Journal of Network and Systems Management (2022) 30:71
https://doi.org/10.1007/s10922-022-09683-3

High Capacity and Resilient Large-Scale Deterministic IP Networks

Vincent Angilella[1] · Filip Krasniqi[1] · Paolo Medagliani[1] · Sebastien Martin[1] · Jérémie Leguay[1] · Ren Shoushou[1] · Liu Xuan[1]

Received: 1 March 2022 / Revised: 27 July 2022 / Accepted: 1 August 2022

Abstract
With 5G networking, deterministic guarantees are emerging as a key enabler. In this context, we present a scalable architecture for Large-scale Deterministic IP Networks (LDN) that meets end-to-end latency and jitter bounds. This work extends the original LDN (Liu et al. in IFIP Networking, 2021) architecture, where flows are shaped at ingress gateways and scheduled for transmission at each link using an asynchronous and cyclic opening of gate-controlled queues. To further optimize the utilization of bandwidth and accept more traffic, our Advanced-LDN (A-LDN) architecture introduces a new shaping mechanism based on transmission patterns. To protect flows against failures, we also implement Frame Replication and Elimination for Reliability (FRER). To do so, we leverage on the A-LDN mechanism for traffic scheduling and activate it at core nodes, i.e., adding the possibility for additional shifts inside core nodes, so that replicas can arrive at destination at the same time. In any case, the network remains stateless (no per-flow states at core nodes), ensuring scalability over large-scale networks. For the control plane, we present variants of a column generation algorithm to quickly take admission control decisions and maximize traffic acceptance. For a set of flows, it determines acceptance and selects the best shaping parameters, routing policy, transmission patterns, and scheduling. Through numerical results and packet level simulations in OMNeT++, we demonstrate that our A-LDN architecture with transmission patterns improves traffic acceptance by up to 67% compared to the original LDN architecture. We also demonstrate that FRER can be efficiently supported at the cost of some extra bandwidth utilization.

Keywords Deterministic networks · Large-scale IP networks · Bounded delay · Bounded jitter

✉ Paolo Medagliani
paolo.medagliani@huawei.com

[1] Huawei Technologies, Boulogne-Billancourt, France

Published online: 23 August 2022

1 Introduction

Latency-guaranteed networking is becoming a must to enable a wide-range of Internet applications like factory automation, connected vehicles, and smart grids [2]. To frame the development of new technologies for deterministic IP networks, the IETF [3] and the ITU-T [4] have specified target requirements for guaranteed bandwidth, bounded End-to-End (E2E) latency, and bounded jitter. In the past decade, a collection of IEEE 802.1 Ethernet standards, known as Time-Sensitive Networking (TSN) [5], has been developed to support professional applications over Local Area Networks (LAN) with layer-2 mechanisms such as priority queuing, preemption, traffic shaping, and time-based opening of gates at output ports. To improve the scalability of TSN solutions, the IETF DetNet (Deterministic Networking) [6] group has been working on the Large-scale Deterministic Network (LDN) [7] architecture that specifies how traffic should be scheduled and forwarded at large-scale in IP networks.

To provide E2E latency and bounded jitter guarantees at large-scale, several challenges must be addressed. The elasticity or burstiness of traffic needs to be controlled, and low complexity solutions are required to avoid core routers to maintain per-flow states. To address the aforementioned challenges, we presented in [1] a comprehensive LDN architecture, that extends current work at IETF [7], to guarantee deterministic E2E latency and bounded jitter for high-priority flows. In this original LDN architecture, flows are shaped at ingress gateways and scheduled for transmission at every hop using a cyclic opening of gate-controlled queues, i.e., IEEE 802.1Qch or Cyclic Queuing Forwarding (CQF), to ensure hop-by-hop guarantees. At the data plane, scalability is achieved thanks to an asynchronous and cyclic opening of 3 transmission queues and a forwarding of packets, exploiting a hop-by-hop permutation of a header label to identify the transmission queue at the next hop for each flow. Forwarding operations at intermediate nodes are of low complexity and totally stateless. At the control plane, an algorithm [1] based on column generation has been proposed to decide about acceptance, ingress shaping and routing for each flow. To ensure isolation between flows and guarantee end-to-end performance bounds, a portion of each transmission cycle is *reserved* for each flow at every hop. This reservation is virtual and remains inside the bandwidth allocation performed by the control plane algorithm. It does not rely on any particular per-flow states in the data plane, making the solution more scalable for large networks. The main drawbacks of LDN are that (i) it can waste bandwidth reserved on the links, especially for flows with low rate, and (ii) no reliability mechanisms are available.

To further optimize the utilization of the network capacity and accept more traffic, we introduce in this paper the Advanced LDN (A-LDN) architecture that leverages transmission patterns for the injection of flows into transmission cycles at ingress gateways. To allocate resources at a finer granularity, A-LDN uses CQF with more than 3 queues at every node (e.g., 32 queues in this paper) and it supports non-uniform *reservations* over transmission cycles, i.e., transmission patterns over a hypercycle (periodical time interval that can be expressed as a

multiple of the basic cycle). It keeps the fundamental design principle of LDN for scalability with flow-level operations at ingress gateways for the injection of traffic into reserved cycles and stateless forwarding over input and output cycles at core nodes thanks to a quasi-static mapping table. For the control plane, we present a revised column generation algorithm to quickly take admission control decisions in large-scale networks. For every flow, it determines acceptance and selects the best shaping parameters (i.e., maximum burst size and transmission pattern) and routing policies (i.e., paths) to maximize the total accepted throughput.

