Moving a step forward in the quest for Deterministic Networks (DetNet)

Vamsi Addanki
LTCI, Telecom Paris,
Institut Polytechnique de Paris
vaddanki@telecom-paris.fr

Luigi Iannone
LTCI, Telecom Paris,
Institut Polytechnique de Paris
luigi.iannone@telecom-paris.fr

***Abstract*---Recent years witnessed a fast-growing demand, in the context of industrial use-cases, for the so-called *Deterministic Networks (DetNet)*. IEEE 802.1 TSN architecture provides link-layer services and IETF DetNet provides network-layer services for deterministic and reliable forwarding. In such a context, in the first part of this paper, we tackle the problem of misbehaving flows and propose a novel queuing and scheduling mechanism, based on Push-In-First-Out (PIFO) queues. Differently from the original DetNet/TSN specifications, our solution is able to guarantee performance of priority flows in spite of misbehaving flows. In the second part of this paper, we present our simulator *DeNS: DetNet Simulator*, based on OMNET++ and NeSTiNG, providing building blocks for link-layer TSN and network-layer DetNet. Existing simulators have important limitations that do not allow simulating the full DetNet/TSN protocol stack. We overcome these limitations, making easy DetNet/TSN evaluations possible. Our simulations clearly show that our solution is able to satisfy constraints of deterministic networks, namely, guarantee zero packet loss and low latency, while at the same time allowing best-effort flows to co-exist. Furthermore, we show how our newly-proposed queuing and scheduling solution successfully limits the impact of misbehaving flows.**

***Index Terms*---Deterministic Network (DetNet), Time Sensitive Network (TSN), simulation, queuing.**

I. Introduction

The Internet works as a best-effort service, without guarantees, where end-to-end latency, jitter, and packet-loss probability have no fixed bounds. Latest years have witnessed a growing interest on the so called *Deterministic Networks (DetNet)* and *Time Sensitive Networks (TSN)*. In this type of networks, a bounded latency and zero packet loss is guaranteed to a set of priority flows. Such requirements are typically of high importance in industrial control process, audio/video streaming, vehicular control and so on. At network layer, mechanisms such as packet and flow replication, explicit bandwidth reservation, and priority queuing are employed to realize DetNet [1]. At the link-layer, the IEEE 802.1 Time-sensitive Networking (TSN) Task Group has specified standards for IEEE 802.3 Ethernet networks enabling bounded latency communication over standard Ethernet [2].

While TSN standards are able to achieve bounded latency, they lack mechanisms to ensure reliability in packet delivery. The most common approach to lower the packet loss probability to zero is to increase the size of buffers, as well as replicating the traffic on several independent links, while filtering the incoming traffic so as to make sure that no bandwidth allocation is ever violated ([1], [3]). IEEE 802.1Qbv Time Aware Shaper (TAS), IEEE Std 802.1Qbu and IEEE 802.3br frame preemption algorithms, IEEE 802.1Qav Credit Based Shaper (CBS), IEEE 802.1Qch Cyclic Queuing and Forwarding (CQF) were proposed as part of Time Sensitive networking (TSN) standard [4]. However, the effects of a misbehaving TSN/DetNet flow has not been taken into account in the IEEE standards for TSN, as well as the effects on worst case Quality of Service (QoS) for best-effort flows. *Our first contribution is to point-out potential problems with current standards and propose an enhanced queuing and scheduling mechanism with a traffic shaper designed for DetNet to co-exist with Internet traffic.* Our proposal is able to achieve bounded end-to-end latency, zero packet-loss for DetNet flows and, at the same time, provide a better worst-case QoS for best-effort flows.

Recent works ([5], [6]) implement simulation models for TSN standards, focusing on TAS, frame preemption, CBS, frame replication and elimination. Although these models satisfy TSN requirements, they fail to satisfy DetNet requirements. Moreover, the state-of-the-art TSN simulators are only limited to link-layer and miss the modular flexibility of INET [7] to extend the models to upper layers or to re-use the presented modules of TSN simulators in different scenarios. A recent study concludes an urgent need for DetNet simulators and rigorous simulations on the standards to better understand the working of such networks [8]. *Our second contribution in this paper is presenting DeNS: Deterministic Networks Simulator, based on OMNET++.* We incorporate TSN standard IEEE 802.1Qbv from [5] by re-implementing it so as to enable extensions to upper-layers. Beyond our proposed queuing and scheduling mechanism, we also implement Cyclic Queuing and Forwarding (IEEE 802.1Qch), resource allocation and sharing, classification based on both DSCP (Differentiated Service Code Point) and 5-tuple to facilitate per-class and per-flow configuration, and DSCP to VLAN Priority Code Point (PCP) mapping. We show by simulations that we successfully included the key aspects of DetNet, such as resource allocation, queuing and scheduling, scalability to large networks and co-existence of DetNet with best effort Internet traffic.

The remaining of the paper is organized as follows. In Sec. II, we describe the key aspects of DetNet. We discuss a

ISBN 978-3-903176-28-7

potential problem with TSN standards when applied for large scale DetNet and propose a novel Time Aware queuing and scheduling in Sec. III. In Sec. V we discuss the related work on DetNet and TSN in the context of simulation models, while describing our simulator *DeNS* in Sec. VI. In Sec. VIII we present the simulation results showing that *DeNS* satisfies the requirements of DetNet. Sec. IX concludes the paper.

II. DetNet Overview

Hereafter, we describe some of the key aspects of the DetNet architecture as specified in details by RFC 8655 [1], while in Sec. VI, we describe the components of ''DeNS'' which individually or collectively achieve such key aspects.

Resource Allocation: In order to guarantee packet delivery and to avoid losses due to congestion, explicit reservation of resources is made for DetNet flows along their path in the network. Strict reservation of resources is often used due to simplicity. However, strict reservation of resources leads to under-utilization of resources when such reserved resources are not being fully utilized. In Sec. VI-A2, we show how a combination of bandwidth and buffer reservation allows maximizing the utilization of bandwidth un-used by DetNet flows. While configuring bandwidth reservation is straightforward, reservation of buffer resources is more complicated. Often, buffer resources are allocated based on flow priority. In particular, the TSN standard uses the Priority Code Point (PCP) value in the VLAN header to prioritize the flows at link-layer for queuing and scheduling, with an egress queue completely dedicated to DetNet flows.

