Large Scale Deterministic Networking: A Simulation Evaluation

Ahmed Nasrallah, Venkatraman Balasubramanian, Akhilesh Thyagaturu, Martin Reisslein, and Hesham ElBakoury

***Abstract*—The use of Ethernet switched networks usually involves best effort service. A recent effort by the IEEE 802.1/3 TSN group has sought to standardize the Ethernet data-link protocol such that it operates on a deterministic service in addition to the best effort service targeting Operational Technology applications, e.g., industrial control systems. This paper investigates the Cyclic Queueing and Forwarding (CQF) and Paternoster scheduling protocols in a typical industrial control loop with varying propagation delays emulating large scale networks. Our main findings for CQF and Paternoster are that CQF has an advantage towards real-time streams with hard-deadlines whilst Paternoster is for streams with more relaxed deadlines but can operate without time synchronization.**

***Index Terms*—Large Scale Deterministic Networking, Time Sensitive Networking, Cyclic Queueing and Forwarding, Paternoster Algorithm.**

I. Introduction

A. Motivation

The open access and ubiquitous use of Ethernet switched networking technology is propelling the use of full-duplex Ethernet standards in LANs and WANs for a variety of real-time and traditional background applications on converged Ethernet switches and links. The use of Ethernet in industrial environments provides increased bandwidth and better interoperability among other benefits. While the idea of using Ethernet devices in Operational Technology (OT), i.e., automotive, avionics, and industrial control systems, is not new (see Fig. 1), the IEEE 802.1Q, Time-Sensitive Networking (TSN), task force recently released a set of standards that augment standard Ethernet switches providing determinism and low latency communication ideal for OT applications.

A key question that needs defining is what constitutes a deterministic system or determinacy in the context of networking and communication? We can establish that it does not mean increased throughput or reduced latency. We conclude that a deterministic system is a system in which *no* randomness is involved and therefore can be modeled or characterized to produce the same output from the same starting conditions (i.e., initial state).

In this report, we implement and utilize Cyclic Queuing and Forwarding (CQF), and the Paternoster scheduling mechanism on a standard industrial control closed-loop unidirectional ring topology. Furthermore, we study and analyze the scheduling mechanism's efficacy for different propagation delays and traffic intensity emulating large-scale networks with both sporadic and periodic traffic. The main goal is to ensure the deterministic attributes governed by the scheduling mechanism used in ensuring proper TSN QoS.

Please direct correspondence to M. Reisslein (reisslein@asu.edu).

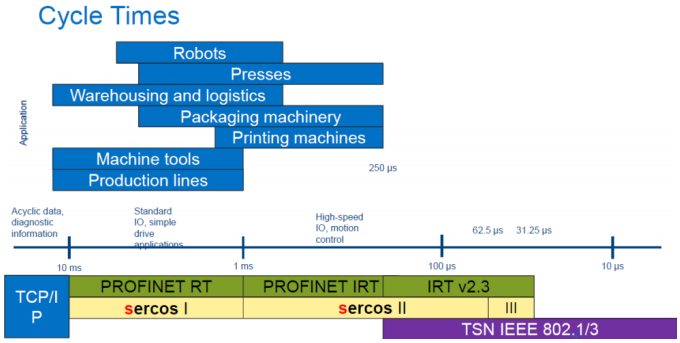

Fig. 1: Industrial QoS between different protocols related to OT applications.

B. Related Work

Groundwork on CQF, which was also previously known as Peristaltic shaper, was conducted by Thangamuthu et al. [1]. Moreover, Thiele et al. [2] have conducted a theoretical analysis of the blocking factors for CQF and TAS.

Zhou et al. [3], [4] have conducted a simulation study on Paternoster, but only for one-hop transmission (they did not consider a full multi-hop network).

In [5] authors model a routing problem in Time sensitive network as an ILP in time sensitive networks.

In [6] authors propose a joint optimization problem of routing and scheduling in one step.

In [7] authors propose a bandwidth optimization based queuing technique.

C. Contributions

We make the following contributions:

i) We implement both standard CQF and Paternoster scheduling models.
ii) We comprehensively evaluate and analyze the two models for both sporadic and periodic sources with cross-interference of BE traffic and varying propagation delays emulating large-scale networks.
iii) We elucidate recommendations and limitations of each model according to the results.

D. Organization

This article is organized as follows. Section II provides the necessary background information to understand the mecha-

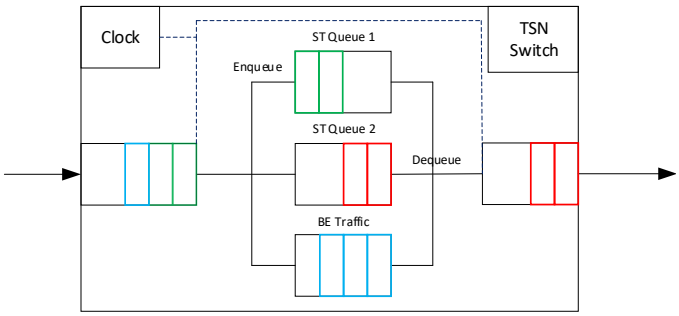

Fig. 2: Simplified CQF Mechanism in TSN switch with two ST queues

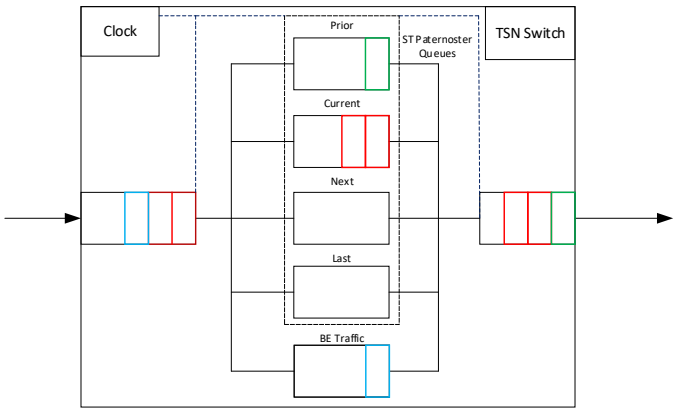

Fig. 3: Simplified Paternoster Mechanism in TSN switch

nisms of CQF and Paternoster. Section III describes and illustrates the simulation environment, network/traffic model, and shows the results collected and metrics involved in analyzing the scheduling mechanisms. Finally, Section IV concludes the document.

II. Background: IEEE 802.1 Time Sensitive Networking

This section provides a brief background overview on TSN standardization, specifically CQF and Paternoster. TSN is a suite of standards aimed at applying deterministic behavior to the traditional best-effort Ethernet standards. Due to the scope of this project, we refer to the survey [8] for further reading on TSN standardization and research areas.

