IEEE 802.1Qcr Asynchronous Traffic Shaping with Linux Traffic Control

Christopher Pfefferle, Florian Wiedner*, Christoph Schwarzenberg*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: ga38pav@mytum.de, wiedner@net.in.tum.de, schwarzenberg@net.in.tum.de

***Abstract*—The widespread TSN standards, as introduced by the IEEE 802.1 working group, provide low latency scheduling and shaping with guaranteed packet transfers. Until recently, they depend on a synchronized clock between all network nodes. The newly introduced ATS standard however revokes this dependency for software with real-time requirements and can introduce TSN to a wider community. This paper describes the requirements to implement the ATS standard using the Linux TC tool for traffic shaping and scheduling. While it is currently impossible to implement LRQ queues or the Paternoster scheduler with TC commands, a model of the UBS algorithm with TBE queues will be presented.**

***Index Terms*—asynchronous traffic shaping, time sensitive network, traffic control, traffic scheduling**

1. Introduction

Network standards take ever-growing steps to revolutionize our daily lives. Presentations are held online with participants scattered all around the globe and even presentation systems in offices send slides from the presenter over the local area network to the monitor. In those scenarios a slight delay will not irritate the speaker, but when dealing with precision machinery a delay of only milliseconds can break fragile components. For these use cases the IEEE 802.1 working group creates standards for Time-Sensitive Networking (TSN) to provide communication protocols and features that can deliver precise communication.

A major contribution was the introduction of synchronization between participating devices in a network, introduced by the IEEE 802.1BA Audio Video Bridging (AVB) standard, allowing precise traffic scheduling and guaranteed packet deliveries [1]. However, the time synchronization mechanism adds high complexity to network setup and maintenance. The IEEE 802.1Qcr Asynchronous Traffic Shaping (ATS) standard intends to bypass the complexity of synchronization by revoking it and allowing every network node to send traffic on its own timing [2]. Nevertheless, it still aims to achieve deterministic and low transmission delays.

In order to shape and schedule traffic on Linux machines a program is required to intervene in the path of packets from creation to the network interface. One such tool is traffic control (TC), it comes pre-installed on Linux distributions and provides powerful possibilities to control the network traffic of a computer [3]. With ATS relaxing the requirements for TSN and TC moreover providing the necessary tools, daily used software is able to establish real-time requirements and further facilitate in the ever-growing impact of the internet.

This paper gives an overview over ATS and Linux TC, presents a possible model to implement a part of the standard with the tool and points out its current limitations. It is structured as follows: Section 2 will give a short history of related work while Section 3 describes implementation details of ATS and a selective overview over the possibilities of TC. In Section 4 the details of how TC can be utilized to implement the requirements of ATS will be shown, and Section 5 will conclude this work and suggest next steps.

2. Related work

The first part of this Section will provide a brief overview over the development of TSN and the background of the ATS standard and its latest developments. The second part will introduce the Linux TC command and its contributions in networking on Linux machines, as well as its relevance.

2.1. Asynchronous Traffic Shaping

Asynchrony in network traffic has been established as the standard in shared networks for a long time and is the backbone of the Internet connecting millions of nodes. Rigolio et al. proposed shaping on the Asynchronous Transfer Mode (ATM) as early as 1991, offering control over the bandwidth and flow exiting a given system [4].

In 2011 the IEEE 802.1BA standard for AVB was approved, proposing the features for TSN that are still relevant today [1]. It introduced time synchronization between network devices and traffic shaping, to minimize delay and jitter for distributed systems with real-time constraints.

Ahead of the introduction as an IEEE standard, in 2016 Specht and Samii introduced the Urgency-Based Scheduler (UBS) in [5] with two proposed algorithms: Length-Rate Quotient (LRQ) and Token Bucket Emulation (TBE). They are both based on Rate-Controlled Service Disciplines (RCSDs) and therefore can provide guarantees on both deterministic and statistical performance by separating the scheduling and shaping components [6].

