Insight into the IEEE 802.1 Qcr asynchronous traffic shaping in time sensitive network

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

Published in:
Advances in Science, Technology and Engineering Systems Journal

Link to article, DOI:
10.25046/aj040128

Publication date:
2019

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Zhou, Z., Berger, M. S., Ruepp, S. R., & Yan, Y. (2019). Insight into the IEEE 802.1 Qcr asynchronous traffic shaping in time sensitive network. Advances in Science, Technology and Engineering Systems Journal, 4(1), 292-301. https://doi.org/10.25046/aj040128
Insight into the IEEE 802.1 Qcr Asynchronous Traffic Shaping in Time Sensitive Network

Zifan Zhou, Michael Stübert Berger, Sarah Renée Ruepp, Ying Yan
Department of Photonics Engineering, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark

ABSTRACT
TSN is an attractive solution for latency-critical frame transmission built upon IEEE 802 architecture. Traffic scheduling and shaping in TSN aim to achieve bounded low latency and zero congestion loss. However, the most widespread solution (i.e. Time-Aware Shaper) requires a network-wide precision clock reference and only targets on cyclical traffic flows. This paper focuses on the performance evaluation of the ATS, which applies shaping algorithm to any flows and requires no clock reference. Simulations are proposed for evaluation and comparison. Metrics including end-to-end delay, buffer usage and frame loss rate are collected to assess the shaping 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.

1 Introduction
To facilitate the information exchange in the network world, diverse devices and network technologies were introduced over the years, the communication has now become more comprehensive and autonomous. The increase of interconnecting level within the network system cause an explosive growth of network traffics. The utilization of Internet of Things (IoT) and Cyber Physical System (CPS) in industrial domain bring about the concept of Industry 4.0, the industrial networks are now a mixture of field bus system, Ethernet approaches and wireless solutions [1]. In mobile communication networks, centralized baseband processing and cloud service, i.e. Centralized-Radio Access Network (CRAN) and Cloud-Radio Access Network (Cloud-RAN) are introduced to the access network between devices and radio transceiver [2][3], regarding as a typical approach to achieve Fifth Generation (5G) mobile network [4]. Accordingly, distributed data transmission turns into be more integrated, it becomes more challenging to fulfill the stringent transmitting requirements for time-sensitive flows.

Having the development of the network applications in mind, it is a prerequisite condition to build a fundamental transmission network which is capable of providing appropriate services. The ongoing works of Time Sensitive Network (TSN) aim to enhance standard Ethernet to fulfill the need for deterministic, reliable and efficient communication. TSN comprises a set of standards, as a part of work in the IEEE 802.1 working group, this work originated from the Audio Video Bridging (AVB) standards, it provides services of bridging, network management and building real-time transmission over Ethernet within a LAN or MAN domain, different features are defined in separated standards to ensure the performance from several perspectives.

One of the primary features in TSN is the timing synchronization, where all nodes and end stations within a TSN domain are set by a common timing signal. Relying on the synchronization, general traffic scheduling and shaping in TSN enhances the delivery of frame with high predictability, namely, the time instance when each transmission occurs is guaranteed in the network. However, it raises strict requirement on the precision of global timing mechanism - any timing misalignment possibly imposes failures to the network. Thus high complexity is required for implementation and maintenance. Time-triggered scheduling and shaping also requires a consistent and recurrent egress gate behavior and it has to be synthesized before flows are transmitted from the source, thus the synchronous scheduling and shaping procedure only applies to a subset of traffic flows arrive periodically.

The Asynchronous Traffic Shaping (ATS) project is created by the IEEE 802.1Qcr working group [5]. It is an approach designed without any dependency on network-wide planning, cycle synchronization or time-triggered actions, while the purpose is to provide
deterministic and relatively low transmission delay for
general time-sensitive flows and has no requirements
on the traffic pattern. An Urgency-Based Scheduler
(UBS) solution was proposed at an early stage of de-
development [6], which contains two algorithms based
on the Rate-Controlled Service Disciplines (RCSDs)
[7], besides, a concept of Paternoster scheduling is also
included in the 802.1 Qcr web page [5]. At the time of
writing, a new ATS algorithm is included in the latest
version of the draft standard, all of these algorithms
are involved in this paper. The main contributions of
this paper are summarized as follows:

- Elaborating the principles of ATS by designing
relatively accurate models and measuring the
performance in simulation scenario. All mod-
 els are built in software modeler that describes
network topology and functionalities.