A. CQF

The published IEEE 802.1Qch (CQF) [9] standard proposes to coordinate enqueue/dequeue operations within a switch in a cyclic fashion. Fig 2 shows a simplified illustration (or snapshot) of the CQF mechanism. The CQF cyclic operation results in an easily calculable latency bound governed by the chosen Cycle Time and the number of end-to-end hops between communicating parties. In CQF, time is divided into slots or intervals. For a given traffic class, two queues are used to enable the cyclic property. Frames arriving in interval x will be transmitted in interval $x+1$. Similarly, frames arriving in interval $x+1$ are transmitted in interval $x+2$, and so on. The maximum and minimum frame delay bounds in CQF with H and CT representing the number of hops and cycle time duration, respectively, are

$$D_{Max} = (H+1) \times CT \quad (1)$$

$$D_{Min} = (H-1) \times CT. \quad (2)$$

Two queues are used to handle enqueue and dequeue operations in separate time intervals. For example, frames arriving in even intervals will be enqueued in one queue, while the frames that were enqueued during the previous interval will be transmitted from the other queue. In CQF, a frame sent by an upstream switch in cycle x must be received by the downstream at cycle x, i.e., the propagation delay must be less than the selected cycle time. Therefore, the cycle time is constrained by the link distance (network scale in general). Essentially, the smaller the network size, the easier it is to guarantee the TSN QoS by CQF. Additionally, CQF has a few challenges that limit its viability, such as i) accurately determining the appropriate cycle time, and ii) cycle duration misalignment where due to processing and transmission delays, a frame can be received in the wrong cycle (i.e., be placed in the wrong outbound queue).

B. Paternoster

The Paternoster algorithm is a proposed enhancement by Mike Seaman [10] to standard CQF. Fig 3 shows a simplified illustration (or snapshot) of the Paternoster mechanism. Paternoster provides bounded latencies and lossless service for flows that are successfully registered across the network without a time synchronization requirement. For each egress port, the Paternoster protocol defines a counter for stream reservation and four output queues (*prior*, *current*, *next*, *last*), whereby all switches under Paternoster operate under an *epoch* timescale where the stand/end of the *epochs* are not synchronized with other switches. In each *epoch* window, frames in the *prior* queue are transmitted first until all frames are transmitted. Once the *prior* queue is depleted, the *current* queue is selected for transmission until the end of the current epoch. While frames are being transmitted from the *prior* and *current* queues, received frames are enqueued in the *current* queue until the bandwidth capacity is reached for the current *epoch*. Any additional frames are enqueued in the *next* and *last* queues in a similar manner, i.e., until the reservation capacity for the current epoch is reached while additional frames are dropped if the *last* queue is completely reserved for the current *epoch*. Note that all ST traffic streams are given guaranteed bandwidth, while BE traffic is given the leftover bandwidth. When a new epoch starts, the previous *current* queue operates as the *prior* queue while the *next* and *last* queues become the *current* and *next* queues, respectively. The previous *prior* queue (which should be empty, and if not, we purge all the contents and register the packets as lost) becomes the new *last* queue. The Paternoster operation repeats at each *epoch*, while the four queues alternate during each *epoch*. While four queues are expected to be sufficient for many LDN scenarios, very long propagation delays may necessitate that another queue

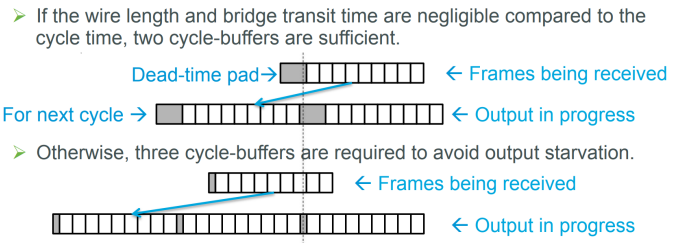

Fig. 4: Standard vs. 3-Queue CQF example illustration.

into the past and another queue into the future are added, for a total of six queues [10].

In summary, the Paternoster approach uses four queues that alternate every epoch (also known as cycle) using only frequency synchronization, i.e., the epoch duration is the same across the nodes. In contrast to CQF, the Paternoster approach gives up some delay predictability in exchange for not requiring clock synchronization and for reducing the average delay.

The evaluations reported for Paternoster in this report considered random time shifts of the equal-duration cycles in the switches. In particular, each switch had an independent uniformly distributed time shift between zero and the cycle time with respect to a common time base.

C. 3-Queue CQF

A critical requirement for the standard CQF is that a frame sent during a cycle has to be received during the same cycle such that the worst case delay is constrained by the cycle time and hop count. The 3-Queue CQF has been proposed to handle networks that have propagation delays that approach and exceed the cycle time [11].

When traffic arrives in the wrong cycle, a 3^{rd} queue is needed to handle such traffic so as to prevent disruption to traffic that conforms to the requirement for CQF. This is illustrated in Fig. 4. Though the general or principle idea behind the 3-Queue CQF is interesting, some questions remain that need solutions so that a full-fledged implementation and evaluation is possible.

How would the 3^{rd} queue (or waiting queue) be used in such a environment without affecting other traffic? Every cycle is needed to send traffic from an egress port, especially for periodic traffic. Therefore, when should traffic that gets enqueued into the waiting queue be dequeued? How would the dead-time be calculated or computed? If the propagation delay exceeds the cycle time for all periodic traffic, wouldn't this delay be consistent for all traffic and therefore act as a constant in the overall worst case delay?

Our tests in Section III indicate that a propagation delay of 50μs for a 50μs cycle time (where ST is given 25μs) gives twice (from 200μs to 400μs) the max or worst case delay than a propagation delay of 25μs. We can hypothesize that for sporadic sources, we can use the strict priority scheduler between the dequeuing queue and the waiting queue, so that any traffic in the waiting queue can be transmitted if no traffic is waiting in the dequeuing queue.

