Moreover, the following network design recommendations are summarized.

The standard 802.1Q can be used to support applications with soft timing requirements such as infotainment and video streaming. It may be used for more critical time-sensitive applications such as camera sensors involved in autonomous driving scenarios, if the whole traffic shape is analyzed at design time regarding the traffic load and priority fo the frames. However, it does not support any guarantees regarding the worst-case latency. For applications with hard bounded maximum latency requirements, the credit-based shaper of AVB or the 802.1Qbv without synchronization can be considered. The advantage of the latter option is that the end-stations do not require to support any special features. For applications which require guarantees for very low latency and jitter values, 802.1Qbv with software-based or hardware-based PTP synchronization has to be applied. The low-cost software-based approach can cover a wide range of real-time applications with synchronization requirements.

Integration of TSN standards in the future automotive E/E-architectures, guarantees timing and other critical requirements such as redundancy (IEEE 802.1CB) which relieves safety engineers from individual and inflexible solutions.

V. Conclusion

We presented our new experimental setup for benchmarking TSN time-triggered standard IEEE 802.1Qbv. The results show that it achieves lowest latency and jitter independent from the traffic load and priority of the competing unscheduled frames. It has been observed that deterministic behavior of the network stack software of the end-stations plays a significant role to reduce the number of outliers regarding the worst-case latency and jitter for critical periodic frames.

In a future work, we plan to apply our recently developed TSN graphical modeling tool [14] to automate the synthesis and configuration procedures. The presented experimental setup in this paper will be used for benchmarking and investigating the synthesis quality of the the modeling tool.

References

[1] Time-sensitive networking. (Date last accessed 20-August-2017). [Online]. Available: http://www.ieee802.org/1/pages/tsn.html

[2] H. T. Lim, D. Herrscher, and F. Chaari, "Performance comparison of ieee 802.1q and ieee 802.1 avb in an ethernet-based in-vehicle network," in *8th International Conference on Computing Technology and Information Management (ICCM)*, vol. 1, 2012.

[3] G. Alderisi, A. Caltabiano, G. Vasta, G. Iannizzotto, T. Steinbach, and L. L. Bello, "Simulative assessments of ieee 802.1 ethernet avb and time-triggered ethernet for advanced driver assistance systems and in-car infotainment," in *IEEE Vehicular Networking Conference (VNC)*, 2012, pp. 187–194.

[4] P. Meyer, T. Steinbach, F. Korf, and T. C. Schmidt, "Extending ieee 802.1 avb with time-triggered scheduling: A simulation study of the coexistence of synchronous and asynchronous traffic," in *IEEE Vehicular Networking Conference*, 2013, pp. 47–54.

[5] T. Steinbach, H. T. Lim, F. Korf, T. C. Schmidt, D. Herrscher, and A. Wolisz, "Beware of the hidden! how cross-traffic affects quality assurances of competing real-time ethernet standards for in-car communication," in *IEEE 40th Conference on Local Computer Networks (LCN)*, 2015.

[6] S. Thangamuthu, N. Concer, P. J. L. Cuijpers, and J. J. Lukkien, "Analysis of ethernet-switch traffic shapers for in-vehicle networking applications," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2015, pp. 55–60.

[7] D. Thiele, R. Ernst, and J. Diemer, "Formal worst-case timing analysis of ethernet tsn's time-aware and peristaltic shapers," in *IEEE Vehicular Networking Conference (VNC)*, 2015, pp. 251–258.

[8] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the symta/s approach," *IEEE Proc. - Computers and Digital Techniques*, vol. 152, no. 2, pp. 148–166, 2005.

[9] W. Steiner, "An evaluation of smt-based schedule synthesis for time-triggered multi-hop networks," in *31st IEEE Real-Time Systems Symposium*, 2010, pp. 375–384.

[10] L. Zhang, D. Goswami, R. Schneider, and S. Chakraborty, "Task- and network-level schedule co-synthesis of ethernet-based time-triggered systems," in *19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2014, pp. 119–124.

[11] S. S. Craciunas and R. S. Oliver, "Combined task- and network-level scheduling for distributed time-triggered systems," *Real-Time Systems*, vol. 52, no. 2, pp. 161–200, 2016.

[12] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in ieee 802.1qbv time sensitive networks," in *Proc. of the 24th International Conference on Real-Time Networks and Systems*, ser. RTNS '16. ACM, 2016, pp. 183–192.

[13] M. H. Farzaneh, S. Shafaei, and A. Knoll, "Formally verifiable modeling of in-vehicle time-sensitive networks (tsn) based on logic programming," in *IEEE Vehicular Networking Conference (VNC)*, 2016.

[14] M. H. Farzaneh, S. Kugele, and A. Knoll, "A graphical modeling tool supporting automated schedule synthesis for time-sensitive networking," in *22nd IEEE International Conference on Emerging Technologies And Factory Automation (ETFA)*, sep 2017.

[15] Ostinato network traffic generator. (Date last accessed 29-August-2017). [Online]. Available: http://www.ostinato.org/