- Collecting the average per-hop delay, buffer us-
age and frame loss rate deriving from simula-
tions. Based on the results, comparisons are done
between all ATS approaches, also the models
are set with different configurations to optimize
scheduling utilization.

2 Related work

To the best of our knowledge, few paper have perfor-
mane evaluation of ATS algorithms through software
simulation. The synchronous scheduling has been men-
tioned in many works, most of them indicate that it is
essential to apply scheduling to provide time-sensitive
services. On the other hand, existing researches on
ATS accomplish the theoretical analysis on the features
of ATS, in this section, a few works that are relevant
with the analysis, measurement and modeling of TSN
scheduling are included.

2.1 Researching on traffic scheduling in
TSN

Some works in the literature elaborate the require-
ments and implementation of the real-time scheduled
traffic in TSN networks. For instance, references [8]
and [9] give examples of applying Ethernet and sched-
uled TSN to in-vehicle and wireless communication
systems, emphasizing the importance of scheduling.
The evaluation in [5] proves that Ethernet is able to
transport the traffics mixing of different vehicle func-
tions but scheduling is necessary in the overload situ-
ations. In [9], results show that it is difficult for con-
ventional Ethernet to fulfill the jitter requirements
of Common Public Radio Interface (CPRI), while this
problem could be solved by implementing enhanced
scheduled traffic.

Multiple time-sensitive flows usually co-exist in
the same network, taking into account the mutual interfer-
ence among these flows, G. Alderisi et al proposed a
temporal isolation of flows that refer to the same traffic
class[10]. The work comprises of simulations of sched-
uled traffic in Audio Video Bridging (AVB) network,
the traffic scheduling is driven by strict priority and off-
line configuration. The results show that the tempo-
al isolation between Scheduled Traffic (ST) and other
traffic classes guarantees low and predictable latency
for ST class.

The procedure of configuring synchronous schedul-
ing is considered to be time-consuming, in [11], a
graphical network modeling tool was designed to auto-
mate synthesis of gate control list in TSN scheduling, it
is able to convert user-defined flow, topology and Qual-
ity of Service (QoS) to the constraints for synthesis.
The tool applies object-oriented modeling, logic pro-
gramming and Satisfiability Modulo Theories (SMT)
to achieve automation of synthesis, it also simplifies
the procedure for configuration. In [12] and [13], two
performance analysis of real-time Ethernet are intro-
duced. In [12], the evaluation of AVB standards are
conducted in a simulation environment, and it shows
the interfering flows have limited influence on the
latency of AVB flows and the latency of the flows are
constrained by size of payload. And in [13], an ex-
perimental setup is proposed to analyze the latency
and jitter of synchronous traffic scheduling in TSN.
The results in this work also indicates that the latency
and jitter of scheduled traffic are independent from
unscheduled traffics, besides, it is worth mentioning
that the network stack software of end station has a
strong effect on the behavior of critical periodic traffics
in such experimental environment.

A prototype real-time Ethernet switch is proposed
in [14], the switch provides real-time communication
based on a time-triggered schedule. The switch sup-
ports frame transmission with a network-coherent time
line and online administration control, and it enforces
isolation of three different traffic classes so as to pre-
vent any interference from non-time-sensitive traffic.
A hardware/software co-design concept of Ethernet
controller is presented in [15], the controller is par-
titioned into communication and application compo-
nents, dedicated modules are allocated to critical trans-
mission to fulfill timing requirement and reduce the
load of microcontroller. The results show that the
hardware extension of Ethernet controller significantly
reduces the working load of software communication
stacks, especially with a mixture of scheduled and non-
scheduled traffics. With the controller, the jitter of
time-scheduled transmission can be improved sharply.

2.2 Relevant work on ATS

One of the most significant metrics to evaluate TSN
networks is the worst-case delay, the calculation of ATS
delay bounds in [6] does not account for accumulative
burstiness of the same traffic class. E. Mohammadpour
et al. proposed a performance evaluation of ATS and
Credit Based Shaper (CBS) in [16]. Firstly, the delay
calculation included in [6] is extended in this paper, in
regard to generic features of TSN. Moreover, backlog
bounds of buffers are given based on network calcu.
A relatively stringent end-to-end latency bound of TSN is computed instead of adding up the bounds calculated at every switch on the path. The work increases the tightness of upper bound of end-to-end delay in TSN network and benchmarks a theoretical analysis for such a network.