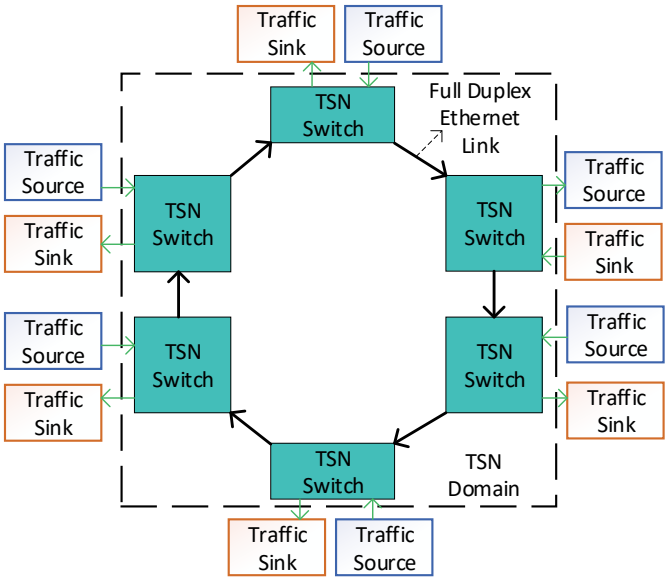

Fig. 5: Unidirectional Ring Topology

TABLE I: Simulation Parameters

Key	Symbol	Value
Simulation duration	Sim_{limit}	100 seconds
Initialized cycle time	GCL_{CT}	50 μs
Initialized gating ratio	ST^{R}_{init}	50% (i.e., 25 μs)
Total streams	γ	6
Stream duration	τ	100 seconds
Link propagation delay	α	500 ns, 25μs, 50μs
Number of frames/packets per cycle for periodic traffic	π	$1-40$
Sporadic traffic intensity	ρ_I	$0.1-2.0$ Gbps
ST sources	S	6
ST stream hop count	TTL	3
Hurst parameter	H	0.5
Queue size	Q_{size}	512 Kb

III. Performance Evaluation

A. System Overview and Simulation Setup

This section describes the simulation setup and model for both standard CQF and the Paternoster scheduling protocols. Furthermore, the topology and simulation scenarios will be presented. Throughout, we employ the OMNet++ [12] simulation environment.

1) Network Model: The topology used to test the CQF and Paternoster scheduling mechanisms is modeled as shown in Fig. 5. Table. I shows the simulation parameters used in testing CQF and Paternoster in the unidirectional ring. Each switch-to-switch link operates as a full-duplex Ethernet link with a capacity (transmission bitrate) $R = 1$ Gbps. Each switch can act as a gateway for a number of traffic sources and one sink. The propagation distance is varied between 500 ns and 50μs. Each switch operates either CQF or Paternoster scheduling between switch to switch egress ports.

2) Traffic Model: We consider periodic (pre-planned) traffic and sporadic self-similar Poisson ($H = 0.5$) traffic for ST

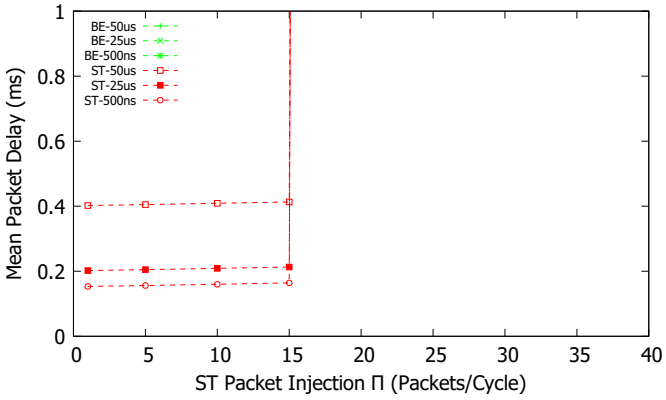

Fig. 6: Periodic traffic ST sources .. CQF mean packet delay

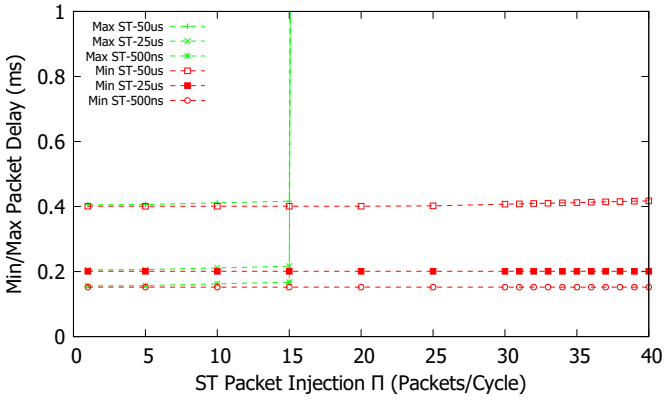

Fig. 7: Periodic traffic ST sources .. CQF max/min packet delay

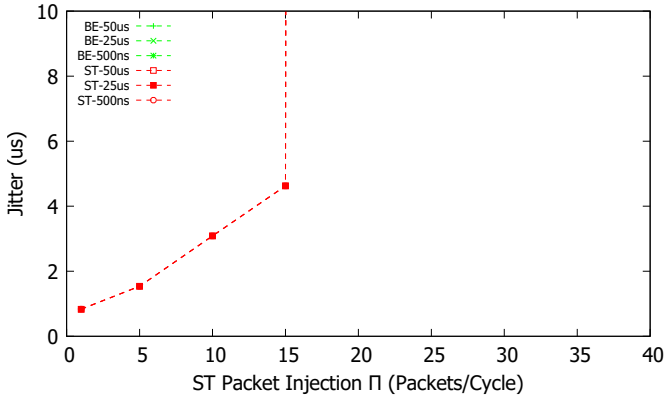

Fig. 8: Periodic traffic ST sources .. CQF Jitter

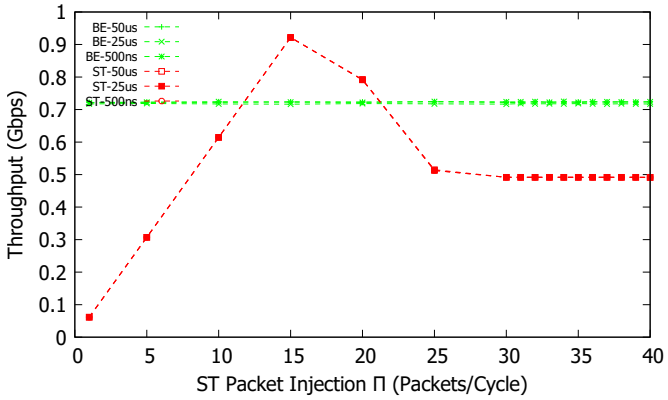

Fig. 9: Periodic traffic ST sources .. CQF Average Throughput

traffic, while solely sporadic traffic for BE. Six sources are used to generate traffic each attached to a TSN switch gateway. A single stream is initiated at the start of the simulation for the entire duration of the simulation. Each frame/packet's destination address is specified by the switch to switch hops around the ring, which is predefined to 3 hops as shown the Table. I. The size of a frame is 64 bytes for ST and 580 bytes for BE. The traffic intensity is varied in each simulation run where the ST injection rate $(1-40)$ is used for periodic ST traffic and the ρ_I traffic intensity. Note that the BE traffic intensity in periodic ST source tests is set to 1.0 Gbps.