Queuing and Scheduling: DetNet achieves bounded latency and low packet-loss probability by reservation of bandwidth and buffer resources. However, this is not sufficient. The fundamental reason is that latency variation may result in the need for extra buffer space, increasing the worst-case per-hop latency. Standard queuing and scheduling algorithms allow to compute the worst case latency contribution of each node to the end-to-end latency, hence, the buffer space required by DetNet flows in each node. To achieve this, IEEE 802.1 working group specified a set of queuing, shaping, and scheduling algorithms, which include Credit Based Shaper (IEEE 802.1Qav), Timed-Aware Scheduling (IEEE 802.1Qbv), Frame-Preemption (IEEE 802.1Qbu). As previously explained, priority is determined by the PCP value, and mapped to a queue using the above-mentioned algorithms. However, there are several problems with Time-Aware scheduling in the context of DetNet. We discuss the issues related to misbehaving flows, and present our solution, in Sec. III.

Scalability to Large Networks: In order to reserve resources for individual DetNet flow a large amount of state information is required. Aggregation of DetNet flows is a technique to improve scalability in large networks. To this end, flows' classification based on MPLS hierarchy and IP DiffServ Code Points (DSCP) are some of the possible techniques described in RFC 8655. Typically network nodes in the path of DetNet flows are configured to classify flows at the ingress, based on certain fields of the packet header (e.g., 5-tuple classification) and mapped to a DSCP value in the IP header. At the egress, packets are classified based on the DSCP value, which determines the priority. Reserved resources are determined and allocated based on the classification based on the PCP value. However, is important to note that the PCP field in VLAN header is limited to 3 bits (8 values). In order to combine classification based on DSCP (for allocation of resources) and PCP (for priority queuing and scheduling at lower-layers), there is the need to map each DSCP value to a PCP value. This enables flows to be classified into a maximum of 64 classes since DSCP is only 6 bits.

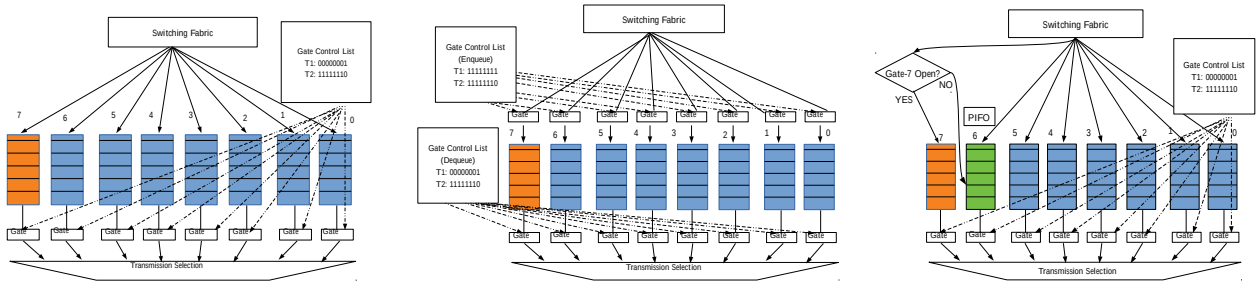
Co-existence of DetNet and Best Effort traffic: While DetNet flows are of high priority with reserved resources, their co-existence with best effort Internet traffic requires avoiding starvation of non-DetNet flows. At the same time, it is important that non-DetNet flows do not disrupt DetNet flows. To achieve this, RFC 8655 states the following rules: *a)* Un-used reserved bandwidth is made available to non-DetNet flows but not to other DetNet flows; *b)* DetNet flows may be rate-limited in order to ensure that any highest priority non-DetNet flows is also ensured a better packet-loss probability and worst case latency; *c)* sufficient opportunities for non-DetNet flows must be allowed by the underlying scheduling algorithm in order to avoid starvation of other flows in the network.

III. TSN and misbehaving Flows

Time sensitive networks are relatively small scale networks. However, DetNet aims at larger scale networks and to co-exist with the Internet, where there are high chances of a having misbehaving traffic, which can disrupt DetNet flows. In this section, we show how misbehaving flows can create problems in TSN and propose a novel mechanism, which deviates from the traditional FIFO queuing in the context of TSN/DetNet. From here on, for the sake of simplicity, we consider VLAN header field PCP, whose values range from 0 to 7 have increasing priority (7 being the highest priority) and are mapped to queues namely, queue-[0-7] in TSN queuing and scheduling architecture. Before describing in details the impact of misbehaving flows we overview the two main TSN scheduling algorithms, namely TAS and CFQ.

Time Aware Shaper (TAS): Fig. 1a shows IEEE 802.1Qbv Time Aware Scheduling mechanism, where there are 8 priority queues, each corresponding to 0-7 VLAN priorities (PCP) and each queue is associated with a gate which opens and closes based on a schedule determined from Gate Control List (GCL).[1] A transmission selection algorithm is used to select packets to be transmitted from the queues with open gate state. IEEE 802.1 Qbv ensures that the packets leave the queue only in a reserved time-slot (determined by GCLs) so as to achieve deterministic departures from queue.

[1]Gates control whether or not the content of a queue can be scheduled for transmission. If the gate of a queue is open, a packet inside this queue may be selected by the transmission selection algorithm. If the gate is closed, a packets in the corresponding queue cannot be transmitted even if the outgoing network link is idle. Each entry of Gate Control List consists of gate states for each queue and the duration. The GCL entries are repeated in cycle.



(a) Original IEEE Time Aware Shaper. (b) Modified TAS with Tail Queue gates. (c) Time Aware PIFO Scheduler.

Figure 1: Original and modified TSN Queuing and Scheduling Architectures.

Cyclic Queuing and Forwarding (CQF): Similar to TAS, CQF (IEEE 802.1Qch) makes use of 8 queues prioritized based on PCP and gates at the head of the queues to control transmission time. However, two queues per-priority are used. During an odd cycle, one queue performs only enqueue operations (i.e., add packets to the queue), while the other only dequeues packets (i.e., taking them out of the queue). During the even cycle, the queue operations are switched. The advantage of such approach is that, for H hops and T cycle time, a maximum end-to-end latency of $(H + 1)\ T$ and a minimum of $(H\ \ 1)\ T$ is guaranteed.

When Flows Misbehave: We argue that there is a potential problem when a DetNet flow misbehaves. TAS takes no action on packets which arrive at the queue out of their reserved time-slot. In this case, packets are buffered and have to wait until the queue gate opens. This creates added queuing latency for subsequent packets which arrive in the reserved time-slot, thus disrupting the deterministic latency requirement. CQF solves this problem by synchronized enqueue and dequeue operation. However, this creates a new problem of congestion when a scheduled flow misbehaves using high transmission rate. CQF cannot solve this problem because packets are enqueued in one cycle and only dequeued in the next cycle, hence, not using the available bandwidth to reduce the congestion by transmitting at link rate. Furthermore, with increased queue lengths due to cyclic dequeue operations, best-effort non-priority flows observe high RTT, as a result of longer waiting time in the queue, resulting in starvation, an effect we show by simulation in Sec. VIII. We conclude that it is important to first separate periodic DetNet traffic and sporadic best effort traffic. Then, packets need to be dequeued and scheduled for transmission without waiting for a whole cycle.