In order to achieve the flexibility of ATS by aggregating flows and assign separate priority level at each hop, the synthesis of ATS becomes a more complicated process, in terms of forwarding flows to queues and assigning priority levels to queues. In [17], Johannes and Soheil present a SMT based solution along with a topology rank, cluster based heuristic of this method. The work demonstrate that with the SMT method, it is feasible to find an existed solution of synthesis and the Topology Rank Solver (TRS) heuristic reduces the computational effort to achieve the method significantly.

3 Asynchronous Shaping

3.1 Modules and Architecture

As depicted in Figure 1, the switch with asynchronous shaping implements an independent clock that does not synchronize with other switches. For a given queue that supports asynchronous shaping, flows sent out from the queue are shaped by a bonded shaper, which calculates eligibility time and assigns the time to frames, the time are then used for traffic regulation by the transmission selection algorithm, a frame is eligible for transmission if the assigned eligibility time is less than or equal to the current time. The flow shaping actions are implemented through an open/closed gate control instance attaching to the queue: the gate for the specific queue will be opened when the frame in that queue is eligible to be transmitted. The algorithms used for calculating eligibility time is described in the following sections.

Transmission latency usually comprises of link propagation delay and intermediate devices delay, a desirable queuing discipline should reduce effectively the storage delay in the devices. In the UBS proposal, a queuing hierarchy is introduced to the ATS pipeline, as Figure 2 shows, the queuing framework contains: (1) per-flow shaped queues, which are classified according to the identification of the frame, e.g. flow ID, traffic class and flow destination address (2) Shared queues, which merge frames with the same internal priority level and egress port but are transmitted from different shaped queues, in shared queue, frames are transmitted based on the First Come First Serve (FCFS) principle.

Queuing schemes for input frames are defined as [6]:

**QAR1**: frames from different transmitters are not allowed to be stored in the same shaped 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 shaped 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 shaped queue.

The main purposes of implementing these queuing schemes are to enable flexible configuration among different flows, and to fulfill network services from various network domains and administrators.

According to 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.

Basically, scheme QAR2 and QAR3 achieve the separation of flows on a priority base, which enable frames with higher priority can bypass the lower-priority frames, to ensure the transmission delay of high-priority flows will not be affected by interfering flows. The isolation of queuing brings benefit to prevent the propagation of malicious flows, assuring that the ordinary flows will not get influence, it also enables flexible operations according to administrative requirements e.g. flow blocking or transmitter blocking. Considering asynchronous shaping, due to the classification of frame queuing, the shaper is able to conduct more granular operations based on larger scale of in-queue frame state. Thus ATS could lessen the queuing time of time-sensitive frames and achieve fast forwarding.

3.2 UBS algorithms

In order to achieve asynchronous shaping and keep low storage delay, an interleaved scheduling algorithm is introduced in UBS. Two approaches deriving respec-
tively from frame-by-frame leaky bucket algorithm and token-based leaky bucket algorithm [27] are introduced: Length-Rate Quotient (LRQ) and Token Bucket Emulation (TBE). Both algorithms enable the shaper with a constraint on the rate of input and output flow as:

\[ l_i(d) \leq b + d \cdot r \]  (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, the constrain can be regarded as a benchmark for mixed traffic network without interaction among different flows.

LRQ and TBE are both designed for asynchronous shaping in TSN, however differ in the shaping 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 convert bursty flow or flows with any pattern to stable, constant and burstiness; transmitted bits, as a function of the time;

\[ \text{Average output rate} \]

For TBE, the eligibility time is calculated as the time it needs to accumulate enough "tokens", as shown in the pseudo code:

**Algorithm 2 TBE algorithm Pseudo code**

1: /* Initialization */
2: for i in (0 : I) do
3:   flow[i].timestamp = 0
4: end for
5: /* Shaping */
6: while true do
7:   if queue[i].size > 0 then
8:     f = queue[i].head
9:     L = f.length
10:    i = f.index
11:   t_i = flow[i].timestamp
12: end if
13: if t_now >= t_i then
14:   output f from queue
15:   t_i = t_now + (L/r_i)
16: end if
17: end while

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. For TBE, the eligibility time is calculated as the time it needs to accumulate enough "tokens", as shown in the pseudo code:

**Algorithm 1 LRQ algorithm Pseudo code**

1: /* Initialization */
2: for i in (0 : I) do
3:   flow[i].timestamp = 0
4: end for
5: /* Shaping */
6: while true do
7:   if queue[i].size > 0 then
8:     f = queue[i].head
9:     L = f.length
10:    i = f.index
11:   t_i = flow[i].timestamp
12: end if
13: if t_now >= t_i then
14:   output f from queue
15:   K_i = min(Burstsize, K_i + (t_now - t_i) * flow[i].bitrate) - L
16:   flow[i].timestamp = t_i
17:   flow[i].token = K_i
18: end if
19: end while

In principle, TBE algorithm could increase the utilization of network resources than LRQ, especially in the case when the network is lightly loaded, since in TBE algorithm, the spacing time between two adjacent frames is not added every times, unless the token level of the flow is less than the pending frame.