B. CQF

1) Periodic Results: Fig. 6 and Fig. 7 show the mean and min/max delays, respectively, for periodic ST traffic using CQF based scheduling under different propagation delays. The BE traffic intensity is set to a constant value of 1.0 Gbps and exhibits the same mean delay of 28 ms for all ST injection rates (due to TAS isolation). As the ST periodic traffic intensity increases, both the mean and max delays are constant up to an ST packet injection rate of $\pi = 16$ packets/cycle, which causes an immediate spike in both mean and max delays due to over-utilizing the link resources, no preventive measures of admission control policies, and none-adaptive TAS slot ratios that change according to the bandwidth consumption. Since CQF gives a simple method of calculating the worst-case end-to-end delay of a stream (shown in Section II), the max delay shown in Fig. 7 illustrates that for the CQF mechanism, the delay is a function of and bounded by the number of hops and GCL time. More precisely, since the cycle time (GCL) is set to 50μs, the total worst case delay for a three hop stream is $50 \cdot (3+1) = 200$ μs which is shown in both figures (except for networks initialized with 50μs propagation delays).

Furthermore, as the propagation delay is increased and approaches the cycle time, the end-to-end delay approaches the CQF worst-case delays, i.e., the min/max and mean delays are bounded and characterized by the number of hops and cycle time. Note that the ST gating ratio (due to TAS operating in the egress port) is half the cycle time (25μs), while the BE traffic is allocated the rest of the transmission time opportunity. When the propagation approaches the cycle time, it is considered twice the ST gating ratio which translates to twice the worst case delay of 400μs.

Fig. 8 shows the network jitter between a source and sink. The jitter is calculated as the standard deviation of the mean delay. As shown in the figure, jitter is around 4μs but then spikes very quickly due to over-utilization of resources and consequently causing congestion.

Fig. 9 and Fig. 10 show the throughput and loss respectively. As the ST injection rate is increased, the throughput increases linearly. However, the throughput sharply declines due to the

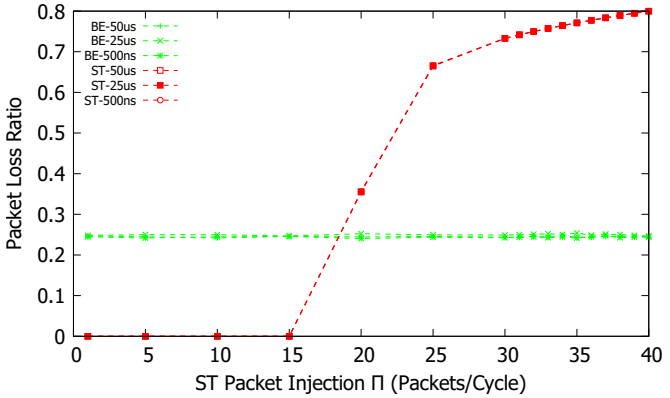

Fig. 10: Periodic traffic ST sources .. CQF Loss Packet Ratio

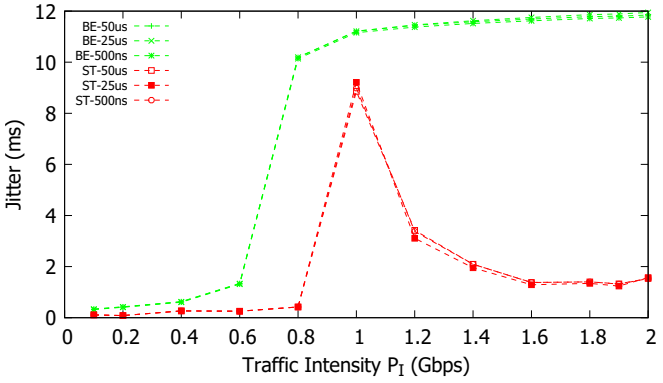

Fig. 13: Sporadic traffic ST sources .. CQF Jitter

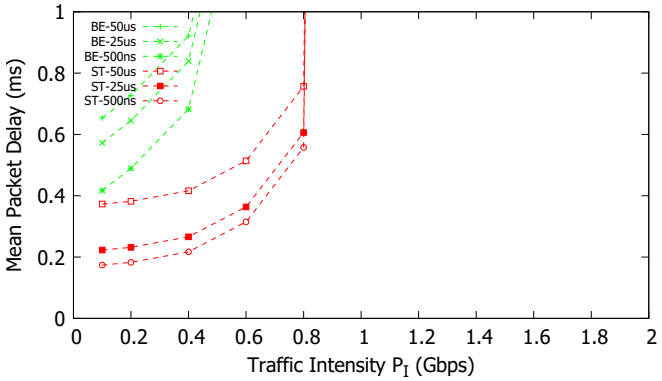

Fig. 11: Sporadic traffic ST sources .. CQF mean packet delay

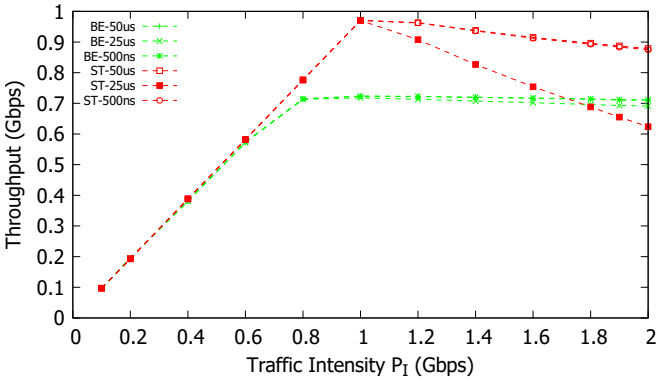

Fig. 14: Sporadic traffic ST sources .. CQF Average Throughput

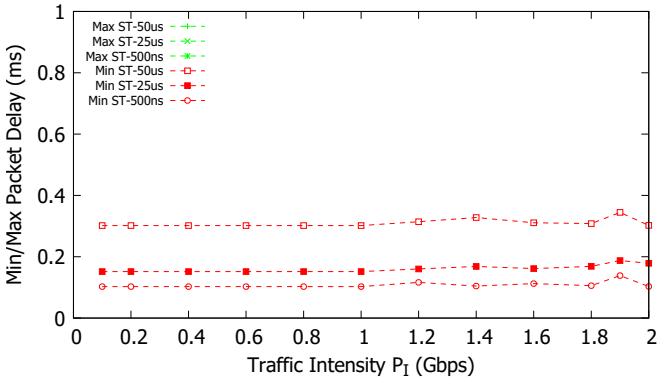

Fig. 12: Sporadic traffic ST sources .. CQF max/min packet delay

congestion caused by injecting more bits than the ST slot can handle within each cycle (16 packets or 1892 bits can be sent by one source per cycle into the network).

In terms of packet/frame loss, BE traffic experiences more or less the same loss due to having the same traffic intensity (1.0 Gbps) in all runs (around 0.25% loss). ST, on the other hand, stays constantly at 0 loss until $\pi = 16$ rate. Due to congestion, the loss increases sharply with higher ST traffic intensity.

