International Journal of Research and Engineering
ISSN: 2348-7860 (O) | 2348-7852 (P) | Vol. 5 No. 9 | September-October 2018 | PP. 494-499
Digital Object Identifier DOI® http://dx.doi.org/10.21276/ijre.2018.5.9.1

Time-aware Traffic Shaper using Time-based Packet Scheduling on Intel I210

ORIGINAL ARTICLE

Author(s): [1]*Syed Sahal Nazli Alhady, [1]Wan Amir Fuad Wajdi Othman, [1]Aeizaal Azman Abd Wahab, [1]Aeizaal Azman Abd Wahab, [1]Por Yin Wong
Affiliation(s): [1]School of Electrical & Electronic Engineering, Universiti Sains Malaysia
Corresponding Author: sahal@usm.my

Abstract - By 2015, the Institute of Electrical and Electronics Engineers (IEEE) time-sensitive networking (TSN) task group has released several TSN standards. Amongst them is 802.1Qbv, also known as time-aware shaper, aiming to provide performance assurances of latency and delivery variation to enable applications in a TSN network. While there are several products and evaluation kits that employ 802.1Qbv in the market now, it is still not widely adopted yet due to the maturity of the standard. Hardware-enabled 802.1Qbv use hardware queues and timers to achieve accurate transmission of packets in the switch and bridge. This research aims to investigate the feasibility of using an existing end-station Ethernet controller, Intel I210, and its launch time control feature (commonly known as time-based packet scheduling) to shape traffic compatible to 802.1Qbv-enabled network bridges. A software solution is developed by implementing a software configurable gate-control list and employing open-source Linux RFC patches for per-packet transmit time specification. By configuring the kernel and mapping kernel-layer traffic classes to the hardware queues, packets can be transmitted out at precise times while attaching 802.1Q VLAN tags, required by bridges to identify packets. Through analysis, it is found that this solution will require an additional 30 µs transmit offset to be used effectively. That is 55% more time is needed to transmit a packet in a back-to-back connection and 17.6% on a 3-switch network to improve period peak jitter performance to just 8.9 µs compared to 1 ms on solutions that send packets out periodically using software sleep functions.

Keywords: *Time-sensitive networking, time-aware shapers, period jitter, latency*

I. INTRODUCTION

Ethernet has been widely used for various consumer applications due to its cost, availability and throughput capabilities. Recent interest has increased in using Ethernet for industrial and automotive applications where existing communication channels such as CAN and FlexRay are not capable of providing the bandwidth needed for infotainment applications and camera-based Advanced Driver Assistance Systems (ADASs) such as, Lane Departure Warning and Traffic Sign Recognition (IEEE, March 18 2016) (Bello, 2014). Consolidation of several automotive domains in a vehicle communication system by primarily using Ethernet would benefit manufacturers by reducing costs, improving manageability and reduce the infrastructure complexity (P. Meyer, 2013).

Several standards were added by the TSN (formerly AVB) task group including; 802.1Qbv, 802.1Qbu and 802.1Qca. These specifications provide a framework which industry players can enhance their Ethernet technology implementations to be used in time-critical scenarios and provide the timing guarantees needed for industrial and automotive applications. In particular, 802.1Qbv aims to solve the problem of interfering frames in a time-sensitive network by introducing time-aware shapers (IEEE, March 18 2016) (Bello, 2014). Time-aware shapers are a means to temporally isolate time-sensitive packet flows as they allow time-sensitive frames to egress from a bridge port at specific intervals.

Problem Statement

Conventional Ethernet is inadequate for real-time or industrial applications due to its non-deterministic behavior (Lee & Suk Lee, 2002). Under heavy loads, the 802.3 MAC may become unfair through due to interfering frames and phenomenon called the capture effect (Decotignie, 2005). In

a time-critical applications, Ethernet must deliver data, in a timely and dependable fashion.
The 802.1Qbv standard specifies how time-aware shapers are "smart shapers" that let frames out based on their size and the knowledge of expected times for time-sensitive frames. However, it is conventionally implemented at the hardware-level. Hardware-based implementations increase productions costs and time-to-market intervals. For an end-station, one possible alternative is by using existing hardware to emulate the behavior of the standard by using time-based packet scheduling.

Objective

This paper proposes a solution using an existing, commercially-available product, the Intel I210 Ethernet controller which has time-based packet scheduling capabilities to transmit packets. This paper aims to develop a functional prototype of a time-aware shaper using time-based packet scheduling with Intel I210 on a Linux platform; and to identify the transmit offset required to effectively implement time-based packet scheduling using a software gate-control list.