Accordingly, the state of output gate relies on current number of "token" in the per-flow "bucket": if the length of pending frame exceeds the current amount of "token", the shaper has to shut down the gate until the number of "token" increases with time and accumulates to an enough amount. Therefore, the TBE shaper allows a limited extent of bursty flow when the number of token is sufficient.

### 3.3 ATS algorithm

In the recently proposed draft of ATS standard[5], a new shaping approach is included. Basically, the approach is also derived from Leak Bucket algorithms, including the concept of token bucket which is used to constrain the output rate of flows, preventing bursty flows spreading along the path. A local system clock function determines the selectability time per frame, which is the time when the frame is queued and available for transmission selection. All frames that reach their selectability time are selected for transmission in ascending order of the assigned eligibility times. Any frame may experience an additional, non-negative processing delay between its arrival time and its selectability time. This delay may vary per frame, thus
Algorithm 3 ATS algorithm Pseudo code

1: /* Initialization */
2: \( T_{\text{eligibility}} = 0 \)
3: \( T_{\text{bucketFull}} = 0 \)
4: \( T_{\text{groupEligibility}} = 0 \)
5: \( T_{\text{bucketEmpty}} = -(\text{burstSize}/\text{rate}) \)
6: /* Frame Processing */
7: \( D_{\text{lengthRecovery}} = \text{frame.length}/\text{rate} \)
8: \( D_{\text{emptyToFull}} = \text{burstSize}/\text{rate} \)
9: \( T_{\text{shaperEligibility}} = T_{\text{eligibility}} + D_{\text{emptyToFull}} \)
10: \( T_{\text{bucketFull}} = T_{\text{bucketEmpty}} + D_{\text{lengthRecovery}} \)
11: \( T_{\text{eligibility}} = \max(T_{\text{arrival}}, \)
12:     \( T_{\text{groupEligibility}}, \)
13:     \( T_{\text{shaperEligibility}}) \)
14: if \( T_{\text{eligibility}} \leq (T_{\text{arrival}} + \text{MaxTime}/1.0e9) \) then
15:     \( T_{\text{groupEligibility}} = T_{\text{shaperEligibility}} \)
16:     \( T_{\text{bucketEmpty}} = (T_{\text{eligibility}} < T_{\text{bucketFull}}) ? \)
17: else
18:     Discard(frame);
19: end if

The bucket full time is the time instant when the bucket is full with tokens, the size of bucket is equivalent to the burst size, on the contrary, bucket empty time is the time when there are no tokens existing in the bucket. The initial bucket empty time should be at least empty to full duration before the initial bucket full time. Basically, the empty to full duration is the duration needed to fill up the bucket with tokens from empty to full by the committed information rate. The length recovery duration denotes the duration that the tokens are accumulated by a number equaling to the length of the frame.

Considering a single shaper, the shaper eligibility time is the time when the number of tokens in the bucket is more or equal to the frame size. Taking into account a group of shapers within the same shaper class, the group eligibility time means the most recent eligibility time from the previous frame processed by the shaper in the same class. Max residence time is a parameter used to limit the time a frame residing in one node, a frame is valid only within the Max residence time.

As the code indicates, the calculation of eligibility time of the frame strongly depends on the size of last transmitted frame and the arrival time of itself. Different with the TBE algorithm, the eligibility time is not directly calculated from number of tokens in the bucket, instead, the bucket full time and bucket empty time are considered. The ATS algorithm also allows a certain scope of bursty flows, while for oversized flow bulk, the shaper will still limit the amount of outputting flow to avoid accumulating bigger flow bulk in the downstream node.