2) Sporadic Results: Fig. 11 and Fig. 12 show the mean and min/max delays respectively for sporadic ST sources. In contrast to periodic ST traffic sources, the use of sporadic traffic with uncontrollable bursts can severely degrade the operation of CQF as shown in both figures. The mean delay for both sporadic traffic classes quickly increases as the traffic intensity increases. The TSN QoS (bounded min/max delays, zero loss, and low jitter) are violated mainly due to the uncontrollable bursts in the sporadic ST sources.

Fig. 13 shows the network jitter between source and sink. Similar to the mean and min/max delays figures, the jitter is much higher compared to the periodic jitter results.

Fig. 14 and Fig. 15 show the throughput and packet loss for sporadic ST sources. Throughput increases as the traffic intensity increases up to traffic intensity, $\rho_I = 1.0$, which causes a large drop in throughput due to congestion in the network. Similarly, the packet loss shows large increase after $\rho_I = 1.0$ for ST, while BE starts to lose more packets earlier.

C. Paternoster

1) Periodic Results: Fig. 16 and Fig. 17 show the mean and max/min delays for periodic ST sources and sporadic BE sources using switches that operate Paternoster. Initially, we observe from Fig. 16 that the mean delays for ST are lower when compared against the CQF performance. However,

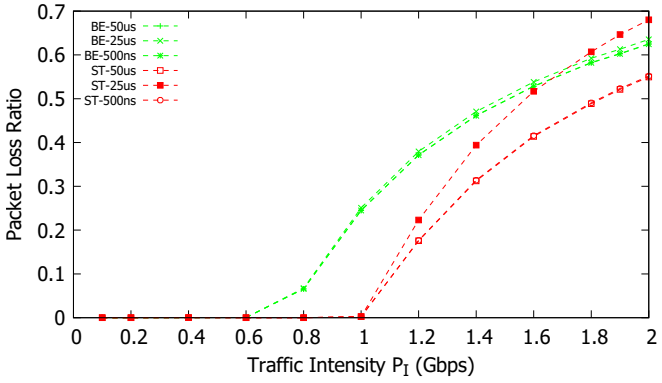

Fig. 15: Sporadic traffic ST sources .. CQF Loss Packet Ratio

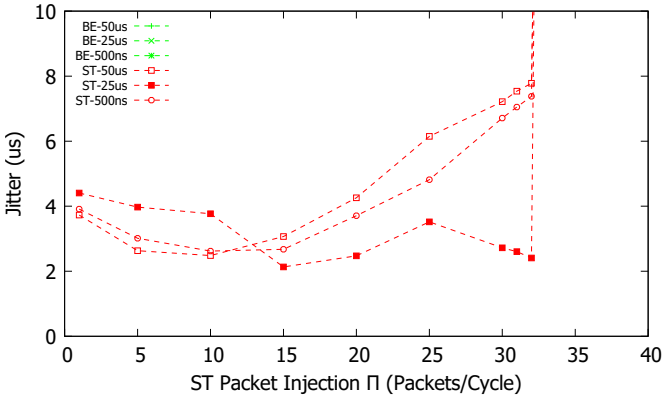

Fig. 18: Periodic traffic ST sources .. Paternoster Jitter

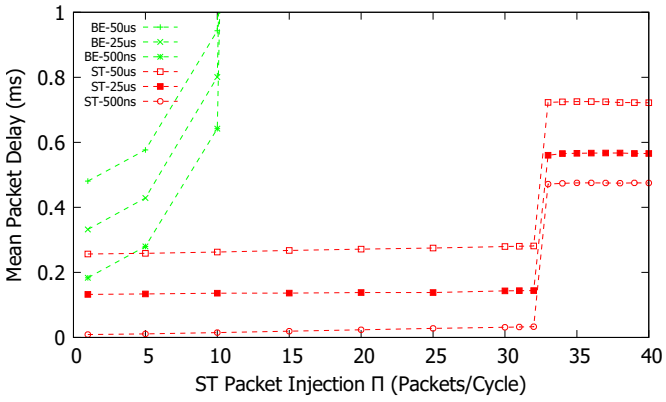

Fig. 16: Periodic traffic ST sources .. Paternoster mean packet delay

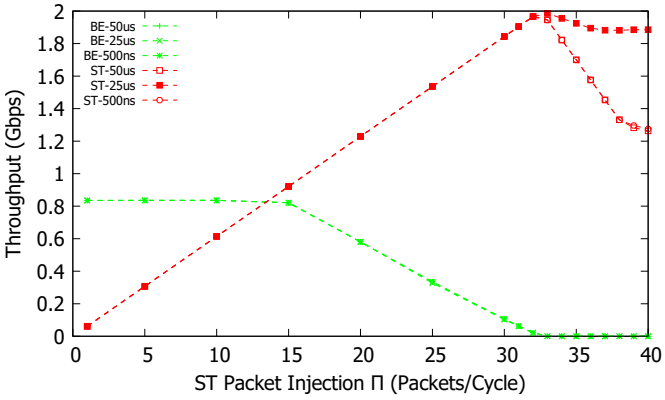

Fig. 19: Periodic traffic ST sources .. Paternoster Average Throughput

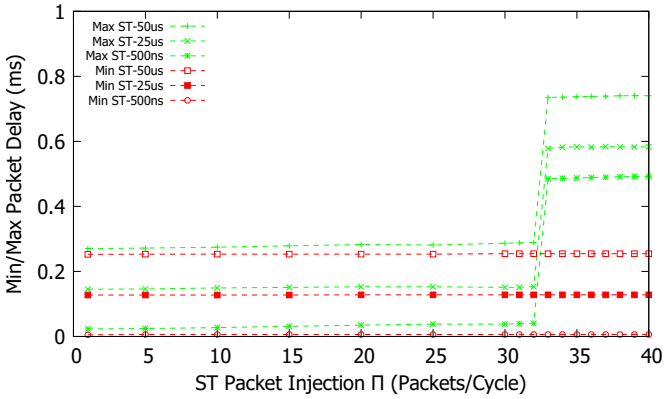

Fig. 17: Periodic traffic ST sources .. Paternoster max/min packet delay

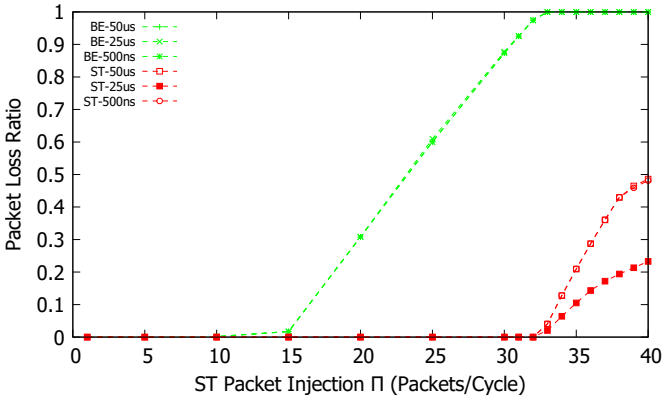

Fig. 20: Periodic traffic ST sources .. Paternoster Loss Packet Ratio

BE get starved by ST when $\pi = 33$ since all transmission opportunities during an epoch/cycle are consumed by ST. ST's delay stabilizes after the spike due to purging the prior queue in Paternoster.