IV. Protection from Misbehaving Flows

A. Tail Queue Gates

As a naive solution, we can enhance TAS with queue tail gates shown in Fig. 1b, introducing strict control on enqueue operations. Priority queue 7, which is reserved for periodic traffic, has head and tail gate states which are time synchronized so that scheduled traffic is only enqueued during the reserved time-slot. Packets arriving out-of reserved time slot will be simply discarded. This solution is only applicable if the time-critical traffic consists of only periodic traffic, but does not take into account random (sporadic) traffic, where packets may be sent in any time interval. Nonetheless, such a naive solution allows observing the effect of traffic-conditioning and bandwidth reservations on TAS in the presence of misbehaving flows (which interfere with periodic flows), without the effects of packet arrivals during closed gate time-interval.

B. Time Aware PIFO Scheduling (TAPS)

With the understanding from previous solution, we propose a novel architecture which enhances TAS with Push-In-First-Out queues [9], which allows to "push-in" packets in any position in the queue (not just at the head or tail), and able to guarantee services for all time-critical traffic. Fig. 1c shows TAPS architecture, where all the queues are FIFO except queue 6 which is Push-In-First-Out (PIFO). Packets of periodic traffic are enqueued into queue 7 during reserved time slot (odd cycle) similar to TAS. However, TAPS enqueues DetNet scheduled packets which arrive out-of reserved time-slot, into queue 6 shared by other types of traffic. Queue 6 PIFO needs to maintain only an index position, namely I_p, which divides queue 6 (Q_6) virtually into two parts Q_{st} (for scheduled traffic st) and Q_{be} (for best effort traffic be), where Q_{st} corresponds to the first part of the queue from the head to I_p, while Q_{be} the second part from I_p to its tail. Finally, periodic traffic (PCP=7) is enqueued at the tail of Q_{st} (pushed in at I_p) and best-effort traffic is enqueued at the tail Q_{be}. TAPS is similar to TAS during reserved time-slot. However, during the unreserved time-slot packets are enqueued into a separate queue which is PIFO and shared by best-effort traffic. Note that, this is not an over-complication compared to simply using a separate dedicated FIFO queue, as such a mechanism will cause similar problems discussed in Sec. III and causes potential starvation for best-effort traffic. TAPS allows best-effort traffic to opportunistically transmit packets during unreserved time-slot at the same time allows for best-effort packets to be dropped (instead of over-buffering which increases RTT) when Q_{st} length increases and finally the division of queue into Q_{st} and Q_{be} prioritizes scheduled traffic. A clear limitation of TAPS, because packet preemption is not allowed, is that the

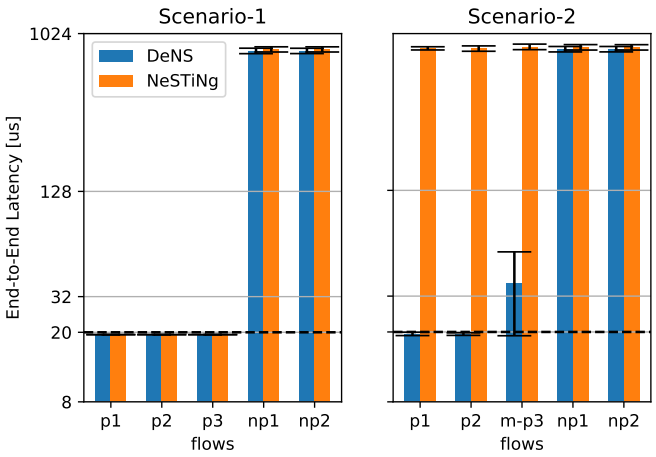


Figure 2: End-to-End latency for three priority flows and two non-priority (best effort) flows. Left hand side shows how latency is bounded for priority flows when all of them respect their bandwidth. Right hand side shows how in the case of TSN (NeSTiNg implementation), when one priority flow uses more bandwidth than expected (p3 in our case), the latency of the other priority flows explode, as high as the best effort traffic.

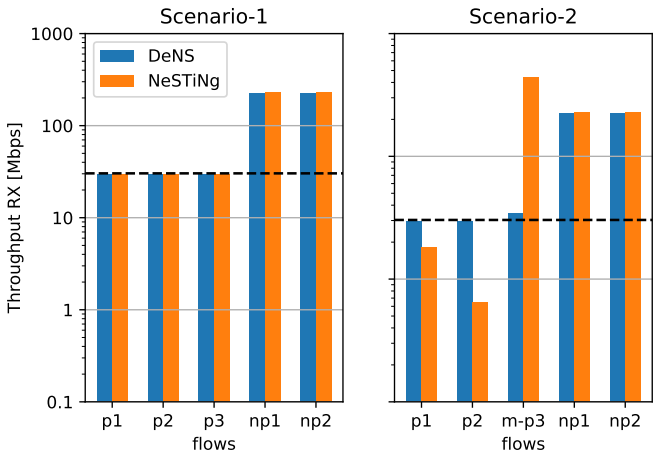


Figure 3: Throughput for 3 priority flows and 2 non-priority (best effort) flows. Left hand side shows how throughput is bounded for priority flows when all of them respect their bandwidth. Right hand side shows how in the case of TSN (NeSTiNg implementation), when one priority flow uses more bandwidth than expected (p3 in our case), the throughput of the other priority flows is ”eaten” by the misbehaving flow.

first packet enqueued into Q_{st} (when I_p is at the head of Q_6) experiences delay due to a best-effort frame in transmission.

V. Related Work

Before going further and introduce the simulator we developed to evaluate the solutions presented in the previous section, we discuss here the related research in the area of simulators for TSN and DetNet.

The current specifications of DetNet evolved as part of TSN, which in-turn evolved from AVB (Audio-Video bridging) and real-time Ethernet extensions. To our best knowledge, no specific simulator has yet been published that fully models Deterministic Networks. With IETF (Internet Engineering Task Force) working on DetNet specifications at network-layer and IEEE working on TSN standards at link-layer, a simulator bridging the gap between the two, enabling extensive evaluations of developed models is of a urgent need for the research community [8].