II. IMPLEMENTATION OF TIME-AWARE SHAPER IN A LINUX SYSTEM

System setup

The proposed solution is positioned as a user-space application within the system which includes 2 end-stations and 3 switches with connected as shown in Figure 1. Analysis is performed either with 2 end-stations connected directly or over 3 switches. A third host is used for generating interfering traffic.

System setup

The proposed solution is positioned as a user-space application within the system which includes 2 end-stations and 3 switches with connected as shown in Figure 1. Analysis is performed either with 2 end-stations connected directly or over 3 switches. A third host is used for generating interfering traffic.

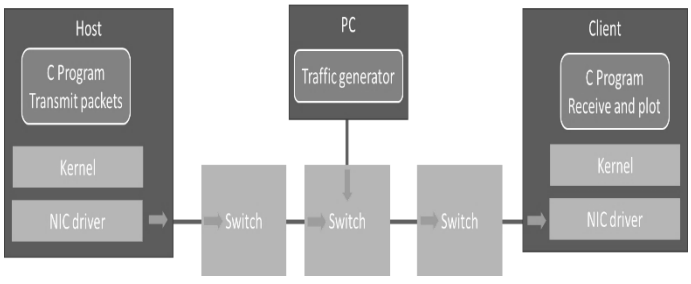

Figure 1 Full system setup and packet path from host to client. A PC generates interfering traffic.

In Figure 1, both end stations, client and host, are Linux 4.14 systems running on 4-cores at 1.6GHz and using the Intel I210 Ethernet controller. The host will construct the packet and payload. The Intel I210 Ethernet controller provides the time-based packet scheduling feature and also comes with hardware receive timestamping capabilities; which is why the same hardware is used as the client as incoming packets can be timestamped right at the moment it reaches its destination.

These two end stations are connected through 3 switches to introduce a network environment which uses 802.1Qbv for traffic shaping and prioritization. All 5 components are connected by a single Ethernet wire from one component to another, except in the case of the second switch which has 1 additional input port for traffic injection.

Each switch have specific gate-open intervals. In this case, each switch opens for 24 us and will do so once every 1 ms. It is the host application's responsibility to launch packets at specific intervals based on the switch's gate-open intervals. Transmitting packets to reach the switches right before the switch transmit time for that particular is key to achieving minimal latency and reducing time wasted waiting at the switch's egress queues.

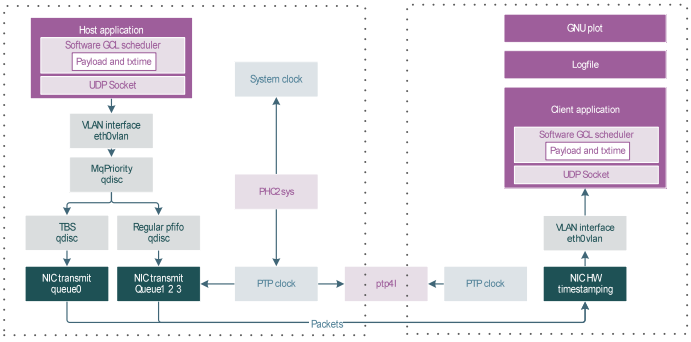

Figure 2 Points where timestamps are indicating.

Figure 2 shows the various software components employed throughout the network and host in order to transmit the packets precisely. In a network, each device operates in their own clock domain as the precision of each clock crystal differs from one device to another. A Linux utility (conforming to 802.1As) called "ptp4l" and "phc2sys" from "linuxptp" (Cochran, 2018) is used to synchronize each clock to the hosts' Ethernet controller clock. Using this utility, the same timestamp and clock domain can be maintained throughout the network accurate to several nanoseconds.

Packets are transmitted through user-defined VLAN (802.1Q) interfaces to attach VLAN headers to the packets. This allows the switches to identify the packets and its priority. Priorities can be mapped from the Linux system's

socket priorities, or traffic classes, to the VLAN priorities specified by the user. In this research, 3 distinct priorities are used; in descending orders of priority, time-critical traffic, interfering traffic and best-effort traffic.

Per-packet scheduling is a hardware feature available on the Intel I210 Ethernet controller. To schedule packets for transmission, a user will have to map the packet traffic class to the respective TBS Qdisc (Corbet, 2018). If the packet transmit time is placed onto the packet, the Ethernet controller will wait until the specified time before transmitting. In this research, a TBS Qdisc is used for both time-critical and best-effort traffic. For analysis purposes, a second host with Intel I210 Ethernet controller is used to timestamp packets as they arrive at the second host. This is useful for calculating latency and period jitter experienced by a receiving end-station.