Fig. 18 shows network jitter. While the jitter is comparable to the CQF protocol, the varying changes as the traffic intensity increases show the unpredictability in Paternoster compared to CQF.

Fig. 19 and Fig. 20 show the throughput and packet loss experienced at the sink. At $\pi = 16$, we see the BE traffic (with $\rho_I = 1.0$ Gbps) starts to drop proportional to the increase in ST before being starved at $\pi = 33$. Any additional increase causes packet loss and congestion which drops the throughput to below optimum levels. Similarly, the loss shows a complement of the throughput and at $\pi = 16$, the BE traffic starts to accumulate loss linearly as the traffic intensity keeps increasing.

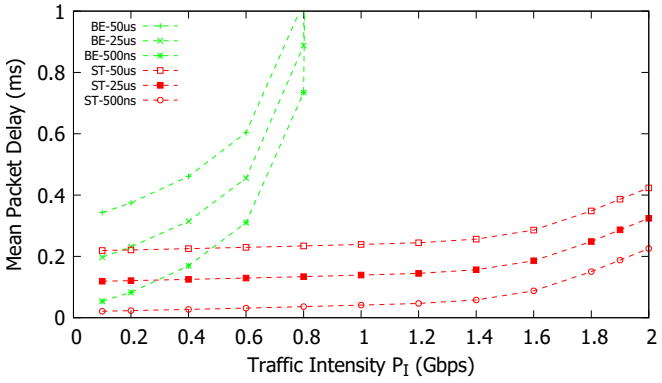

Fig. 21: Sporadic traffic ST sources .. Paternoster mean packet delay

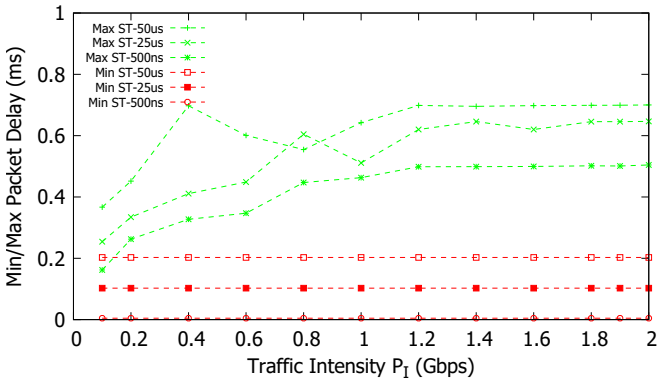

Fig. 22: Sporadic traffic ST sources .. Paternoster max/min packet delay

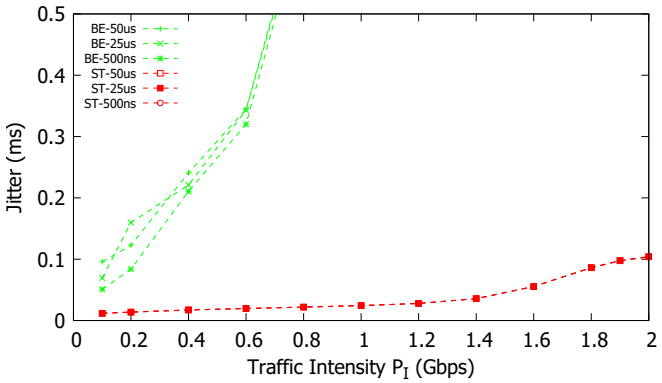

Fig. 23: Sporadic traffic ST sources .. Paternoster Jitter

2) Sporadic Result: Fig. 21 and Fig. 22 show the mean and min/max delays for sporadic ST sources using Paternoster. Compared to the periodic results, the Paternoster performs better for sporadic traffic sources. However, accurately predicting the worst case delays still remains difficult compared to CQF. Since strict priority scheduling is employed at the egress port, the max or worst case delays are highly unpredictable compared to CQF and also Paternoster under the periodic ST sources.

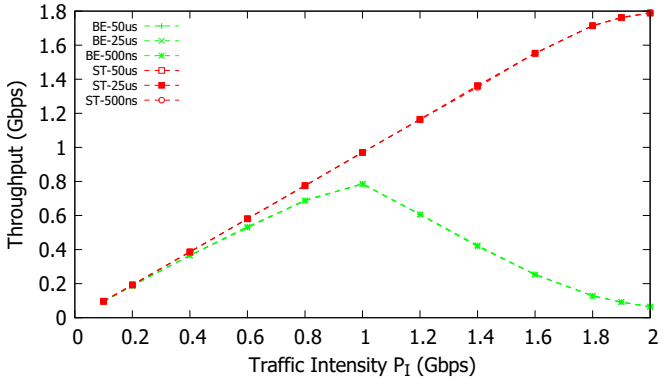

Fig. 24: Sporadic traffic ST sources .. Paternoster Average Throughput

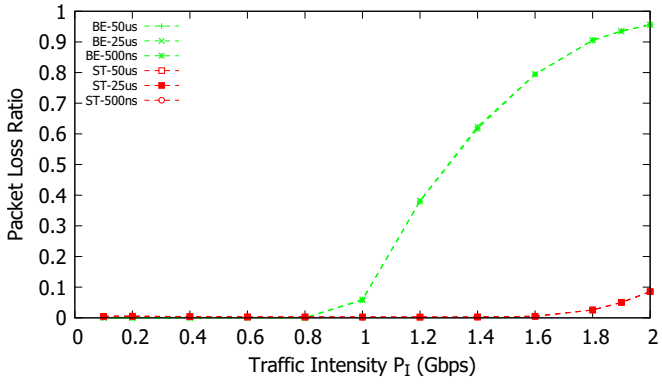

Fig. 25: Sporadic traffic ST sources .. Paternoster Loss Packet Ratio

Fig. 23 shows the network jitter. Since strict priority scheduling is used to arbitrate between competing traffic classes, BE (lower priority) can block ST (higher priority) if the port is currently transmitting BE traffic when ST waits for the transmission to finish. This causes higher unpredictable jitter, which is seen Fig. 23.

Fig. 24 and Fig. 25 show the throughput and loss at the sink. Similar to the periodic case, BE traffic throughput decreases after 1.0 Gbps while it's loss increases. In contrast, the ST throughput continues to increase, but ST does experience some loss after about an injection rate π corresponding to 1.6 $Gbps$.

