Analysis and modeling of asynchronous traffic shaping in time sensitive networks

Zhou, Zifan; Yan, Ying; Berger, Michael Stübert; Ruepp, Sarah Renée

Published in:
Proceedings of 2018 14th IEEE International Workshop on Factory Communication Systems

Link to article, DOI:
10.1109/WFCS.2018.8402376

Publication date:
2018

Document Version
Peer reviewed version

Analysis and Modeling of Asynchronous TrafficShaping in Time Sensitive Networks

Zifan Zhou, Ying Yan, Michael Berger, Sarah Ruepp
DTU Fotonik
Technical University of Denmark
2800 Kongens Lyngby
zifz,yiya,msbe,srru@fotonik.dtu.dk

***Abstract*—Time Sensitive Network (TSN) is an attractive solution for time-critical frame transmission in data link layer. Traffic scheduling and shaping in TSN aim to achieve bounded low latency and zero congestion loss. TSN provides traditional frame switching with reliable Quality-of-Service (QoS) parameters, however, synchronization challenges, e.g. jitter, lost timing frames, clock inaccuracy, threaten the reliability of TSN network. Emerging Asynchronous Traffic Shaping (ATS) guarantees determinism for real-time applications, e.g. automotive and industrial control, while removing the dependence on synchronous communication. This paper focuses on performance evaluation of the recently proposed ATS amendment to the IEEE 802.1 standard, two approaches are discussed: Urgency-Based Scheduler (UBS) and Paternoster policing and scheduling. Models and simulations are carried out for evaluation and comparison. Statistics on the end-to-end delay, buffer usage and frame loss rate are collected to assess the scheduling performance. Results show that ATS achieves effective traffic shaping and switching without synchronous mechanisms, while there is an evident trade-off for using these specific algorithms.**

***Index Terms*—Traffic Scheduling, Time Sensitive Network, Asynchronous Traffic Shaping, Urgency-Based Scheduler, Paternoster Policing and Scheduling, Quality-of-Service.**

I. Introduction

One primary and fundamental feature of TSN is the synchronous timing used among all nodes and end stations within defined network scope. Relying on the identical timing notion, TSN time-aware traffic scheduling enhances frame transmission with high predictability, where operations are scheduled on the basis of synchronous timing. However, the timing mechanism raises stringent requirement on the precision of synchronization, since any timing misalignment possibly imposes failure to the whole network.

ATS provides alternative methods for transmission scheduling in TSN which is independent of timing synchronization and broad-wide schedule. As illustrated in Fig. 1, each switch receives time signaling from an independent clock, queues are controlled by an attached shaper, in which an ATS algorithm is implemented. In ATS, based on the two proposed solutions: UBS [1] and Paternoster [2], switches in TSN are still able to guarantee deterministic and relatively low delay for time-critical streams. The main challenges for ATS are concluded as: (1) real-time data flows are scheduled in separated nodes

978-1-5386-1066-4/18/$31.00 ©2018 IEEE

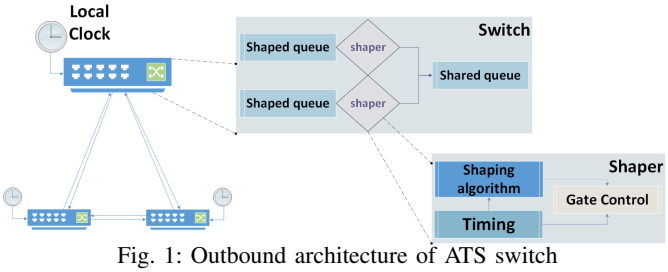

Fig. 1: Outbound architecture of ATS switch

with local time (2) flows are provided with different levels of service, but the service have to be consistent network wide (3) input flows have to fulfill certain required pattern.

Related work: Few paper has a summary of ATS algorithms and comparison through simulation. Some works in the literature elaborate the requirements and implementation of the real-time scheduled traffic in TSN networks. For instance, references [3] and [4] give examples of applying Ethernet and scheduled TSN to in-vehicle and wireless communication systems, emphasizing the importance of scheduling. The evaluation in [3] proves that Ethernet is able to transport the traffics mixing of different vehicle functions but scheduling is necessary in the overload situations. In [4], results show that it is difficult for conventional Ethernet to fulfill the jitter requirements of Common Public Radio Interface (CPRI), while this problem could be solved by implementing enhanced scheduled traffic. The paper [5] proposed simulations of scheduled traffic in Audio Video Bridging (AVB) network. However the simulation is only driven by strict priority and off-line scheduling, thus it lacks flexibility and is complex to add more flows. A prototype real-time Ethernet switch is proposed in [6], the switch provides real-time communication based on a time-triggered schedule. The switch supports frame transmission with a network-coherent time line and online administration control, and it consolidates synchronous and asynchronous transmissions in the same system, however the paper does not mention a situation where the switch faces with a synchronization failure. A hardware/software co-design concept of time-triggered controller is presented in [7], the controller could be implemented in the legacy Ethernet MACs, however based on the architecture description, it is obvious that synchronous traffic scheduling introduces high implementation complexity.