Analysis 1: Performance over 3-switches

The first analysis is performed between 2 end stations with a direct connection to each other. This analysis is performed on systems running the proposed solution, and the regular non-TBS approach of transmitting packets. The two configurations are to send 480000 packets over 8 minutes to a client which will timestamp incoming packets as they arrive and store them in a file for analysis. The key metrics calculated from the data stored are period jitter and frame delay variance.

As the non-TBS configuration would not have packet transmit timestamps, this analysis is performed by storing the packet construction timestamp into the packet payload while the packet is being constructed at the host application. Using this packet construction timestamp and the receive timestamp, the client application calculates the total latency, frame delay variance and period jitter. The minimum, maximum and average values are presented in this paper.

Period jitter is measured to show how the output's periodicity is impacted when TBS is used. By calculating the difference between a packet's receive timestamp with the previous packet timestamp the period between packets is known. Since the ideal period is 1ms, period jitter can then be found as in equation 1 (Tranchemontagne, 2016).

$$Period\ Jitter = Real\ period - ideal\ period \quad (1)$$

The Inter-Packet Delay Variation (IPDV) or Frame Delay Variance (FDV) can be calculated by comparing a packet's latency with the previous packet's latency. This value can indicate how much a packet's latency is changing. Without TBS, this value should fluctuate significantly. Equations 2 to 4 (A. Morton, 2009) are used to calculate the FDV in this analysis.

$$Latency\ X = (receive\ timestamp\ B - packet\ construction\ timestamp\ B) - (receive\ timestamp\ A - packet\ construction\ timestamp\ A) \quad (2)$$

$$Latency\ Y = (receive\ timestamp\ C - packet\ construction\ timestamp\ C) - (receive\ timestamp\ B - packet\ construction\ timestamp\ B) \quad (3)$$

$$Frame\ delay\ variation\ (FDV) = latency\ Y - latency\ X \quad (4)$$

Using the frame delay variance in total transmission time and period jitter, both TBS and non-TBS applications can be compared equally in terms of its determinism. TBS is expected to perform better and this experiment will show by what margin.

Analysis 2: Minimum Transmit Offset

This analysis is performed between the host and client with a direct connection. Transmit offset is the time interval into the future the host application sets the packet transmit timestamp. Ideally, an application would set the timestamp as far into the future as possible, however that is not an opportunity every platform or system has. Theoretically, it is possible to transmit packets with minimal delay, however software timing variances may cause the packets to be transmitted out several nanoseconds to microseconds later, resulting in additional jitter. By comparing the receive timestamp and the packet construction timestamp, the total transmission time experienced by the packet is known. The total transmission time is compared between the two configurations over 8 minutes with 480000 packets.

Firstly a non-TBS system is used to obtain its total transmission time. Then, the TBS solution is used with the value obtained as transmit offset. However this value is not expected to use TBS effectively, there will be occasions where the packet misses the transmit time when it reaches the Ethernet controller and still get sent out anyway as there will be occasions where it was sent too late by the kernel. A difference of several nanoseconds or even microseconds, will increase peak-to-peak frame delay variance and period jitter. So the transmit offset value is incremented until the period jitter improves to the point similar to the results obtained in Analysis 1 and this will be the minimum time that is suitable to be used at the CPU's existing load and capability. This value is important because it indicates how much additional time is required in order to use TBS effectively in order to gain the benefits indicated by Analysis 1.

Analysis 3: Latency of High-priority Traffic during Traffic Interference

This analysis is to verify that high-priority packets are not interrupted and software gate-control list implementation is compatible with 802.1Qbv capable switches. This setup is similar to Analysis 1 except that the second switch being injected using a regular PC running a traffic generator which outputs interfering VLAN packets with a different VLAN ID and priority value set to 3. Every 1ms at the host, a TSN packet is transmitted and 0.5us later a BE packet is transmitted. Two packets are transmitted every 1ms. Only one host configuration with the TBS solution is used in this test. The analysis is compared with and without traffic injection. If the previous experiments were conducted successfully, this experiment will show the proposed solution is fully functional with physical 802.1Qbv-capable switches.

Expected results include the frame delay variance and period jitter or high-priority traffic remaining the same while low-priority traffic is delayed. Without TBS and using only software sleep functions, this would not be possible and TSN traffic would miss the time slices and get delayed because high-priority traffic is only processed during its own time slice.