D. CQF and Paternoster Comparison Analysis

Both the CQF and Paternoster scheduling algorithms attempt to enhance the data-link layer with some form of deterministic behavior. CQF coordinates the ingress and egress operations to provide bounded delays and zero traffic loss that conforms to its reservation, while Paternoster uses four queues per port to spread traffic burstiness/bunching and guarantees bounded delays for traffic that conforms to its resource reservation. In essence, Paternoster applies the concept of Credit-Based scheduling (CBS) [13] with the cyclic attribute of CQF.

Paternoster is considered an enhancement to CQF since it guarantees TSN QoS whilst removing the time synchroniza-

tion requirement, i.e., an asynchronous scheduling protocol (though frequency synchronization is still needed to keep the cycle/epoch duration the same at all switches). While the results are favorable towards Paternoster in terms of minimizing mean and max delay, CQF is generally more predictable and therefore more deterministic than Paternoster, particularly for OT applications with critical QoS and hard deadline requirements. In particular, for periodic traffic and using the CQF protocol (with 50% gating ratio for ST, or 25μs transmission opportunity), all the streams with $\pi \leq 16$ have mean/max/min delays between 150μs and 200μs regardless of the path and cross traffic. This bounded delay guarantee is not easily predictable for Paternoster due to the fundamental loss of time synchronization between switches. Additionally, since Paternoster uses the strict priority scheduling at the egress, it contains elements of best-effort service which causes loss of determinism.

CQF provides complete isolation between ST and BE (due to TAS being used at the egress port) and as a result performs fairer in resource allocation between ST And BE, especially at high traffic intensity. Paternoster does not isolate traffic classes (though it does provide resource allocation) which can degrade the predictability and deterministic behavior for ST. This effect is evident in the periodic jitter results where the mean jitter at varying traffic intensities for CQF is monotonically increasing (up to a bounded jitter value), while the jitter measurement for Paternoster is highly erratic.

In terms of packet loss, both CQF and Paternoster guarantee zero loss for streams that conform to their reservations. However, Paternoster does perform slightly worse when packets remain in the *prior* queue and a cycle change occurs causing the *prior* queue to be purged of its content. This rarely happens with periodic traffic sources, but can occur more frequently with sporadic traffic sources where a switch can abruptly receive a large number of traffic before a cycle change.

Since Paternoster uses strict priority scheduling at the egress, Paternoster achieves significantly better delays due to having more transmission opportunities than the CQF protocol, i.e., CQF's use of TAS at the egress divides and isolates the transmission opportunities and does not adapt these opportunities to varying changes in traffic intensity. Moreover, the main issue with CQF in guaranteeing QoS for sporadic ST streams is that the burst usually is much greater than the allowable bandwidth per cycle (the transmission opportunities given). Applying ingress policing and admission control (either centralized or distributed) can mitigate this issue by using control signals and negotiating network resources and QoS to streams that request it (this has been investigated in [14] switches utilizing TAS only).

IV. Conclusions and Future Work

A performance evaluation has been conducted in this report to compare standard CQF and Paternoster. Since Paternoster uses more queues, i.e., more complexity, and provides less deterministic behavior compared to CQF, CQF performs better for OT applications with hard real-time requirements. While Paternoster performs worse than CQF in ensuring deterministic properties, it provides a relaxed traffic predictability in networks that do not have time synchronization. While this performance evaluation used statically defined traffic slots in the cycle (for CQF/TAS), an adaptive method (Adaptive TAS [15]) can be used to accurately determine the needed slot duration to optimally service registered traffic classes (in this case, BE and ST). Using TAS in Paternoster at the egress port between ST and BE instead of strict priority scheduling is another recommendation that can reduce the jitter for streams. While the tests in this evaluation involved uniform link transmission and propagation delays, a more complex problem that involves different link transmission and propagation delays is an interesting direction for future research since it can cause cycle misalignment between adjacent ports according to the standard.

In the wider context of QoS networking and related applications, QoS oriented routing approaches, e.g. [16], [17] should be investigated. Furthermore, deterministic networking should be studied in the context of emerging multiple-access edge computing (MEC) [18]–[22], in particular MEC settings for low-latency applications [23]–[25]. As an alternative approach to coordinating the reconfigurations, emerging softwarized control paradigms, such as software defined networking can be explored [26]–[30]. Regarding the reliability aspects, a potential future research direction is to explore low-latency network coding mechanisms, e.g., [31]–[37], to enhance networking protocols targeting reliable low-latency communication.

References

[1] S. Thangamuthu, N. Concer, P. J. L. Cuijpers, and J. J. Lukkien, "Analysis of ethernet-switch traffic shapers for in-vehicle networking applications," in *Proc. IEEE Design, Automation Test in Europe Conf. Exhibition*, Mar. 2015, pp. 55–60.

[2] D. Thiele, R. Ernst, and J. Diemer, "Formal worst-case timing analysis of ethernet TSN's time-aware and peristaltic shapers," in *Proc. IEEE Vehicular Networking Conference (VNC)*, 2015, pp. 251–258.

[3] Z. Zhou, M. S. Berger, S. R. Ruepp, and Y. Yan, "Insight into the IEEE 802.1Qcr asynchronous traffic shaping in time sensitive network," *Advances in Science, Technology and Engineering Systems Journal*, vol. 4, no. 1, pp. 292–301, 2019.

[4] Z. Zhou, Y. Yan, M. Berger, and S. Ruepp, "Analysis and modeling of asynchronous traffic shaping in time sensitive networks," in *Proc. IEEE Int. Workshop on Factory Commun. Systems (WFCS)*, Jun. 2018, pp. 1–4.

[5] A. A. Atallah, G. Bany Hamad, and O. Ait Mohamed, "Multipath routing of mixed-critical traffic in time sensitive networks," in *Advances and Trends in Artificial Intelligence. From Theory to Practice*, F. Wotawa, G. Friedrich, I. Pill, R. Koitz-Hristov, and M. Ali, Eds. Cham: Springer International Publishing, 2019, pp. 504–515.

[6] M. Pahlevan, N. Tabassam, and R. Obermaisser, "Heuristic list scheduler for time triggered traffic in time sensitive networks," *ACM Sigbed Review*, vol. 16, no. 1, pp. 15–20, 2019.

[7] F. Heilmann and G. Fohler, "Size-based queuing: an approach to improve bandwidth utilization in TSN networks," *ACM SIGBED Review*, vol. 16, no. 1, pp. 9–14, 2019.

[8] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 88–145, 2019.

[9] "IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks–Amendment 29: Cyclic Queuing and Forwarding," *IEEE 802.1Qch-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd(TM)-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, IEEE Std 802.1Qbz-2016, and IEEE Std 802.1Qci-2017)*, pp. 1–30, Jun. 2017.