Several simulators covering TSN standards were published in the past. Jiang et al. [10] presented a TSN simulation model based on CoRE4INET framework [11], which offers functionality of IEEE802.1Qbv and IEEE802.1AS for traffic scheduling and time synchronization respectively. The authors just show by simulations that scheduled traffic is unaffected by best effort flows. The implementation by Jiang et al. satisfies one of the key features of DetNet, i.e., co-existence of DetNet with Internet's best effort traffic. However, the source code is not available online, making it inaccessible to the research community for further development and evaluations. In [12], Heise et al. propose TSimNet, based on OMNET++. TSimNet focuses on non-time based aspects of TSN, targeting industrial and avionic networks use-cases. Frame preemption, frame replication, per-stream filtering techniques were included in TSimNet. Although, the authors show that these techniques significantly reduce latency, the simulator does not include mechanisms to provide any guaranteed services for priority flows. Hence, TSimNet is not suitable for simulating DetNet.

In an inspiring recent work, J. Falk et al. [5] propose NeSTiNg, a simulator based on OMNET++ and INET framework. NeSTiNg includes TSN standards such as IEEE802.1Qbv (Time Aware Shaper), IEEE802.1Qav (Credit Based Shaper), IEEE802.1Qbu (Frame Preemption). It also provides priority queuing based on the PCP (Priority Code Point) field, in accordance to IEEE802.1 standards. However, NeSTing is not suitable for simulating DetNet. Indeed, the protocol stack implemented in NeSTiNg is limited to the link-layer, and although the simulator is based on INET framework, we observed that several NeSTiNg modules could not be used directly with INET standard modules. More specifically, no extensions to upper layers to make use of a traffic conditioner where proposed. As an example of such limitation, we show a small-scale simulation results in Fig. 2 and Fig. 3 showing end-to-end latency and throughput. In both figures we show two scenarios. In *Scenario-1* all DetNet flows respect the demanded bandwidth, while in *Scenario-2* one of the DetNet flows start using more bandwidth than they should. We compare TAS using NeSTiNg simulator and TAS with traffic conditioner and with upper layer extensions using our simulator DeNS. We observe from those figures that TAS with our extensions does guarantee services for scheduled traffic which arrives in reserved time-slot even in the presence of misbehaving flows as a result of traffic conditioning and bandwidth reservation.

Such simulations were not possible with state-of-the-art TSN simulators, especially for DetNet. In the following section, we briefly describe the key components of our DeNS simulator.

VI. DENS: DETNET SIMULATOR

In this section, we briefly describe the modules of *DeNS*, our simulator based on OMNET++ discrete event simulator [13] and INET the framework [7]. Due to space constraints, in the following subsection we only describe some of the important features which contribute to the realization of the key features of DeNS. DeNS simulator provides the building blocks for DetNet services at network-layer and TSN at link-layer, enabling easy experimentation to understand the overall effect of combined services. Although not discussed in this paper, we would like to emphasize the great amount of implementation work of *DeNS* in order to facilitate the re-use of modules in any other scenario and their compatibility with existing INET modules. We developed several modules, including VLAN interface with PCP parameters, VLAN encapsulation module, able to rewrite PCP in VLAN header when requested, all compatible with INET typenames for the proper functioning. The source code and a full documentation of *DeNS* is available at *https://github.com/vamsiDT/DeNS/*. We organized our code in to main modules which, we simply called Traffic Shaper and Queue Scheduler.

A. Traffic Shaper

Strict reservation is the most simple approach for bandwidth reservation and guarantees reserved resources to DetNet flows. However, this leads to low bandwidth available for non-DetNet (best effort) flows and under-utilization of bandwidth when DetNet flows are inactive. The Traffic Shaper of DeNS guarantees reserved bandwidth for DetNet flows, at the same time maximizes reallocation of the unused bandwidth to best effort traffic. It consists of a DSCP classifier, Per-Flow-Per-Class shaper (PFPC), and a PCP Marker module. This module operates at the network layer of the protocol stack, and packet go first through the DSCP Classifier, then the PFCP module, and finally through the PCP Marker module, as described in the following.

1) DSCP classifier: The DSCP classifier has one input gate and $N + 1$ output gates, where N is the number of of possible classes. This component takes an array of N DSCP values as parameters and classifies flows by matching the DSCP field in the IP header. The unclassified flows (or non-DetNet) are sent out through the gate $N + 1$. Each output of the DSCP classifier is connected to a corresponding input of the PFPC module. The use of DSCP classifier allows to scale to large networks, as described in Sec. II. It also solves the problem of limited number of priorities in PCP. We use DSCP classification, also to facilitate per-class resource allocation.

2) PFPC - Per-Flow-Per-Class shaper: PFPC has one input and one output gate for each of the $N + 1$ classes. As shown in Fig. 4, for each class i there are n_i bit-buckets and n_i token buckets, where n_i is the number of flows (identified with the classic 5-tuple) of the class with bit-bucket length greater than zero. Each bit-bucket of size B_{max} has an instantaneous occupancy of B_j^k,where k is class-id and j is 5-tuple flow-id in the class. Similarly, the token buckets having maximum T_{max} tokens have an instantaneous occupancy of T_j^k, where k is class-id and j is 5-tuple flow-id in the class. On packet arrival at the i^{th} gate of the PFPC component, only the i^{th} class actually receives the packet of length L, while all the other classes assume a hypothetical packet with zero packet length ($L = 0$). Then, every class k (with k ranging from 1 to $N + 1$) and for all flow-ids j in each class, the following steps are performed:

1) If $(L < (B - B_j^k)$ then accept the packet and continue; otherwise drop the packet.
2) Increment the bit-bucket by the packet length L.
3) Increment token buckets by t_j^k:
$$t_j^k = r_k * (Now - LastArrival_j) \quad (1)$$
where r_k is the reserved bandwidth for each flow of class k.
4) If $T_j^k \leqq B_j^k$ then B_j^k is decremented by T_j^k and T_j^k is set to zero; else T_j^k is decremented by B_j^k, B_j^k is set to zero.
5) A residual token bucket $T_{residual}$ is incremented by a value $t_{remaining}$ calculated as:
$$t_{remaining} = t_{unused} + t_{unreserved} \quad (2)$$
where
$$t_{unused} = \sum_{k=1}^{N} \sum_{j=1}^{n_k} T_j^k \quad (3)$$
$$t_{unreserved} = r_{unreserved} * (Now - LastArrival_{N+1}) \quad (4)$$
6) $T_{residual}$ can shared among all class $N + 1$ flows (best effort traffic) in T_j^{N+1} by any resource sharing algorithm.