One year later UBS and the then newly introduced Paternoster scheduler, which is based on Cyclic Queuing and Forwarding (CQF) [7], were approved as an official standard by the IEEE TSN working group within the

doi: 10.2313/NET-2022-01-1_03

IEEE P802.1Qcr ATS amendment. Zhou et al. have compiled a detailed insight into ATS together with a performance evaluation in various simulated environments [8].

Because of its recent publication there are not many works incorporating it yet; performance evaluations are given by Specht and Samii in [9] and by Zhou et al. in [10], and Mohammadpour et al. provides computed worst-case bounds for latency and backlog in [11]. Le Boudec analyzes a more generalized approach of UBS in [12] and Grigorjew et al. propose an addition to increase jitter control in [13]. Also new introduced standards take ATS into account, the IEEE P802.1Qdd Resource Allocation Protocol standard incorporates support for ATS [14].

The most recent performance assessment of ATS was released by Fang et al. in November 2020, taking various released standards of the TSN working group into account, revealing performance advantages of ATS especially in heavy-load cases [15].

2.2. Linux traffic control

The Linux TC tool was introduced in 2001 with the Linux kernel version 2.2, within the iproute2 package. An influential work is done by Hubert, combining definitions and application examples in a well-arranged document and continuing updated support on his online HOWTO document [16].

TC is a powerful tool for distributing and shaping network traffic, allowing the user to define detailed rules. It is used to rule over multiple services communicating through a network with restricted capabilities, examples like the work done by Vila-Carbo et al. in [17] show that its qualities are capable to define rules for real-time transmissions.

The introduction of an implementation of the Credit-Based Shaper (CBS) for TC [18], as defined by the IEEE 802.1Q-2014 standard, further shows the potential of TC for TSN applications and is used to implement synchronous traffic shaping on computers using Linux-based operating systems.

3. Architecture Details

Here, a short selective overview over the architecture of the ATS standard and Linux TC will be given. Particularly the later is a powerful tool to work with and has an accordingly large documentation. The focus will therefore lay on a subset of the traffic shaping capabilities needed to compare the possibilities of TC with the requirements of ATS in Section 4.

3.1. ATS algorithms

This Section is mainly based on [8]. The idea behind ATS is the independent clock of every connected device in a network, discarding the problems that arise when distributed devices have to agree on a synchronized timer. Its main requirement are queues that support asynchrony. A shaper is bond to each which assigns eligibility times to the frames in the queue, and on this information a transmission selection algorithm decides when frames are transmitted. This algorithm can be described as a simple gate control, taking the eligibility times into consideration. These initial shaped queues are simple FIFO queues and they ensure the processing of high-priority flows is not affected by malicious or other interfering flows.

The shaped queues need to follow the queue allocation rules, direct quoted from [8]:

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.

After shaping the eligible frames, they are sent from the shaped queues and stored in shared queues. These are managed by one of the shapers described below which selects and forwards them to the network interface releasing them into the network.

The ATS standard proposes two scheduling algorithms which can be used to realize asynchronous shaping queues, the foremost introduced UBS algorithm and the Paternoster algorithm.

The UBS scheme allows two types of shaped queues to be used: LRQ and TBE, respectively based on the frame-by-frame leaky bucket algorithm and token-based leaky bucket algorithm [6].

LRQ disregards the incoming flow pattern and shapes with a stabilized transmitting/leaking rate, by calculating the eligibility time of a packet as "the quotient between the size of the previously transmitted frame and the reserved link rate of the particular class" [8].

TBE allows some level of bursty traffic transmitting while maintaining an average rate, it uses the accumulation time of "tokens" in a "bucket" to calculate the eligibility time of a packet. In comparison to LRQ, it provides a better utilization of the given bandwidth on a lighter load.

The scheduling is achieved using the ATS algorithm based on a Leaky Bucket approach. Frames are processed with respect to their eligibility times, their arrival time, the size of the last frame, and the current system clock allowing to drop overdue frames. Therefore, the ATS scheduler acts as the final shaper for the available bandwidths.