In this paper, two ATS approaches are investigated: UBS

and Paternoster. In order to achieve zero congestion loss and deterministic latency without using synchronous information, ATS introduces an additional layer with shaped egress queues to Ethernet switch and merges flows into the existing queue structure. ATS is independent of synchronous communication, thereby getting less complexity of implementation to achieve higher bandwidth utilization. The main contributions of this paper are summarized as follows:

- Elaborating the principles of ATS by designing accurate models and measuring the performance in different scenarios. All models are built in Riverbed modeler that describes network topology and functionalities.
- Analyzing the worst-case delay of ATS using mathematical expressions.
- Collecting and comparing the average per-hop delay, buffer usage and frame loss rate deriving from simulations. The models are set with different configurations to maximize scheduling efficiency.

II. System Model

A. UBS

In this section, the architecture and features of the system models are described. Firstly, UBS contains a hierarchical queuing framework, as the example shown in Fig. 2, the queuing framework contains:(1) per-flow shaped queues, which are assigned with fixed-priority according to the upstream sources (2) Shared queues, which are assigned with scheduler internal fixed priority. Shared queues merge streams that are shaped from several shapers with the same internal priority level, frames are transmitted from shared queue based on the First Come First Serve (FCFS) principle. Queuing schemes are defined as [1]: **QAR1**: frames from different transmitters are not allowed to be stored in the same queue. **QAR2**: frames from the same transmitter but not belong to the same priority in the transmitter are not allowed to be stored in the same queue. **QAR3**: frames from the same transmitter with the same priority in the transmitter, but not belong to the same priority in the receiver are not allowed to be stored in the same queue.

The purpose of queuing scheme is to enable a flexible configuration to provide optimized forwarding and to fulfill the requirements from individual network domain, basically, schemes QAR2 and QAR3 enable the separation of queuing on a priority base, which achieve that frames with higher priority can bypass the lower priority frames. The isolation of queuing prevents the propagation of malicious streams, assuring that the ordinary streams will not get interference, also enables flexible operations according to administrative requirements e.g. flow blocking or transmitter blocking. Based on the queuing schemes, the minimum number of shaped queues is limited by the number of ports in device. An n-port node needs at least $n-1$ mandatory queues to fulfill QAR1 scheme.

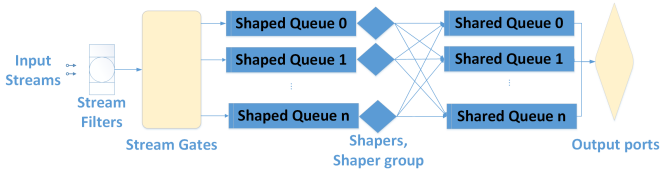

Fig. 2: Example of queue model in UBS

In order to achieve asynchrony, interleaved scheduling algorithms are developed for UBS. Two approaches deriving respectively from frame-by-frame leaky bucket algorithm and token-based leaky bucket algorithm [8] are introduced: Length-Rate Quotient (LRQ) and Token Bucket Emulation (TBE). Both algorithms enforce a constraint on the rates of input and output flow of the scheduler, on the basis of:

$$l_i(d) \leq b + d \cdot r \tag{1}$$

Where l_i donates the accumulative amount of transmitted bits, as a function of the time; b is the size of burstiness; d and r are the time duration and data rate.

These two algorithms are designed for ATS and rate limiting, however differ in concept: the principle of LRQ is to shape the traffic flow with a stable transmitting/leaking rate, regardless of the incoming flow pattern, it is able to converts bursty streams to stable and distributed output. Instead of scheduling synchronously on timing basis, each shaper keeps an eligibility time to indicate when next frame is allowed to be transmitted. For LRQ algorithm, the eligibility time is calculated as the quotient between the size of the previously transmitted frame and the reserved link rate of the particular class. The shaper updates the per-flow state every time a transmission is finished. Consequently, the LRQ shaper forces a time vacancy between frames and closes the gate for the shaped queue until next frame gets eligible for transmission, so that it keeps a stable average output rate.