After the above calculation, the packet is sent out from the i gate where the PFPC had received it. The PFPC component takes the bandwidth reservations as configuration parameters. The bandwidth reservation has to be specified for each of the N classes and this is reserved and guaranteed by an enforcing mechanism, for each flow belonging to a class. All the unclassified flows are sent to the $N + 1^{th}$ class. As the unused tokens from all the classes are collected, essentially, the $N + 1^{th}$ class is able to use all the unused and unreserved bandwidth. Although we collect unused tokens at every step to maximize the link utilization, we are still able to allow for all the classes, a maximum burst of B_{max}, the size of the bit-bucket. This way, we guarantee the reserved bandwidth and also be able to accept short duration of bursts which is typical for short-flows. The PFPC component realizes the following keys features of DetNet (as described in Sec. II) *a)* Resource Allocation; *b)* Co-existence of DetNet with best effort traffic. Essentially, this component helps in satisfying the requirement of bandwidth reservation while allowing non-DetNet flow and denying other DetNet flows to utilize the unused bandwidth.

3) PCP Marker: As described in Sec. II, there is a need for mapping DSCP to PCP values, in order to facilitate classification based on PCP at lower layers, for priority queuing and scheduling. For this reason, we develop a PCP Marker module, which has $N + 1$ input gates and one output gate. Based on the DSCP classification, a packet arrives at one of

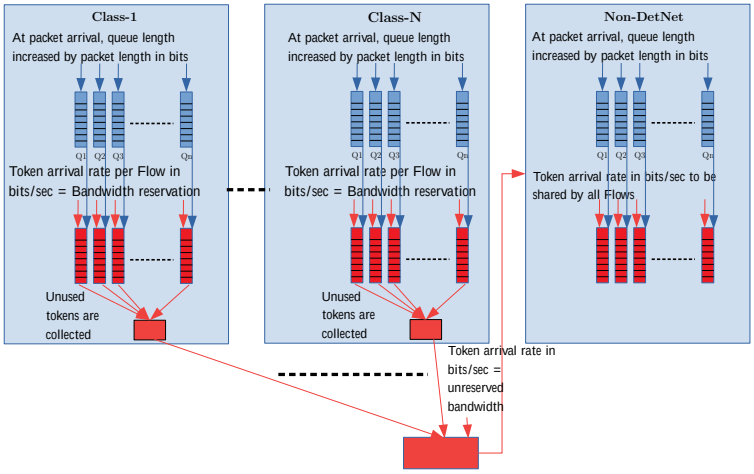


Figure 4: PFPC - Per-Flow-Per-Class Traffic Shaper.

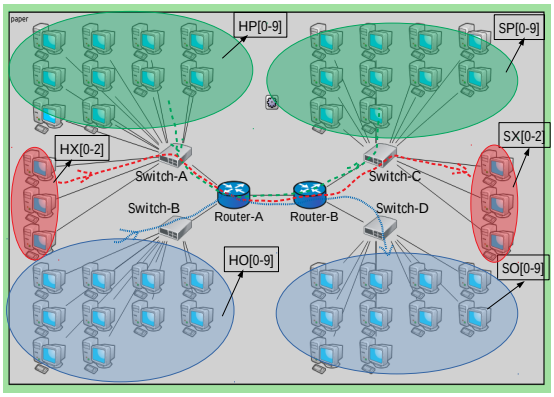


Figure 5: Simulation Topology.

the input gates of the PCP Marker. Each gate of PCP Marker is configured with a PCP value. When a packet arrives at i^{th} gate, a VLAN tag with PCP value equal to p_i (set by configuration), is attached to the packet.

B. Queue Scheduler

The Queue Scheduler module complements the Traffic Shaper module by further implementing key DetNet features such as queuing and scheduling, resource allocation, and co-existence of DetNet with Internet traffic. The Traffic Shaper module operates at network layer, while the Queue Scheduler module guarantees bandwidth reservations and other features of DetNet at layer 2. This module includes transmission algorithm, queue head gates, and transmission selection, all three of which taken from NeSTiNg [5], to extend the functionality of Time Aware Shaper. We also provide PIFO queue [9] modules in order to configure TAPS, described in Sec. III. Furthermore, it adds two new components, namely the Layer 2 PFPC Dropper and the Tail Gates.

1) Layer 2 PFPC Dropper: The algorithm executed by the L2 dropper is similar to the PFPC component described in Sec. VI-A2. However, unlike that component, the L2 dropper has only two classes, high-priority (if PCP is 7) and low-priority (if PCP $<$ 7). The dropper component has 8 input and 8 output gates, where the 8^{th} input/output is connected to high-priority class and the rest to low-priority class. Although flows have reserved bandwidth and are also rate-limited in PFPC at network layer, it is still possible for DetNet flows to experience unwanted delay due to queue saturation at lower layers as they share a queue with other priority flows. A L2 dropper component is introduced here to avoid saturation for the priority queue, which is shared by high priority DetNet flows. All the queues are FIFO or PIFO which are length-aware and the components connected to it can access the state of the queue (for example, instantaneous length of queue in packets or bits). This component has access to queues and can take additional drop actions also based on the queue state (e.g., using ARED [14] or FRED [15] algorithms) only for non-DetNet flows, which do not have reserved bandwidth. Similar to the PFPC component, any DetNet flow will be limited to the configured maximum of reserved bandwidth.

2) Tail Gates: The Tail Gates component (discussed in Sec. III) provides gating mechanism at the tail of queues, to control enqueue operations. This component includes an enqueue-switch function providing cyclic enqueue operations in-order to simulate IEEE 802.1Qch or TAPS, where packets are enqueued in one queue during odd cycle and another queue during even cycle.

VII. DeNS Simulator Performance Evaluation

Before diving in the evaluation of the TAPS algorithm presented in Sec. IV-B, we made a preliminary evaluation of the overhead of DeNS simulator in comparison with standard INET framework and state-of-the-art TSN simulator NeSTiNg.