3.4 Paternoster queuing and scheduling

Paternoster algorithm is developed based on a cyclically scheduling approach, Cyclic Queuing and Forwarding (CQF) [8], it provides deterministic and bounded delay but removes the dependence on synchronous timing. The principle of Paternoster is to implement four cyclic egress 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 \( \tau \). Four terminologies: prior, current, next and last are used to describe all epochs and cyclic queues, Table 1 illustrates the mechanism:

| 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 |

Every epoch has an associated current queue, all incoming frames will be directed to the same current queue during one epoch unless the queue gets full, the following frames arrive at the same epoch are forwarded to the next and last queue as far as next epoch starts. Frames will be dropped if the volume of frames exceed reserved storage in all three queues. At the egress ports, only the current queue works as out-bound queue per epoch, which means only the current queue is allowed to transmit and receive frames simultaneously.

The length of epoch of each traffic class remains its consistency within the defined network. 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 \( \tau \) should be configured to long enough for all reserved transmission and at least one best-effort frame with maximum size.

Compared with synchronous scheduling, 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 of synchronization. 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 CQF.
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 $\tau$. 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:

$$d_{P,\text{hop}} \leq (Q - 1) \cdot \tau$$

(2)

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

$$d_{P,\text{ETE}} \leq (Q - 1) \cdot \tau \cdot h$$

(3)

Therefore, in Paternoster scheduling, frames are distributed to egress queues in a more sparse manner, and cut-through transmission is also feasible when the current queue receives and transmits frames at the same time, which cannot be done with CQF. These features of Paternoster guarantee more accurate per-flow state to the shaper and enable fast forwarding without having synchronous timing signaling.

4 Modeling

In this section, models of ATS approaches in a simulation environment are proposed. We used Riverbed modeler for designing models and running simulations, it is a discrete-event simulation tool providing performance evaluation for internet technology applications.

4.1 Modeling UBS

Based on the root-child process hierarchy, the ingress modules and functionality in UBS are done in the root process, which parses the frames and forwards to the shared queue group, which are a series of child processes created by the root, representing the per-flow shaper associated with each shaped queue, the benefits of applying this hierarchy for UBS is that each bridge and end station, from the root process point of view, has the direct access to every shaper, which means it is able to monitor and evaluate real-time state of all per-flow shaping, for instance, it could terminate a child process of one shaper when there are no more queued frames, and generates a new process for the shaper once new frames arrive at the same class of flow.

Figure 5 depicts the root process in UBS. In process model, green circles represent forced states and red circles for unforced states: unforced states allow a pause between enter and exit executives of the states during execution, and the process remains suspended only until invocation causes it to progress into the exit executives of its current state; On the contrary, forced state does not allow the process to wait during the execution, thus forced state starts, finishes both enter and exit executives immediately.

Figure 4: Process Hierarchy

Figure 5: Process model of UBS

The root process of UBS includes two kinds of in-
Packet arrival: external interrupt when new frames arrive at the ingress port, frames are then parsed and recognized by an assigned identification tag (e.g. VLAN, source and destination address) in header field and will be forwarded to the corresponding child process.

Timer out: internal interrupt scheduled by the root process itself, two timers are used here: a shared timer is set to simulate the elapsed time for the whole transmission procedure of each frame, from the first bit to the last bit leave the egress port; another cycle timer is set with a higher frequency than the shared timer, it is used for a periodic check on shared queues, since in the modeler, a process could not detect the frames being passed from child process to root process, namely, the timer is used to check periodically if there are any frames waiting in the shared queue to be transmitted to egress ports.

Two types of child processes, implementing LRQ and TBE algorithms respectively, are presented in this paper. In the simulation, root and child processes both have the same access to the shared memory, frames received by the root process will be allocated to the specific memory block and invoke the corresponding child process, afterwards the child process extracts the frames from the given memory address, calculates the eligibility time and forwards them to the corresponding shaped queue.

For LRQ algorithm, the child processes first need to get the size of head-of-queue frame, and sets up a timer lasts for a duration equals to the quotient between frame size and reserved link rate. A draft of state machine for LRQ child process is shown in Figure 6.

In shaping state, the token level is kept by a variable $b_i$, the level increases with a constant rate of each shaper, the system kernel updates the state of variable $t_i$ with current time, when an eligible frame is transmitted from the shaper, two possible comparison results between token level and frame size are considered in shaping state:

$b_i \geq \text{size}$: forward head-of-queue frame to shared queue directly

$b_i < \text{size}$: start a timer with duration of $(\text{size} - b_i)/r_i$ once the timer expires the frame is forwarded

The process model of ATS use the same architecture as the TBE model, as introduced in this section, the model contains two major parts: root process and child process, the former covers the operations such as forwarding frames to shaped queue and extracting frames from shared queue, the latter implements the ATS algorithm inside each shaper. Shared memory block between root and child processes enables the internal frame transfer inside one node.