The TBE shaper controls the traffic flow with an average rate while allows a certain level of burst. As long as sufficient number of "token" exists in the "bucket", a transmission can therefore get started immediately, otherwise, the eligibility time is calculated as the time it needs to accumulate enough "tokens". In principle, TBE algorithm increases the utilization of network resources than LRQ, especially in the cases where there is light traffic load, because in TBE algorithm, the time vacancy between two adjacent frames is partially removed. The status of gate relies on current number of "token" in the per-flow "bucket". If the length of one frame exceeds the current amount of "token", the shaper has to shut down the gate until the number of "token" increases with time and recovers to an enough amount.

B. Paternoster policing and scheduling

Paternoster algorithm is developed basing on a cyclically scheduling approach [9], it provides deterministic and bounded delay but removes the dependence on synchronous timing. The principle of Paternoster is implementing four cyclic output queues per class of service per port, each node and end station has local timing, and the time is counted in the unit of epoch duration τ. Four terminologies: *prior*, *current*, *next* and *last* are used to describe all epochs and cyclic queues, Table I illustrates the mechanism.

Every epoch has an associated *current* queue, all the incoming frames will be directed to *current* queue until it gets full. Otherwise, frames are forwarded to the *next* and *last* queue till next epoch starts. Frames are abandoned if all three queues get full with reserved flows. At the output ports, only the *current* queue works as outbound queue per epoch, which means the *current* queue is able to transmit and receive frames simultaneously. The implementation requires a local clock instead of synchronous signaling between adjacent nodes, due to the flexibility of more sparse queuing scheme.

The length of epoch of each traffic class remains its consistency within the defined network scope. Higher-priority flows are assigned with a shorter epoch to ensure less delay bound. Transmission of best-effort flows only fills the remaining bandwidth left from reserved flows. The best-effort frames will be dropped if the anticipated transmission time is beyond the current epoch. In principle, the length of τ should be configured to long enough for all reserved transmission and at least one best-effort frame with maximum size.

Compared with synchronized scheduling[9][10], Paternoster sacrifices some of the delay predictability but removes the synchronous timing signaling. Meanwhile, it reduces the lower bound of delay and distributes received frames to four queues, which provides similar scheduling performance with synchronized schedule and simplifies the implementation. From the perspective of buffer usage, the division of queuing in Paternoster offers more available storage resources, thus guarantee a lower frame loss rate compared with conventional cyclic queuing and forwarding[9].

III. Modeling and Mathematically Analyzing Asynchronous Traffic Shaping

In this section, simulation models of UBS and Paternoster are proposed to run in diverse scenarios of simulation environment. Giving sets of different configurations to the models and relevant results with scheduling performance will be collected. The majority of the simulation model is divided into four domains: network, node, process and external system, first three domains focus on different levels that identified in a real system, the external system domain focuses on necessary communication devices for simulations.

The simulation model contains a two-level process architecture: root process and child processes, a root process generates a series of child processes representing the per-flow shaper associated with each shaped queue. Therefore the root process is able to monitor and evaluate the state of all per-flow shapers, for instance, it terminates a child process when there are no more queued frames, and generates a new process once frames arrive. The following formulas give the worst-case per-hop delay of LRQ and TBE algorithms:

TABLE I: Timing and queuing in Paternoster

Epoch \ Queue	Queue0	Queue1	Queue2	Queue3
Epoch0	prior	current	next	last
Epoch1	last	prior	current	next
Epoch2	next	last	prior	current
Epoch3	current	next	last	prior
Epoch4	prior	current	next	last

$$\textbf{LRQ}: d_{LRQ} \leq max_{j\in I}(\frac{\hat{b}_H + \hat{b}_{C(j)} + \hat{l}_L}{r - \hat{r}_H} + \frac{\hat{l}_j}{r}) \quad (2)$$

$$\textbf{TBE}: d_{TBE} \leq max_{j\in I}(\frac{\hat{b}_H + \hat{b}_{C(j)} + \hat{b}_j - \check{l}_j + \hat{l}_L}{r - \hat{r}_H} + \frac{\check{l}_j}{r}) \quad (3)$$

Where *H,L* denotes all flows in higher and lower priority queues compared with flow *j*. *C* denotes all flows in the same priority queues with flow *j*. *I* represents all flows. And l, b, r represent the frame size, burst size and rate, respectively.