We show several performance metrics in Table. I for the simulations whose results are shown in Sec. VIII. We use the performance of INET as baseline and we show the deviation from this baseline for both DeNS and NeSTiNG. The performance metrics are averaged over 10 runs each lasting 1s (simulation time). In order to fairly make a comparison with INET and NeSTiNG, we use UDP only traffic type (instead of the TCP flows used in Sec. VIII), so that the transmission of packets is same for all simulators throughout the simulation time. It is very clear from Table. I that DeNS simulator has better performance compared to NeSTiNg simulator, specifically lower memory usage, less time required for a simulation, resulting in higher simulated-seconds per second. However, this is an effect due do packet-dropping by DeNS simulator at earlier stages, which reduces the number of events for several packets, leading to lower number of total events in the simulation. Indeed, we can see that NeSTiNg simulator, which has no traffic conditioner, has more number of total events for the same simulation, most of which are unnecessary for packets which get dropped at queues due

Table I: Performance Evaluation of DeNS in comparison with Standard INET framework and NeSTiNg

Simulator	*Memory (MB)*	*Cpu-Time (s)*	*Total-Events*	*Avg. events/sec*	*Avg. sim-sec/sec*	*Avg. # Events*	*Avg.FES (Future Event Set)*
DeNS	580.3 **(+97.89%)**	746.2 **(-12.31%)**	22723010 **(+78.35%)**	233186.9 **(+103.37%)**	0.0013 **(+14.02%)**	1671.9 **(-1.53%)**	194.2 **(+201.84%)**
NeSTiNg	620.4 **(+111.56%)**	852.5 **(+0.16%)**	25398116 **(+97.76%)**	226473.9 **(+97.51%)**	0.0011 **(-0.12%)**	1681.1 **(-0.99%)**	193.0 **(+199.98%)**
INET	293.2	851.1	18710510	114660.1	0.0011	1698.0	64.3

Table II: Detailed Configuration of the Simulated Network

End-Host	*Flow Type*	*Link BW*	*Application Traffic*	*Packet Transmission Cycle-Interval: 200 s (50 s/150 s) (odd/even)*	*Priority (PCP)*	*DSCP-Marker/PFPC-class/Bandwidth-Reservation/PCP-rewrite*	*Queue gate open/close (200 s Cycle)*
HP[0-8]	DetNet	1Gbps	Time scheduled 350B packet every 200 s (UDP) until 1s	odd cycle	7	AF11/Class-0/0.02Gbps/7	switch&router: Queue-7 50 s/150 s
HP[9]	DetNet	1Gbps	Time scheduled 350B packet every 200 s (UDP) until 1s	even cycle	7	AF11/Class-0/0.02Gbps/7	switch&router: Queue-7 50 s/150 s
HX[0]	DetNet	10Gbps	1Gbps (UDP) for 50 s in every 200 s cycle until 1s	even cycle	7	AF11/Class-0/1Gbps/7	switch&router: Queue-7 50 s/150 s
HX[1]	Misbehaving DetNet	10Gbps	10Gbps (UDP) until 1s	entire-time	7	AF12/Class-1/1Gbps/7	switch&router: Queue-7 50 s/150 s
HX[2]	High-load Best-effort	10Gbps	10Gbps (UDP) until 1s	entire-time	6	AF12/Class-1/1Gbps/5	switch: Queue-6 150 s/50 s router: Queue-5 150 s/50 s
HO[0-9]	Best-effort	1Gbps	TCP with 43.75MB of Application Data	n/a	6	BE/Class-3(class-N+1)/No-reservation/6	switch: Queue-6 150 s/50 s router: Queue-6 150 s/50 s

to congestion at later stages in the packet processing path. Therefore, we conclude that DeNS simulator scales similar to NeSTiNg simulator, with a better performance in the case of congestion at the level of priority queues. The comparison of performance with standard INET framework shows that, the ratio (simulator against INET) of performance metrics are in the range of well known simulators based on OMNET++ ([5], [12]).

With very wide configuration space of DeNS simulator, we leave it for our future work to investigate on the effects of various possible configurations on the performance of the DeNS simulator.

VIII. Simulation Results

In this section, we present the simulation results obtained using our DeNS simulator. We compare five mechanisms, namely: *i)* IEEE 802.1Qbv (**TAS**); *ii)* DeNS simulator with IEEE 802.1Qbv and a Traffic-Conditioner (**TAS+TC**); *iii)* DeNS with IEEE 802.1Qbv, Traffic-Conditioner, and Tail Queue Gates (**TAS+TATC**); *iv)* Time Aware PIFO Scheduling with a Traffic-Conditioner (**TAPS+TC**); *v)* IEEE 802.1Qch with Traffic-Conditioner (**CQF+TC**). The comparisons show the performance of combined services of IETF DetNet network layer and IEEE TSN link-layer.

The topology used throughout our simulations is shown in Fig. 5. It is consistent with the DetNet network topology examples in RFC 8655 [1]. There are three categories of hosts (left hand side) and sinks (right hand side) which are classified as: *Priority* (HP[0-9]) which generate DetNet flows; *X* (HX[0-2]) which generate either DetNet or UDP flows; *Other* (HO[0-9]) which generate best effort TCP flows. The class *X* is intended to emulate flows such as a misbehaving DetNet flow (violating time-schedule or throughput) or a high-load non-priority flow masked with a high-priority header value. The detailed configuration of DeNS module and the traffic model is shown in Table. II. All switch-router and router-router links have 10Gbps bandwidth, while host to switch links are as shown in Table. II. All the links have a propagation delay of 0.1 s. All the switches and routers have a processing delay of 5 s. The cycle time for all mechanisms are as shown in Table II, except CQF for which a cycle of 100 s (50 s/50 s) is used. We reduced the cycle time for CQF as we observed that a 200 s cycle simply increased the end-to-end latency without any benefits. Nevertheless, accurately determining the cycle time is one of the challenges faced by CQF [8]. Our traffic-model consists of a mixture of DetNet periodic and sporadic traffic, high-throughput sporadic traffic, misbehaving DetNet traffic, and best-effort UDP or TCP traffic.

1) End-to-End Latency: The theoretical minimum end-to-end latency in our topology for DetNet flows is the sum of transmission delays and propagation delays from a host HP[ ] to sink SP[ ] amounting to 28.081 s. In Fig. 6, we present boxplots of the end-to-end latency for DetNet and misbehaving DetNet flows in our simulations. With the original TAS, we can observe that there is no guaranteed service due to congestion. When adding a traffic conditioner (TAS+TC), we can observe that latency is bounded (with slightly higher bounds) for only scheduled periodic flows. Fig. 6 clearly shows the well-known problem of TAS, the waiting time for packet arrivals out-of reserved time-slot (HP[9]) and high variation in latency for high-throughput random scheduled flows (HX[0]) as described in Sec. III. For TAS+TATC, where all the packets arriving out-of the reserved time-slot are dropped, low bounded latency can be guaranteed for periodic flows. This clearly shows the need for separate queues. We can observe that cyclic queuing and forwarding with traffic-conditioner (CQF+TC), achieves bounded latency for all the scheduled flows. However, CQF+TC has higher bounds due to non-deterministic packet arrivals from HX[0] and HX[1]. Finally, with our proposed TAPS+TC, packets of DetNet flows (including scheduled periodic; sporadic, and sporadic high-