III. RESULTS AND DISCUSSION

Analysis 1: Performance over 3-switches

Results are shown in Table 1. The peak-to-peak in the time between packets is observed to be higher than the solution without TBS. Without TBS, packets are being more inconsistently transmitted out. They have a peak-to-peak of almost 2 ms compared to only 0.017 µs of the TBS solution. While this outcome is expected, this experiment shows the margin of improvement when using time-based packet scheduling.

Without using time-based packet scheduling the solution is showing a maximum of additional 1 ms in time between 2 packets. This is highly undesirable in a time-sensitive environments. Key takeaways from this experiment shows that the solution is successfully developed to utilize the time-based packet scheduling feature. Period jitter performance is significantly improved compared to a solution which uses regular sendmsg() calls without the TBS feature.

Additionally, when not using TBS the average period is consistent, most of the data is. However there are 112 occasions, within the 480000 data collected, where the TSN packets are out of sync with the gate control lists. This causes the peak-to-peak jitter to spike by up to 1ms of delay. When a TSN packet misses the time slice in the switch it has to wait in queue until the next window opens, only then it will transmit out immediately along with another packet that was intended in that time slice.

This observation is crucial as it shows how without TBS a packet has a higher probability of missing the time slice and being delayed at the switches' transmit gates. In a TSN and real-time system, this behavior could be highly undesirable.

Analysis 2: Minimum transmit offset required to specify transmit time

This analysis involves observing the impacts of using TBS with minimum transmit offset. In the previous analysis, the program is given 500 µs to place the packet into the qdisc and the hardware queue, which in reality (as this analysis found) only requires about 60 µs to occur. The overhead is used to provide a buffer to eliminate the software variations when performing analysis.

Results shown in Table 2 display a consistent minimum jitter of 3 ns, this is normal in a system as a miniscule amount of jitter is still expected due to link properties. Without using time-based packet scheduling, there is an average total transmit time of 58 µs. Using this value as reference, the experiment starts from scheduling packets 65 µs to the future to 85 µs. This is because the program requires ~5 µs to wake up and start constructing the packet. The experiment is repeated with four other transmit offset values until a stable minimum period jitter is observed.

Table 1
Period jitter calculated using total transmission time and receive time

	Time since last packet (ns)		1ms Period Jitter (ns)		Frame Delay Variance (ns)	
	TBS	**No TBS**	**TBS**	**TBS**	**No TBS**	**TBS**
Average	1,000,000	1,000,000	25	486	25	1,771
Minimum	991,204	6,880	3	3	3	0
Maximum	1,008,884	2,000,088	8,884	1,000,088	8,884	1,001,304
Peak-to-peak	17,680	1,993,208				

Table 2
Period jitter calculated using total transmission time and receive time

	Total transmit time (ns)					
	No TBS	**TBS 65 µs**	**TBS 75 µs**	**TBS 85 µs**	**TBS 90 µs**	**TBS 95 µs**
Average	58,182	60,235	70280.32	80345	85,298	90,197
Minimum	15,925	53,555	53816	61213	67,133	70,048
Maximum	88,708	81,161	86044	84170	85,739	90,688
	Period Jitter (ns)					
Average	1,421	16	11	10	10	10

From Table 2, at 85us, significant period jitter improvement is observed. At 65 µs and 75 µs, the period jitter is not improved even though the application is configured to use time-based packet scheduling. While there is an improvement over the original configuration where no TBS is used, there is still a variance which occurs intermittently. Hence this experiment shows that even when TBS is enabled, there could still be intermittent jitters due to the system's software and hardware uncertainties and an additional overhead buffer time is required.

From this experiment, it is concluded that on a no-load end station environment, the recommended transmit offset is 95 µs. This is of course still subjective to the implementer and the hardware available as these results could easily be affected by the CPU and its processes. The higher amount of delay is always recommended where possible.

This result shows that this test methodology can be used to evaluate at which point a system can reliably send out packets with an effective and current TBS implementation. As explained in the previous paragraph, 95 µs is the best minimum transmit offset value for this configuration. This indicates that the system requires a minimum of 30 µs of transmit offset before TBS is used effectively by the system. This also shows that the TBS solution would take 55% longer in average total transmission time to effectively use TBS compared to non-TBS solutions in back-to-back connections. On a 3-switch setup that would be 17.6%.