The Paternoster queuing and scheduling algorithm is a cyclic approach. It utilizes four queues, which in every epoch pass through one of the four states *prior*, *current*, *next* and *last* as depicted in Table 1. In each epoch, frames are enqueued into the *current* queue, if it is full they are passed through to the *next* or *last* queue or get dropped, and are only dequeued from the *current* queue. The epoch length will influence the delay and has to stay consistent within the network. [8]

doi: 10.2313/NET-2022-01-1_03

TABLE 1: Queuing in Paternoster, adapted from [8]

Queue / Epoch	Queue0	Queue1	Queue2	Queue3
Epoch 0	prior	**current**	next	last
Epoch 1	last	prior	**current**	next
Epoch 2	next	last	prior	**current**
Epoch 3	**current**	next	last	prior
...				

3.2. TC for traffic shaping

This Section is largely based on the information compiled by [16]. Every network packet, which can either be produced by a local program or is being forwarded, has to pass through the TC architecture.

The main components are Queuing Disciplines (qdiscs), i.e., specified queuing algorithms. They can be either classless or classful, the later supporting an internal division into classes that again contain configurable qdiscs. Hence, they are arranged in a tree structure, with the Linux kernel interacting (enqueuing and dequeuing of network packets) with the root node only. A qdisc can perform three actions on packets queued into it: (1) scheduling, i.e., prioritization of packets over others, (2) shaping, i.e., delaying or even dropping packets to satisfy maximum traffic rate requirements, and (3) policing (if used on incoming traffic), i.e., dropping packets to satisfy internal requirements on incoming traffic.

If a qdisc may delay packets for the purpose of maintaining a constant transmission rate it is considered to be non-work-conserving. If it, on the other hand, sends out packets as soon as they are available, it is considered work-conserving.

Finally, a filter can be assigned to each class, it holds conditions with which packets can be classified. If a filter matches a packet, it will be forwarded to the qdisc of the respective class. In the following the qdiscs addressed in this paper are explained:

pfifo_fast
: The default qdisc, a classless shaper with three FIFO queues, one for each priority level as defined by the Type of Service (TOS) flag of network packets. Packets are dequeued starting from the highest priority queue.

Token Bucket Filter (TBF)
: A classless shaper that supports a set maximum rate with short bursts.

Stochastic Fairness Queuing (SFQ)
: A classless scheduler that dequeues packets in a round-robin fashion through flows, which mostly correspond to a TCP/IP connection.

PRIO
: A classful scheduler similar to pfifo_fast, but it supports enqueuing with filters and subclasses other than simple FIFO queues. Dequeuing is done the same way, starting at the defined first qdisc.

Hierarchical Token Bucket (HTB)
: A classful shaper that allows to precisely limit the bandwidth of its child qdiscs with a possibility to borrow unused resources.

Clark-Shenker-Zhang algorithm (CSZ)
: A complicated classful scheduler proposed by the three eponymous researchers in [19], providing guaranteed service for real-time applications along with best-effort queues, reducing delay and jitter by deliberately not shaping.

Credit-Based Shaper (CBS)
: A classless shaper that implements the CBS algorithm as introduced in the IEEE 802.1Qav standard, relying on set bandwidths.

Earliest TxTime First (ETF)
: A classless shaper that is constructed to support shaping in TSN, dequeuing packages on a configurable timer.

4. Implementation of ATS with TC

With the requirements and available tools introduced, a possible implementation of the ATS standard with the UBS scheme and TBE queues is presented. Section 4.2 will then explain the problems encountered when trying to implement LRQ queues or the Paternoster scheme. Both Subsections refer to details of the ATS algorithm as requirements and the functionalities of TC as capabilities to implement the former. For detailed descriptions and definitions refer to Section 3 and the corresponding Subsections.