4.2 Modeling Paternoster

In the model of Paternoster, the epoch updating is done by setting a cyclic timer each with duration $\tau$. The queue indexes of each service class are represented by four fixed numbers, so that prior, current, next and last queues are allocated with corresponding numbers at different epochs, the number of queue groups is a configurable option, in this work, two groups of queues numbered from 0 to 3 and from 4 to 7 stand for two service classes, shows the state transitions in the Paternoster model.
Figure 8: Process model of Paternoster

Timer 0 and 1 are set for class 0 and class 1 epochs updating, every time before transmit frames from current queue the shaper has to make sure prior queue is drained, otherwise it is supposed to dequeue frames from prior, the remaining frames need to be dropped if the amount exceeds a specific threshold, if not, the frames in prior are allowed to be transmitted. Threshold variables are defined in the model and used in epoch updating states Update class1 and Update class0. Additionally, if no transmission happens during the epoch updating time and frames exist in the current queue, the shaper has to launch new transmission after the update is finished. Thus the functions of epoch update contains:

- Update index of prior, current, next and last queues
- Update reserved bandwidth
- Check the amount of remaining frames in prior, if exceeds the threshold then drop all left frames, if not, transmit all frames
- Check any undergoing transmission exists, if not, start transmitting from current queue, else, wait for the ongoing transmission to be finished

According to Stream Reservation Protocol (SRP), the bandwidth reservation is done in terms of allow a certain amount of frames counting in bit during a time period, the amount equals to epoch duration ($\tau$) multiply reserved data rate ($r_i$), it is represented by rsv_remaining variable in the model, an amount equals to the frame size is subtracted from the reserved bandwidth when a frame is transmitted. Inside the Enqueue state, the model enqueues frames to current, next and last queues successively, and drops the frames when all three queues are full with reserved bandwidth.

Dequeue state contains the procedure of getting frames from queues and transmission selection based on priority classes, class with higher priority, class0 in this project, is checked before other classes, an example of checking sequence is: prior queue of class0 → current queue of class0 → prior queue of class1 → current queue of class1 → ... when a frame is extracted from queue, the process starts a timer stands for the transmission time of the frame from egress port, since the modeler is driven by discrete events.

5 Simulations and Results

To evaluate the asynchronous shaping algorithms, in this section, the LRQ, TBE and Paternoster models proposed in the last section are used for carrying out simulations in Riverbed simulator, the ATS algorithm proposed in the standard draft is not given in this work. A simple topology where only one flow exists, as depicted in Figure 9, is evaluated to test the behavior.

Simulation parameters are given in Table II. In this paper, the main concern is to simulate different working environments for the ATS switch, thus all the values are taken based on simulation requirements instead of real-world use cases.

| Parameters                        | Value                  |
|-----------------------------------|------------------------|
| Frame size                        | 720 – 1228 bytes       |
| Reserved bandwidth                | 50 Mbps                |
| Link rate                         | 1 Gbps                 |
| Bandwidth of input flow           | 32 – 1208 Mbps         |
| Paternoster epoch length          | 0.01, 0.005, 0.0025 s  |

The simulation results are based on our previous proposed paper [19]. In Table III, the average frame loss rate comparison among Paternoster with different epoch length and UBS are given. Since all devices have limited storage space and each traffic class is assigned with dedicated bandwidth, excess frames over the bandwidth limit will be dropped. Independent to the algorithm, the input flows to UBS follow the leaky bucket constraint, thus LRQ and TBE algorithms have similar frame-loss performance as shown in the first row.

| Algorithms       | 32 | 48 | 64 | 80 | 96 | 112 | 128 | 144 | 160 | 176 | 192 |
|------------------|----|----|----|----|----|-----|-----|-----|-----|-----|-----|
| TBRTE            | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 | 1.1 | 1.3 | 1.5 | 1.7 | 1.9 | 2.1 |
| Paternoster A(r = 0.01s) | 0.0 | 0.3 | 0.5 | 0.7 | 0.9 | 1.1 | 1.3 | 1.5 | 1.7 | 1.9 | 2.1 |
| Paternoster B(r = 0.005s) | 0.0 | 0.3 | 0.5 | 0.7 | 0.9 | 1.1 | 1.3 | 1.5 | 1.7 | 1.9 | 2.1 |
| Paternoster C(r = 0.0025s) | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 | 1.1 | 1.3 | 1.5 | 1.7 | 1.9 | 2.1 |