Analysis 3: Latency during traffic interference

The last analysis shows how an interfering traffic injected to the 3-switch system will compromise low-priority traffic without affecting the high-priority traffic. Results tabulated in Table 3 show that with interference, BE packets have a much higher peak-to-peak total transmission time, more than 1ms more than TSN packets. This environment shows how in a real-world scenario a TSN traffic can be protected from latencies using 802.1Qbv switches and also demonstrates that the proposed solution is able to work with these switches with minimal input and modifications.

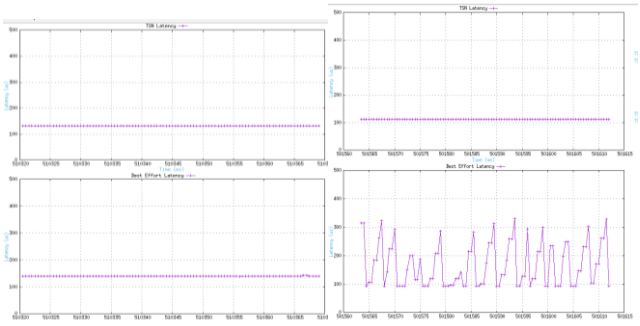

Figure 4-1 (Left) TSN and BE packets are sent without notable latency variances. (Right) BE traffic is shown to be affected by the interfering traffic, latency varies up to 200us more.

Figure 3 shows a graphical representation of the TSN and BE traffic latency. When traffic is introduced the low-priority traffic (BE) is delayed consistently by ~200 µs while the high-priority traffic is not affected at all by the traffic injected by the system. Key takeaway from this experiment includes the successful testing of the application on a network with traffic injected to the system, demonstrating the effectiveness of the time-aware shaper.

Table 3
Total transmission time with and without traffic interference

	With interference		No interference	
Total transmission time (ns)	**TSN**	**BE**	**TSN**	**BE**
Average	112,173	312,352	112,182	93,268
Minimum	111,549	92,168	111,588	91,986
Maximum	121,966	137,4825	120,641	102,849
Peak-to-Peak	10,417	1,282,657	9,053	10,863

IV. CONCLUSION

The proposed solution was successfully developed and tested on a 3-switch network. Packets are sent at their respective specified transmit times and is shown to be compatible with network switches enabled with gate-control.

On average, a minimum additional transmit offset of 30 µs is required for a host to schedule packets far enough into the future to effectively use time-based packet scheduling. This is 55% more time needed in a back-to-back connection while 17.6% more on a 3-switch network. Analysis showed that period jitter performance across switches peak at 8.9 µs compared to up to 1 ms by solutions using software sleep functions. Future works may involve developing the solution to conform closer to the 802.1Qbv specifications, specifically on handling transitions between different gate control lists. Combining with transmission selection algorithms from other specifications such as 802.1Qav credit-based shapers is also a possible avenue to provide users with more control over the transmission bandwidth.

V. ACKNOWLEDGMENT

The authors gratefully acknowledge use of service and facilities of their colleagues in Intel Microelectronics Malaysia, funded alongside by USAINS.

REFERENCES

[1] A. Morton, B. C. (2009, March). *Packet Delay Variation Applicability Statement.* Retrieved July 2018, from https://ietf.org/rfc/rfc5481.txt

[2] Bello, L. L. (2014). Novel trends in automotive networks: A perspective on Ethernet and the IEEE Audio Video Bridging. *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA).* Barcelona.

[3] Cochran, R. (2018, April). *Linuxptp*. Retrieved May 2018, from https://github.com/richardcochran/linuxptp

[4] Corbet, J. (2018, March). *Time-based packet transmission*. (Eklektix, Inc) Retrieved May 2018, from https://lwn.net/Articles/748879/

[5] Decotignie, J. D. (June 2005). Ethernet-Based Real-Time and Industrial Communications. *IEEE, vol. 93*(no. 6), pp. 1102-1117.

[6] IEEE. (March 18 2016). *"IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic" in IEEE 802.1Qbv-2015*. New York: IEEE Standards Association.

[7] Kyung Chang Lee, S. L. (2002). Performance evaluation of switched Ethernet for real-time industrial communications. *Computer Standards & Interfaces, 24*(5), 441-423.

[8] P. Meyer, T. S. (2013). Extending IEEE 802.1 AVB with time-triggered scheduling: A simulation study of the coexistence of synchronous and asynchronous traffic. *IEEE Vehicular Networking Conference,* , (pp. pp. 47-54.). Boston, MA, 2013, .

[9] Tranchemontagne, M. (2016, June). *Jitter Basics, Advanced, and Noise Analysis.* Retrieved July 2018, from https://www.ieee.li/pdf/viewgraphs/jitter_basics_advanced.pdf