4.1. A TC model of UBS/TBE

The root qdisc of the TC scheme does not necessarily need any shaping capabilities, shaping will be achieved by further qdiscs. But to contain further classes it needs to be classful. Its purpose is the separation of packets into queues with regards to their priorities, so packets with the same priority are handled by the same queue. This enforces the separation of the different priorities as required by QAR2. As the distribution of packets into classes can be controlled by filters, the focus lies on the correct dequeue strategy. The root qdisc must abide to the ATS scheduling algorithm. Potential candidates as root nodes include a PRIO qdisc that achieves a strict prioritization, the CSZ algorithm that provides guaranteed service for high-prioritization packets while possibly neglecting lower-prioritization ones, and HTB which is a generally good competitor that provides no strict prioritization and acts as a shaper, which may introduce delay and jitter. The ATS algorithm on one hand performs shaping, which would render the HTB as the best option, but on the other hand it also performs strict prioritization which would require a PRIO qdisc. As a solution, both will be used to account for the requirements of the ATS algorithm.

Each class assigned to a priority level has to implement the requirements of the TBE queue. A viable option is the TBF shaper applying nearly the same token/bucket technique, but as it is a classless qdisc its application at this point would violate QAR1.

The only qdisc realizing both QAR1 and QAR3 is the classless SFQ scheduler which permits to split the packages by conversations. As it does not support shaping, a separate shaper is needed.

doi: 10.2313/NET-2022-01-1_03

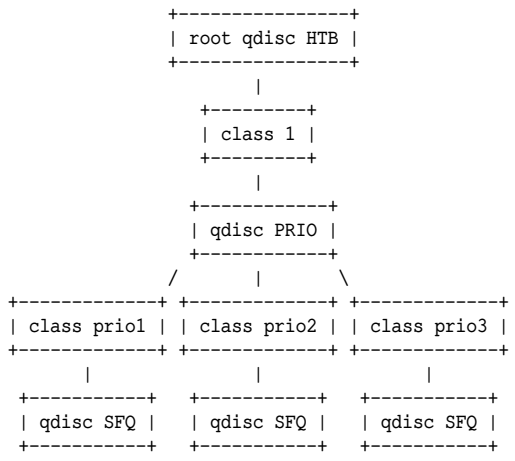

Figure 1: Proposed TC scheme for ATS

The solution that is presented in this paper is the usage of a HTB root node with only one subclass, a PRIO qdisc. With this setup, as shown by Figure 1, a SFQ qdisc can be inserted per priority level to ensure all three queue allocation rules are met.

4.2. Limitations

LRQ queues cannot be realized with the qdiscs provided by TC, because it heavily relies on the calculation of eligibility times and their propagation through the scheme. ETF is currently the only qdisc to allow the calculation of eligibility times. Because it is classless, it lacks the requirements to be used as a root node by itself, and as the calculated times cannot be used in further schedulers a new qdisc would be required to account for the scheduling done by the ATS algorithm considering the timer of ETF.

Similarly, Paternoster is not realizable with the available tools. It requires en- and dequeuing into and from different queues depending on the current epoch, which can neither be accomplished with the current qdiscs nor with filters.

5. Conclusion and future work

The IEEE 802.1Qcr ATS standard was introduced to increase the real time capabilities of networks that have no synchronized timer available for every node, reducing the requirements while still providing guarantees for time sensitive applications. Using the TC tool this paper shows a possible model to realize this standard, in particular the UBS scheme with TBE queues, on Linux machines. Using only the HTB, PRIO, and SFQ qdiscs it is possible to comply to the conditions of ATS.

It further highlights the problems that arise when working on the other possible schemes of the standard, namely UBS with LRQ queues and the Paternoster queuing/scheduling algorithm. With most of the requirements met, TC currently lacks key features like the usage of eligibility times or dynamically changing the roles of queues. However, it seems possible to add these features in future updates.

The next step would be an implementation of the proposed scheme to analyze its practicability and performance. Even though TSN standards are only applied in a specialized field, if this approach turns out feasible it may allow more general applications in Smart Homes or the Internet of Things.