The end-to-end queuing delay of Paternoster is independent of the network topology and interfering traffics, the primary factor that bounds the delay is the duration of cycle epoch τ. The best case of end-to-end delay occurs when frames are forwarded from and to *current* queues in all relays, accordingly, the waiting time in queues is negligible. The minimum end-to-end delay depends on the number of hops (h) and processing time. The worst case caused by the situation where all three queues - current, next and last are assigned fully with frames. Thus per-hop queuing delay increases to:

$$d_{P\_hop} \leq (Q-1)\cdot\tau \quad (4)$$

Where Q denotes the total number of queues, then end-to-end queuing delay becomes

$$d_{P\_ETE} \leq (Q-1)\cdot\tau\cdot h \quad (5)$$

IV. Simulation Results

In this section, the simulation results of ATS are given. The work is carried out as an one-hop transmission, statistics are collected from different perspectives to measure the performance of the scheduling approaches mentioned above. Simulations parameters are implemented as in Table. II. In this paper, the main concern is to simulate different working statues for the scheduler, thus all the values are taken based on simulation requirements instead of real-world use cases.

Firstly, the average frame loss rate comparison among Paternoster with different epoch length and UBS are given in Table III. Since all devices have limited storage space and each traffic class is assigned with dedicated bandwidth, excess frames over the upper limit will be dropped. Independent to the algorithm, the input flows to UBS follow the leaky bucket constraint, thus LRQ and TBE scheduling perform identity frame-loss feature as shown in the first row.

As traffic intensity keeps increasing, the loss rate also constantly increase for all schedulers. The comparison shows that under the same input intensity, UBS has relatively lower loss rate, while the rate of Paternoster is related to the epoch

TABLE II: Simulation Parameters

Parameters	Value
Frame size	$720 - 1328$ bits
Reserved bandwidth	5.76 Mbps
Link rate	100 Mbps
Bandwidth of input flow	$4.096 - 20.48$ Mbps
Paternoster epoch length	$0.01, 0.005, 0.0025$ second

length (τ): shorter length means less storage space per epoch thus causing higher loss rate, the reserved bandwidth of each flow is calculated as : $3 \cdot \tau \cdot datarate$.

Table IV shows the average number of frames that are stored in the queue during simulation time. Within schedulers, the forwarding time of one frame depends on frame size, data rate also arrival and departure time of adjacent frames for LRQ algorithm. From the results: when the bandwidth of input flow is less than reserved level (input intensity = $4.096, 5.12Mbps$), UBS/LRQ has the most queued frames on average while Paternoster A with the longest epoch length has the least. As outlined above, the storage of frames in Paternoster schedulers are related with epoch length. Compared with Paternoster B and C, A is able to accommodate and forward more frames during one epoch, thus A has the least queued frames. Since UBS/LRQ scheduler enforces a waiting time after each transmission, the forwarding efficiency is lower than others.

On the other hand, when the bandwidth of input flow is equal or greater than reserved level (input intensity $\geq$ $5.69Mbps$), Paternoster C has the least number of queued frames: according to Table III, C discards most frames among all schedulers under the same condition, moreover, it iterates more epoch update during the entire running time, which means more forwarding operations are executed.

Table V lists the average delay measurement comparison. The variation of the delay statistics conforms with that of average number of queued frames. Paternoster A has the shortest delay when the input intensity is less than reserved, because Paternoster with longer epoch length enables more forwarding operations, also less frames are dropped due to bandwidth limitation, while Paternoster C performs faster operations when the input overflows. The average delays of all Paternoster schedulers keep increasing with input intensity.

The average delay of LRQ and TBE increases with the input traffics before overload, however, because of the linearly increasing feature of the Leaky Bucket Constraint, LRQ and TBE schedulers allow more transmissions of frames with smaller size under overload environment, thus the per-hop delay decreases sharply with the increase of input intensity.

TABLE III: Frame loss rate (in percentage) comparison

Scheduler \ Intensity(Mbps)	4.096	5.12	5.69	5.78	5.85	6.02	6.4	6.83	7.31	10.24	20.48
TBE,LRQ	0	0	0	0.51	1.6	4.27	9.63	15	10.16	42.06	69.63
Paternoster A(τ = 0.01s)	0	0	0.36	1.56	2.61	5.38	10.93	16.52	22.07	44.3	72.16
Paternoster B(τ = 0.005s)	0	0	1.27	2.55	3.55	6.23	11.73	17.23	22.72	44.83	72.41
Paternoster C(τ = 0.0025s)	0	0.07	3.22	4.51	5.45	7.99	13.36	18.73	24.15	45.8	72.9