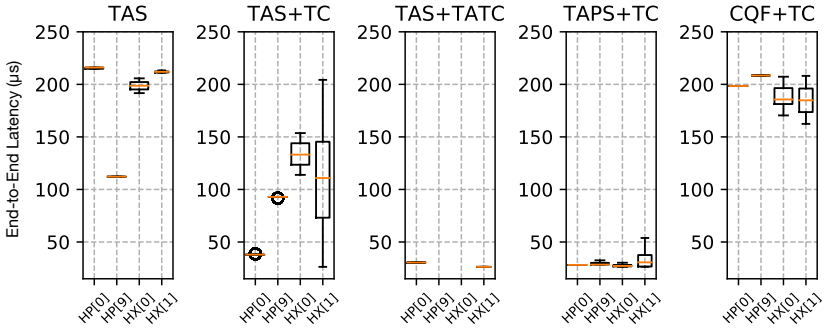


Figure 6: End-to-End latency for DetNet flows: periodic odd-cycle (HP[0-8], showing only HP[0], the other flows having the same behavior), periodic even-cycle (HP[9]), periodic even-cycle max-reserved bandwidth (HX[0]) and misbehaving periodic (HX[1]).

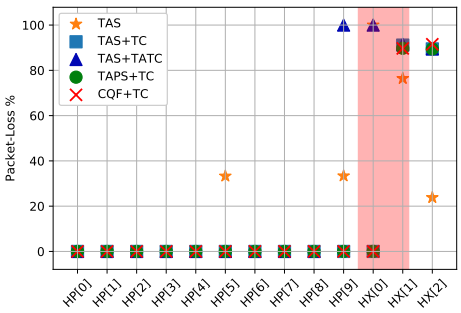


Figure 7: Throughput and packet loss for DetNet and other UDP flows in the network, showing zero packet loss for all DetNet flows with DeNS simulator traffic-conditioning.

throughput flows) have very low end-to-end latency with mean 28.176 s and bounded to 0.740 s variation, which is close to the theoretical minimum. The misbehaving flow although rate-limited, experiences high variation in end-to-end latency due to a limitation of TAPS+TC, i.e., when packets arrive close to the end of a cycle, they are enqueued in a queue for which the gate closes before its transmission. However, this effect can be reduced by changing to asynchronous enqueue/dequeue operations, similarly as for CQF [4], only for a short duration t before every cycle change. This allows additional t time for last enqueued scheduled packet to be dequeued. However, choosing t requires further evaluations as it adds to latency bounds. Solving this issue is left as future work.

2) Throughput and Packet-Loss: One of the key aspects of DetNet is to achieve zero packet loss. In Fig. 7, we see that the traffic conditioner of DeNS simulator is able to guarantee zero packet loss for all the DetNet flows, even in the presence of congestion due to misbehaving flows. This is achieved through packet dropping enforcing guaranteed reserved bandwidth for priority flows. As a result, all the DetNet flows have a guaranteed packet delivery when traffic-conditioner is used. This shows the effectiveness of our PFPC module. We can also see that the misbehaving DetNet flow HX[1] is limited in throughput due to violation of DetNet policies and non-DetNet flow HX[2] is limited in throughput due to violation of maximum allowed bandwidth (1Gbps). On the other hand, we can see that TAS without traffic-conditioner cannot guarantee zero packet loss for DetNet flows.

3) TCP Performance: TCP flows in our simulations have a PCP value of 6, hence, all experiencing congestion at Router-A (cf., Fig. 5). TCP flows are seen as unknown flows by the classifier and arrive at Class $N+1$ of the PFPC module, where the unused bandwidth tokens are shared among all TCP flows. In particular we use Weighted-Round-Robin (WRR) algorithm to share the tokens among the token buckets of all TCP flows. The PCP value of 6 for TCP flows remain unchanged, while the PCP value for UDP flow HX[2] is changed to 5 in the PCP Marker. This ensures priority for TCP flows, at the same time not affecting the requirements for HX[2] as it is classified into class 1 with 1Gbps bandwidth limitation. Thanks to the PFPC module, TCP flows have some available bandwidth, while thanks to the PCP Marker it is possible to separate TCP and UDP flows (or any high-load non-responsive flows). Due to the queue gating mechanism at lower-layers (IEEE 802.1Qbv), it is not possible to avoid losses for packets transmitted when the gates are closed and the queue length is full. We expect TCP to adjust its transmission rate based its congestion avoidance mechanism.

Fig. 8 and Fig. 9 show the variation of throughput (measured at application layer of sinks SO[0-9]) and RTT variation measured by TCP at Hosts HO[0-9]. In Fig. 8, we can see that with DeNS simulator (TAPS+TC, TAS+TATC, TAS+TC), all the TCP flows never experience starvation due to low RTO value calculated based on the RTT measurements. We conclude that our traffic conditioner of DeNS simulator is not only able to guarantee bandwidth reservation for DetNet flows but also able to fairly share the remaining bandwidth among other flows. In the case of TAS without traffic-conditioner, all the TCP flows attempt to transmit and measure a very high RTTVAR, due to congestion at queue 6 of Router-A, which is shared by a non-responsive UDP flow HX[1]. As a result, all the TCP flows back-off with a high RTO leading to temporary starvation. As pointed out in Sec. III, in the case of CQF+TC, even with a traffic-conditioner, one TCP flow experiences temporary starvation due to high RTT variance.

IX. Conclusion

Recent years have shown a growing interest in networks able to accommodate real-time constrains, with IEEE developing the TSN technology at layer 2, while the IETF works on the DetNet at network layer. However, the impact of misbehaving flows in TSN/DetNet has been overlooked insofar and lacks an appropriate solution. To this end, and as a first contribution of this paper, we proposed TAPS, a novel queuing and scheduling mechanism using Push-In-First-Out (PIFO) queues and able to partially solve the problem of misbehaving flows. We implemented our proposal on OMNET++ adding code that actually allows to simulate the full DetNet/TSN

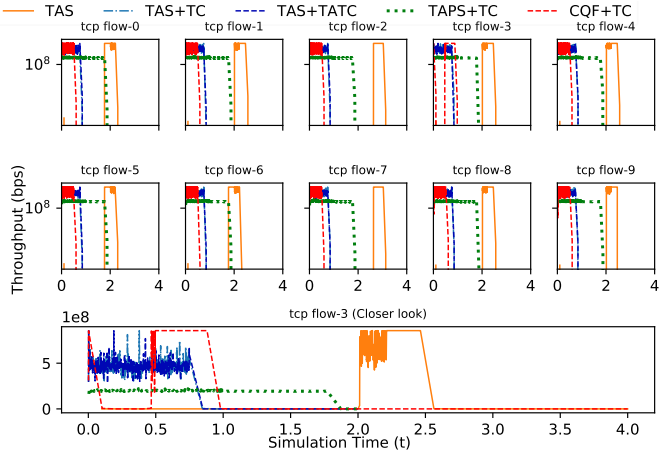


Figure 8: Throughput variation of TCP flows in the presence of DetNet, misbehaving DetNet, and non-DetNet flows.