References

[1] "IEEE 802.1BA-2011 - IEEE Standard for Local and Metropolitan Area Networks--Audio Video Bridging (AVB) Systems," https://standards.ieee.org/standard/802_1BA-2011.html, 2011, [Online; accessed 25-March-2021].

[2] "P802.1Qcr – Bridges and Bridged Networks Amendment: Asynchronous Traffic Shaping," https://1.ieee802.org/tsn/802-1qcr/, 2018, [Online; accessed 20-March-2021].

[3] B. Hubert, "iproute2 - TC (8)," Linux man page (8), 2001.

[4] G. Rigolio, L. Verri, and L. Fratta, "Source Control and Shaping in ATM Networks," in *IEEE Global Telecommunications Conference GLOBECOM '91: Countdown to the New Millennium. Conference Record*, vol. 1, 1991, pp. 276–280.

[5] J. Specht and S. Samii, "Urgency-Based Scheduler for Time-Sensitive Switched Ethernet Networks," in *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*, 2016, pp. 75–85.

[6] H. Zhang and D. Ferrari, "Rate-Controlled Service Disciplines," *Journal of High Speed Networks*, vol. 3, no. 4, pp. 389–412, 1994.

[7] "P802.1Qch – Cyclic Queuing and Forwarding," https://1.ieee802.org/tsn/802-1qch/, 2016, [Online; accessed 25-March-2021].

[8] Z. Zhou, M. S. Berger, S. R. Ruepp, and Y. Yan, "Insight into the IEEE 802.1 Qcr Asynchronous Traffic Shaping in Time Sensitive Network," *Advances in Science, Technology and Engineering Systems Journal*, vol. 4, no. 1, pp. 292–301, 2019.

[9] J. Specht and S. Samii, "Synthesis of Queue and Priority Assignment for Asynchronous Traffic Shaping in Switched Ethernet," in *2017 IEEE Real-Time Systems Symposium (RTSS)*, 2017, pp. 178–187.

[10] Z. Zhou, Y. Yan, M. Berger, and S. Ruepp, "Analysis and Modeling of Asynchronous Traffic Shaping in Time Sensitive Networks," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2018, pp. 1–4.

[11] E. Mohammadpour, E. Stai, M. Mohiuddin, and J.-Y. 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. 02, 2018, pp. 1–6.

[12] J.-Y. Le Boudec, "A Theory of Traffic Regulators for Deterministic Networks with Application to Interleaved Regulators," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2721–2733, 2018.

[13] A. Grigorjew, F. Metzger, T. Hossfeld, J. Specht, F.-J. Götz, F. Chen, and J. Schmitt, "Asynchronous Traffic Shaping with Jitter Control," 2020.

[14] "P802.1Qdd – Resource Allocation Protocol," https://1.ieee802.org/tsn/802-1qdd/, 2018, [Online; accessed 20-March-2021].

[15] B. Fang, Q. Li, Z. Gong, and H. Xiong, "Simulative Assessments of Credit-Based Shaping and Asynchronous Traffic Shaping in Time-Sensitive Networking," in *2020 12th International Conference on Advanced Infocomm Technology (ICAIT)*, 2020, pp. 111–118.

[16] B. Hubert, "Linux Advanced Routing & Traffic Control," in *Proceedings of the Ottawa Linux Symposium*, 2002, pp. 213–222.

[17] J. Vila-Carbo, J. Tur-Masanet, and E. Hernandez-Orallo, "An Evaluation of Switched Ethernet and Linux Traffic Control for Real-Time Transmission," in *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, 2008, pp. 400–407.

[18] V. C. Gomes, "tc-cbs (8)," Linux man page (8), 2017.

[19] D. D. Clark, S. Shenker, and L. Zhang, "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism," ser. SIGCOMM '92. Association for Computing Machinery, 1992, pp. 14–26.