TABLE IV: Average number of queued frames comparison

Scheduler \ Intensity(Mbps)	4.096	5.12	5.69	5.78	5.85	6.02	6.4	6.83	7.31	10.24	20.48
LRQ	0.9	3.9	44	449	167	83	42	27	18	9	4.3
TBE	0.05	0.42	36	420	158	80	31	29	15	0.07	0.06
Paternoster A(τ = 0.01s)	0.04	1.5	54	83	96	105	109.5	110.6	110.8	111	110.8
Paternoster B(τ = 0.005s)	0.12	2.5	33	39	44	50	54.4	54.4	54.5	54.5	54.4
Paternoster C(τ = 0.0025s)	0.3	4.1	16.8	18.6	19.5	22.3	24.5	25.2	25.8	26.25	26.2

TABLE V: Average per-hop delay (millisecond) comparison

Scheduler \ Intensity(Mbps)	4.096	5.12	5.69	5.78	5.85	6.02	6.4	6.83	7.31	10.24	20.48
LRQ	0.23	0.76	8	79	28	15	7.5	4.7	3.4	1.75	0.73
TBE	0.011	0.085	6.8	75	28	14.3	5.4	5.1	3	0.0084	0.01
Paternoster A(τ = 0.01s)	0.0108	0.045	5.7	10.7	12.7	14.5	15.5	16	16	17.5	18.87
Paternoster B(τ = 0.005s)	0.0135	0.13	3.8	5.18	5.8	6.8	7.5	7.9	8.1	8.7	9.4
Paternoster C(τ = 0.0025s)	0.026	0.375	2.2	2.55	2.75	3.15	3.81	3.84	3.9	4.3	4.7

V. Conclusion

ATS, as an emerging technology in TSN, aims at achieving high scheduling performance without using synchronous timing. This paper provides simulations and solid results of ATS algorithms focusing on queuing and forwarding process in terms of average per-hop delay and frame loss etc.

From an one-hop case, the simulation results show the comparison between two approaches and indicate the features of the schedulers with different input intensity. UBS/LRQ does not allow any bursty transmission in the cost of longer average delay and inefficient bandwidth utilization under light loaded situations; UBS/TBE has improved performance over UBS/LRQ, due to the feature that token bucket algorithm allow a certain level of bursty transmission, while the two algorithms have similar average delay performance and buffer usage when the amount of input traffic is around the reserved level. Compromising between delay and frame loss rate is emphasized for Paternoster scheduler, reasonable arrangement of epoch length could fulfill configuring requirements without using time synchronization.

Taking realistic traffic parameters into consideration for simulations and setting up more complex scenarios, e.g. multi-hop transmission and introducing interfering flows could be added to future works.

Acknowledgment

This work has been partially supported by the ERAN project founded by Danish Innovation Foundation

References

[1] J. Specht and S. Samii, "Urgency-based scheduler for time-sensitive switched ethernet networks," in *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*, 7 2016, pp. 75–85.

[2] M. Seaman, "Paternoster policing and scheduling," *IEEE 802.1Qcr*, 3 2017.

[3] H.-T. Lim, L. Völker, and D. Herrscher, "Challenges in a future ip/ethernet-based in-car network for real-time applications," in *Proceedings of the 48th Design Automation Conference*. ACM, 2011, pp. 7–12.

[4] T. Wan and P. Ashwood-Smith, "A performance study of cpri over ethernet with ieee 802.1 qbu and 802.1 qbv enhancements," in *Global Communications Conference (GLOBECOM), 2015 IEEE*. IEEE, 2015.

[5] G. Alderisi, G. Patti, and L. L. Bello, "Introducing support for scheduled traffic over ieee audio video bridging networks," in *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on*. IEEE, 2013, pp. 1–9.

[6] R. G. V. dos Santos, "Enhanced ethernet switching technology for adaptive hard real-time applications," Ph.D. dissertation, Universidade de Aveiro (Portugal), 2011.

[7] F. Groß, T. Steinbach, F. Korf, T. C. Schmidt, and B. Schwarz, "A hardware/software co-design approach for ethernet controllers to support time-triggered traffic in the upcoming ieee tsn standards," in *Consumer Electronics–Berlin (ICCE-Berlin), 2014 IEEE Fourth International Conference on*. IEEE, 2014, pp. 9–13.

[8] H. Zhang and D. Ferrari, "Rate-controlled service disciplines," *Journal of high speed networks*, vol. 3, no. 4, pp. 389–412, 1994.

[9] "Ieee standard for local and metropolitan area networks–bridges and bridged networks–amendment 29: Cyclic queuing and forwarding," *IEEE 802.1Qch-2017*, pp. 1–30, June 2017.

[10] "Ieee standard for local and metropolitan area networks – bridges and bridged networks - amendment 25: Enhancements for scheduled traffic," *IEEE Std 802.1Qbv-2015*, pp. 1–57, March 2016.