| TABLE V: Average per-hop delay (milliseconds) comparison |
|---------------------------------------------------------|
| Algorithms       | 32 | 48 | 64 | 80 | 96 | 112 | 128 | 144 | 160 | 176 | 192 |
|------------------|----|----|----|----|----|-----|-----|-----|-----|-----|-----|
| LRQ              | 0.35 | 0.39 | 0.38 | 0.36 | 0.34 | 0.32 | 0.30 | 0.28 | 0.26 | 0.24 | 0.22 |
| TBE              | 0.40 | 0.44 | 0.48 | 0.52 | 0.56 | 0.60 | 0.64 | 0.68 | 0.72 | 0.76 | 0.80 |
| Paternoster A(r = 0.01s) | 0.003 | 0.005 | 0.007 | 0.009 | 0.011 | 0.013 | 0.015 | 0.017 | 0.019 | 0.021 | 0.023 |
| Paternoster B(r = 0.005s) | 0.00257 | 0.00517 | 0.00757 | 0.01097 | 0.01437 | 0.01777 | 0.02117 | 0.02457 | 0.02797 | 0.03137 | 0.03477 |
| Paternoster C(r = 0.0025s) | 0.063 | 0.124 | 0.184 | 0.244 | 0.304 | 0.364 | 0.424 | 0.484 | 0.544 | 0.604 | 0.664 |
As traffic intensity keeps increasing, the loss rate also constantly increase for all scheduling algorithms. 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 length ($\tau$): shorter length means less storage space per epoch thus causing higher loss rate, the reserved bandwidth of each flow is calculated as: $3 \cdot \tau \cdot \text{datarate}$. 

Table IV shows the storage usage of the switch during simulation. Regarding the switch, time for transmission of one frame depends on frame size, data rate also arrival and departure time of adjacent frames in LRQ algorithm. From the results: when the bandwidth of input flow is less than reserved level (input intensity $= 32, 48 \text{Mbps}$), 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 algorithm enforces a pending 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 $\geq 50 \text{Mbps}$), 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.

Finally, 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 and UBS/TBE have 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 overloads. 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.

6 Conclusion

Currently, the standard 802.1Qcr for Asynchronous Traffic Shaping (ATS) is still not finalized. Asynchronous shaping aims at providing low congestion loss and deterministic performance while not using time synchronization in TSN, the objective of this paper is to test the performance of ATS in a series of simulations. With models built in Riverbed modeler, it is able to simulate the work of ATS and collect results for analyses.

The evolution of ATS starts with priority-based traditional Ethernet, where traffic flows are sorted by assigned priority, upon which frames are selected for transmission. Then approach like CBS is introduced to shape egress flows to prevent congestion caused by bursty flows. Also new traffic class like AVB is created, providing services with limited level of deterministic to specific flows.

In TT Ethernet and TSN, the notion of global time synchronization is integrated in all nodes and end stations, scheduling approach is then able to carry out operations on time-triggered schedule and offer determinism for time-sensitive flows. Schedulers like TAS and CQF are both set up on the time base, the performance requires high precision on time synchronization, which is possible to be affected by failures.

ATS presents scheduling without global notion of timing, while still provide service with high-level of determinism. From results collected in the simulations, UBS with LRQ algorithm performs weakly in lightly loaded networks, in terms of delay and bandwidth utilization. While TBE algorithm allows a certain level of bursty transmission during idle period also limits the egress flow with leaky bucket constraints when network is rather full or overfull.

Deriving from CQF, Paternoster scheduling increases the number of cyclic queues for each traffic class, providing bounded transmission delay for all flows. Knowing from simulation results, the trade-off between frame loss probability and delay in Paternoster depends on the length of epoch, short duration could be set for flows taking less bandwidth than reserved value, also assigning shorter epoch to higher priority flows results in lower transmission delay.

7 Acknowledgement

This work has been partially supported by the “Fronthaul (FH) and time sensitive network (TSN) technologies for Cloud Radio Access Network (C-RAN)” project founded by Eurostars Funding.