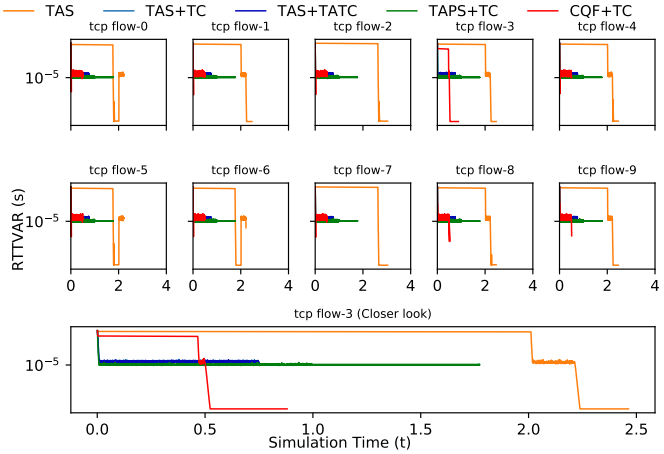


Figure 9: RTT variation of TCP flows in the presence of DetNet, misbehaving DetNet, and non-DetNet flows.

protocol stack, creating DeNS, the second main contribution of this paper. DeNS simulator provides the functionalities of IEEE 802.1Qav, IEEE 802.1Qbv, IEEE 802.1Qch and the building blocks necessary to develop any general queuing and scheduling algorithm in the context of TSN link-layer and provides extensions to network-layer services such as traffic-conditioning, resource allocation, classification based on DSCP and mapping to PCP etc., providing a framework for easy evaluations of DetNet. The performance of DeNS is in line with the INET framework and TSN simulator NeSTiNg, showing that it is scalable and has an overhead similar to other Time Sensitive Networks simulators based on OMNET++.

We clearly show with simulation results that our solution is able to satisfy key aspects of DetNet, i.e., guarantee zero packet loss, low latency and jitter for DetNet flows, while at the same time allowing other flows to co-exist with DetNet without resulting in starvation. We also show by simulations that our proposed mechanism TAPS is able to guarantee services for all the scheduled flows even in the presence of misbehaving scheduled flow.

Our simulator currently misses some DetNet features, such as packet and flow replication and elimination, fault mitigation, which are left for future work. The source code and full documentation of DeNS simulator is made available online on GitHub (`https://github.com/vamsiDT/DeNS`).

Acknowledgments: The work presented in this article benefited from the support of NewNet@Paris, Cisco's Chair ''Networks for the Future'' at Telecom ParisTech (https://newnet.telecom-paristech.fr). Any opinions, findings or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of partners of the Chair.

REFERENCES

[1] N. J. Farkas, Finn, P. Thubert, and B. Varga, *Deterministic Networking Architecture*, ser. RFC 8655. Internet Engineering Task Force - (IETF), October 2019.

[2] ''Time-sensitive networking task group,'' 2019. [Online]. Available: http://www:ieee802:org/1/pages/tsn:html

[3] F. Norman, ''Time-sensitive and deterministic networking whitepaper,'' Tech. Rep., July 2017. [Online]. Available: https://mentor:ieee:org/802:24/dcn/17/24-17-0020-00-sgtg-contribution-time-sensitive-and-deterministic-networking-whitepaper:pdf

[4] ''Ieee standard for local and metropolitan area network--bridges and bridged networks,'' *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)*, pp. 1--1993, July 2018.

[5] J. Falk, D. Hellmanns, B. Carabelli, N. Nayak, F. Dürr, S. Kehrer, and K. Rothermel, ''Nesting: Simulating ieee time-sensitive networking (tsn) in omnet++,'' in *2019 International Conference on Networked Systems (NetSys)*. IEEE, 2019, pp. 1--8.

[6] M. Pahlevan and R. Obermaisser, ''Evaluation of time-triggered traffic in time-sensitive networks using the opnet simulation framework,'' in *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*. IEEE, 2018, pp. 283--287.

[7] ''Inet framework,'' 2019. [Online]. Available: https://inet:omnetpp:org/

[8] A. Nasrallah, V. Balasubramanian, A. S. Thyagaturu, M. Reisslein, and H. Elbakoury, ''Cyclic queuing and forwarding for large scale deterministic networks: A survey,'' *CoRR*, vol. abs/1905.08478, 2019. [Online]. Available: http://arxiv:org/abs/1905:08478

[9] A. Sivaraman, S. Subramanian, A. Agrawal, S. Chole, S.-T. Chuang, T. Edsall, M. Alizadeh, S. Katti, N. McKeown, and H. Balakrishnan, ''Towards programmable packet scheduling,'' in *Proceedings of the 14th ACM workshop on hot topics in networks*. ACM, 2015, p. 23.

[10] J. Jiang, Y. Li, S. H. Hong, A. Xu, and K. Wang, ''A time-sensitive networking (tsn) simulation model based on omnet++,'' in *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2018, pp. 643--648.

[11] ''Core4inet_background – core simulation models for real-time networks,'' 2019. [Online]. Available: http://core4inet:core-rg:de/trac/wiki/CoRE4INETn_Background

[12] P. Heise, F. Geyer, and R. Obermaisser, ''Tsimnet: An industrial time sensitive networking simulation framework based on omnet++,'' in *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2016, pp. 1--5.

[13] ''Omnet++ discrete event simulator,'' 2019. [Online]. Available: https://www:omnetpp:org/

[14] S. Floyd, R. Gummadi, S. Shenker *et al.*, ''Adaptive red: An algorithm for increasing the robustness of red's active queue management,'' 2001.

[15] W.-J. Kim and B. G. Lee, ''Fred-fair random early detection algorithm for tcp over atm networks,'' *Electronics Letters*, vol. 34, no. 2, pp. 152--154, Jan 1998.