[10] M. Seaman, "Paternoster policing and scheduling, Revision 2.1," May 2019, available from http://www.ieee802.org/1/files/public/docs2019/cr-seaman-paternoster-policing-scheduling-0519-v04.pdf, Last accessed May 25, 2019.

[11] N. Finn, J.-Y. L. Boudec, E. Mohammadpour, J. Zhang, B. Varga, and J. Farkas, "DetNet Bounded Latency," Internet Engineering Task Force, Internet-Draft draft-finn-detnet-bounded-latency-03, Mar. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-finn-detnet-bounded-latency-03

[12] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proc. ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, 2008, pp. 1–10.

[13] "IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks," *IEEE Std 802.1Q-2014 (Revision of IEEE Std 802.1Q-2011)*, pp. 1–1832, Dec. 2014.

[14] A. Nasrallah, V. Balasubramanian, A. S. Thyagaturu, M. Reisslein, and H. Elbakoury, "Reconfiguration algorithms for high precision communications in time sensitive networks: Time-aware shaper configuration with IEEE 802.1qcc (extended version)," *CoRR*, vol. abs/1906.11596, 2019. [Online]. Available: http://arxiv.org/abs/1906.11596

[15] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Performance comparison of IEEE 802.1 TSN Time Aware Shaper (TAS) and Asynchronous Traffic Shaper (ATS)," *IEEE Access*, vol. 7, pp. 44 165–44 181, 2019.

[16] U. Chunduri, A. Clemm, and R. Li, "Preferred Path Routing - a next-generation routing framework beyond Segment Routing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec 2018, pp. 1–7.

[17] J. W. Guck, A. Van Bemten, M. Reisslein, and W. Kellerer, "Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 388–415, First Qu. 2018.

[18] T. Doan-Van, A. Kropp, G. T. Nguyen, H. Salah, and F. H. Fitzek, "Programmable first: Automated orchestration between MEC and NFV platforms," in *Proc. IEEE Consumer Commun. & Netw. Conf. (CCNC)*, 2019, pp. 1–2.

[19] Y. Gao, W. Tang, M. Wu, P. Yang, and L. Dan, "Dynamic social-aware computation offloading for low-latency communications in iot," *IEEE Internet of Things Journal, in print*, 2019.

[20] J. Martín-Pérez, L. Cominardi, C. J. Bernardos, A. De la Oliva, and A. Azcorra, "Modeling mobile edge computing deployments for low latency multimedia services," *IEEE Transactions on Broadcasting, in print*, 2019.

[21] P. Shantharama, A. S. Thyagaturu, N. Karakoc, L. Ferrari, M. Reisslein, and A. Scaglione, "LayBack: SDN management of multi-access edge computing (MEC) for network access services and radio resource sharing," *IEEE Access*, vol. 6, pp. 57 545–57 561, 2018.

[22] M. Wang, N. Karakoc, L. Ferrari, P. Shantharama, A. S. Thyagaturu, M. Reisslein, and A. Scaglione, "A multi-layer multi-timescale network utility maximization framework for the SDN-based layback architecture enabling wireless backhaul resource sharing," *Electronics*, vol. 8, no. 9, pp. 937.1–937.28, 2019.

[23] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward low-latency and ultra-reliable virtual reality," *IEEE Network*, vol. 32, no. 2, pp. 78–84, 2018.

[24] Z. Xiang, F. Gabriel, E. Urbano, G. T. Nguyen, M. Reisslein, and F. H. Fitzek, "Reducing latency in virtual machines: Enabling tactile internet for human-machine co-working," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1098–1116, 2019.

[25] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Mobile edge computing and networking for green and low-latency Internet of Things," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 39–45, 2018.

[26] R. Amin, M. Reisslein, and N. Shah, "Hybrid SDN networks: A survey of existing approaches," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3259–3306, 2018.

[27] N. Deric, A. Varasteh, A. Basta, A. Blenk, R. Pries, M. Jarschel, and W. Kellerer, "Coupling VNF orchestration and SDN virtual network reconfiguration," in *Proc. Int. Conf. on Networked Systems (NetSys)*, 2019.

[28] A. Destounis, S. Paris, L. Maggi, G. S. Paschos, and J. Leguay, "Minimum cost SDN routing with reconfiguration frequency constraints," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1577–1590, 2018.

[29] W. Kellerer, P. Kalmbach, A. Blenk, A. Basta, M. Reisslein, and S. Schmid, "Adaptable and data-driven softwarized networks: Review, opportunities, and challenges," *Proceedings of the IEEE*, vol. 107, no. 4, pp. 711–731, April 2019.

[30] H. Sándor, B. Genge, and Z. Szántó, "Infrastructure and framework for response and reconfiguration in Industry 4.0," in *Proc. IEEE Int. Symp. on Digital Forensic and Security (ISDFS)*, 2018, pp. 1–6.

[31] J. Acevedo, R. Scheffel, S. Wunderlich, M. Hasler, S. Pandi, J. Cabrera, F. Fitzek, G. Fettweis, and M. Reisslein, "Hardware acceleration for RLNC: A case study based on the xtensa processor with the tensilica instruction-set extension," *Electronics*, vol. 7, no. 9, p. 180, 2018.

[32] A. Cohen, D. Malak, V. B. Bracha, and M. Medard, "Adaptive causal network coding with feedback for delay and throughput guarantees," *arXiv preprint arXiv:1905.02870*, 2019.

[33] A. Engelmann, W. Bziuk, A. Jukan, and M. Médard, "Exploiting parallelism with random linear network coding in high-speed Ethernet systems," *IEEE/ACM Transactions on Networking (TON)*, vol. 26, no. 6, pp. 2829–2842, 2018.

[34] F. Gabriel, S. Wunderlich, S. Pandi, F. H. Fitzek, and M. Reisslein, "Caterpillar RLNC with feedback (CRLNC-FB): Reducing delay in selective repeat ARQ through coding," *IEEE Access*, vol. 6, pp. 44 787–44 802, 2018.

[35] D. E. Lucani, M. V. Pedersen, D. Ruano, C. W. Sørensen, F. H. Fitzek, J. Heide, O. Geil, V. Nguyen, and M. Reisslein, "Fulcrum: Flexible network coding for heterogeneous devices," *IEEE Access*, vol. 6, pp. 77 890–77 910, 2018.

[36] Z. Ma, M. Xiao, Y. Xiao, Z. Pang, H. V. Poor, and B. Vucetic, "High-reliability and low-latency wireless communication for internet of things: Challenges, fundamentals and enabling technologies," *IEEE Internet of Things Journal, in print*, 2019.

[37] S. Wunderlich, F. Gabriel, S. Pandi, F. H. Fitzek, and M. Reisslein, "Caterpillar RLNC (CRLNC): A practical finite sliding window RLNC approach," *IEEE Access*, vol. 5, pp. 20 183–20 197, 2017.