References

[1] Vitturi, Stefano, Paulo Pedreiras, Julian Proenza, and Thilo Sauter. "Guest editorial special section on communication in automation.” IEEE Transactions on Industrial Informatics 12, no. 5 (2016): 1817-1821.
[2] Wu, Jun, Zhifeng Zhang, Yu Hong, and Yonggang Wen. "Cloud radio access network (C-RAN): a primer.” IEEE Network 29, no. 1 (2015): 35-41.
[3] Checko, Aleksandra, Henrik L. Christiansen, Ying Yan, Lara Scolari, Georgios Kardaras, Michael S. Berger, and Lars Dittmann. "Cloud RAN for mobile networks — A technology overview.” IEEE Communications surveys & tutorials 17, no. 1 (2015): 405-426.
[4] Chih-Lin, I., Corbett Rowell, Shuangfeng Han, Zhikun Xu, Gang Li, and Zhengang Pan. "Toward green and soft: a 5G perspective.” IEEE Communications Magazine 52, no. 2 (2014): 66-73.
[5] IEEE Std. ’802.1Qcr IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks Amendment: Asynchronous Traffic Shaping’, 2018. [Online]. Available: https://1.ieee802.org/tsn/802-1qcr/. [Accessed: 06- Nov-2018].

[6] Specht, Johannes, and Soheil Samii. “Urgency-based scheduler for time-sensitive switched ethernet networks.” In Real-Time Systems (ECRTS), 2016 28th Euromicro Conference on, pp. 75-85. IEEE, 2016.

[7] Zhang, Hui, and Domenico Ferrari. "Rate-controlled service disciplines." Journal of high speed networks 3, no. 4 (1994): 389-412.

[8] Lim, Hyung-Taek, Lars Volker, and Daniel Herrscher. "Challenges in a future IP/Ethernet-based in-car network for real-time applications." In Proceedings of the 48th Design Automation Conference, pp. 7-12. ACM, 2011.

[9] Wan, Tao, and Peter 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, pp. 1-6. IEEE, 2015.

[10] Alderisi, Giuliana, Gaetano Patti, and Lucia Lo Bello. "Introducing support for scheduled traffic over IEEE audio video bridging networks." In Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on, pp. 1-9. IEEE, 2013.

[11] Farzaneh, Morteza Hashemi, and Alois Knoll. "A graphical modeling tool supporting automated schedule synthesis for time-sensitive networking." In Emerging Technologies and Factory Automation (ETFA), 2017 22nd IEEE International Conference on, pp. 1-8. IEEE, 2017.

[12] Lim, Hyung-Taek, Daniel Herrscher, Martin Johannes Wallt, and Firas Chaari. "Performance analysis of the IEEE 802.1 ethernet audio/video bridging standard." In Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques, pp. 27-36. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012.

[13] Farzaneh, Morteza Hashemi, and Alois Knoll. "Time-sensitive networking (TSN): An experimental setup." In Vehicular Networking Conference (VNC), 2017 IEEE, pp. 23-26. IEEE, 2017.

[14] Santos, Rui, Ricardo Marau, Alexandre Vieira, Arnaldo Oliveira, and Luis Almeida. "A synthesizable ethernet switch with enhanced real-time features." In Industrial Electronics, 2009. IECON'09. 35th Annual Conference of IEEE, pp. 2817-2824. IEEE, 2009.

[15] Groß, Friedrich, Till Steinbach, Franz C. Schmidt, and Bernd 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, pp. 9-13. IEEE, 2014.

[16] Mohammadpour, Ehsan, Eleni Stai, Maaz Mohiuddin, and Jean-Yves Le Boudec. "Latency and Backlog Bounds in Time-Sensitive Networking with Credit Based Shapers and Asynchronous Traffic Shaping." In 2018 30th International Teletraffic Congress (ITC 30), vol. 2, pp. 1-6. IEEE, 2018.

[17] Specht, Johannes, and Soheil Samii. "Synthesis of Queue and Priority Assignment for Asynchronous Traffic Shaping in Switched Ethernet." In Real-Time Systems Symposium (RTSS), 2017 IEEE, pp. 178-187. IEEE, 2017.

[18] IEEE Std. ’802.1Qch IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks Amendment: Cyclic queueing and forwarding’, 2016. [Online]. Available: https://1.ieee802.org/tsn/802-1qch/. [Accessed: 06- Nov-2018].

[19] Zhou, Zifan, Ying Yan, Michael Berger, and Sarah Ruepp. "Analysis and modeling of asynchronous traffic shaping in time-sensitive networks." In 2018 14th IEEE International Workshop on Factory Communication Systems (WFCS), pp. 1-4. IEEE, 2018.