Thanks to the finer scheduling that A-LDN allows, we extended this solution with the support of FRER (Frame Replication and Elimination for Reliability), also referred to as IEEE 802.1CB [8], that duplicates packets over two disjoint paths to protect flows against failures. In this context, we re-introduce the possibility for scheduling at core nodes to guarantee that arrival times of packet copies are the same at destination. We show that at the cost of this extra complexity, flows can be routed over totally disjoint paths (1+1 protection) or on pairs of paths that meet a reliability target (i.e., a given loss probability). We present how the control plane algorithm, based on column generation, can operate with a *pathbook* containing resilient pairs of paths to provide an (almost) optimal solution in a reasonable amount of time. The use of a pathbook may either allow to provide more scalable solutions or meet path requirements coming from other layers, such as the optical one.

Though numerical results and packet level simulations in OMNeT++ [9], we demonstrate that A-LDN improves traffic acceptance by up to 67% compared to the original LDN, while keeping the same properties of determinism for accepted demands, thanks to a finer allocation of resources. Additionally, through extensive tests on multiple topologies, we demonstrate that FRER allows to satisfy reliability requirements, at the cost of a higher bandwidth consumption introduced by backup paths.

The rest of this paper is organized as follows. In Sect. 2, we present some related works. Sect. 3 recalls the main features of the LDN architecture presented in [1]. Section 4 introduces A-LDN and Sect. 5 presents the extension for FRER. Section 6 provides the control plane algorithm and its variants. Finally, Sect. 7 describes performance results and Sect. 8 concludes the paper.

2 Related Work

The IEEE 802.1 TSN task group has produced a series of standards to provide deterministic transmissions, and to guarantee flow integrity. IEEE 802.1Qch specifies the CQF method based on 2 queues to synchronize transmissions and bound the latency at each hop. Time-sensitive traffic is transmitted and queued for transmission along a network path in a cyclic manner. In this context, scheduling methods have been proposed [10] to find a proper schedule of sending/injection time for time-sensitive flows. However, CQF requires time synchronization and can only be used over short-distance links. IEEE 802.1Qcr, known as ATS (Asynchronous Traffic Shaper), describes how to handle latency without requiring

time synchronization between devices. However, it uses an inter-leaved shaping method to shape every flow at every hop. This method lacks of scalability as it requires all core routers to maintain per-flow states to be updated for each packet. Asynchronous Time-Aware Shaping (ATAS) [11] presents a distributed approach to control gates such that low priority streams are not interfering with high-priority ones. This approach, based on stream prediction, despite it does not require synchronization between nodes, lacks of scalability. Under the TSN architecture, IEEE 802.1CB [12] describes a standardized stand-alone module named Frame Replication and Elimination for Reliability (FRER), that proposes to duplicate frames and to send them over multiple paths to protect flows against failures.

To integrate TSN technologies into large-scale IP networks, the IETF DetNet working group has defined a general Deterministic Networking (DN) architecture [6]. More concretely, Large-scale DN (LDN) [7] and Cycle Specified Queuing and Forwarding (CSQF) [13] drafts have been proposed for the implementation of deterministic data scheduling and forwarding. They both rely on the use of CQF with more than 2 transmission queues in order to relax tight time-synchronization constraints and to schedule, in a more flexible way, transmissions at each hop. The main differences are the following. In CSQF, (i) packet arrivals are required to be known in advance (i.e., time-triggered traffic) and (ii) a Segment Routing (SR) [14] or Cycle Tag [15] header is used to route traffic (i.e., to define the next hop) and schedule packets (i.e., to select the outgoing queue) at each node, leading to a significant packet overhead. The information carried out in the header is mapped to the corresponding outgoing port and queue. Several control plane algorithms for joint routing and scheduling with CSQF have been proposed [14, 15]. On the other hand, LDN introduces a more scalable hierarchical architecture with (i) the possibility to control shaping parameters for each flow at ingress nodes (it can handle more elastic traffic sources and only controls the *arrival curve* of flows) and a (ii) cycle-based deterministic forwarding at core nodes in which no per-flow states are needed. Compared to the LDN draft [7], our original paper on LDN [1] details how we efficiently implemented shaping and routing at data plane and control plane levels. However, in LDN the bandwidth for a flow must be reserved over all the cycles, as there exists no mechanism for finer bandwidth reservation. Our work on the A-LDN architecture adds the possibility to select advanced transmission patterns to further improve bandwidth utilization and traffic scheduling possibilities, allowing to optionally add a Frame Replication and Elimination for Reliability (FRER) module to handle critical flows that require reliability guarantees.

In [16], the authors proposed a new metric for path computation when paths cannot be totally-disjoint. The goal is to avoid unintentional elimination of packets. In [17] authors study the resilient path computation problem where the goal is to find a set of paths with minimum cost from the source to the destination, such that the probability that all paths fail simultaneously is lower than a given threshold. Several heuristics have been proposed for SRLG (Shared Risk Link Group) disjoint path computation [18, 19]. However, the implementation of FRER on top of A-LDN needs to consider the temporal aspect so that the difference of arrival times of replicas is bounded.

The OMNeT++ network simulator [9] has been already used for a number of TSN/DetNet studies [1, 20–22] using the CQF scheduler. However, to our knowledge, we are the first to implement the LDN architecture proposed at IETF, with an optional FRER module to meet target reliability requirements.

3 Background: A Reminder of LDN Features

In this section, we provide an overview of the LDN architecture implemented in our previous work [1]. We will explain in Sect. 4 its extension to improve throughput and add frame replication. As a reminder, LDN is based on the asynchronous cyclic opening of 3 Gate-Controlled Queues (GCQs) in order to schedule the transmission of data packets. A new queue is open every T second, while the previous opened queue is closed for transmission to avoid interference. The central controller decides about the shaping of incoming flows to enforce, using network calculus terminology [23], an arrival curve $A_f(t) = b_f + r_f \cdot t$, with r_f and b_f the arrival rate and the maximum burst size for each flow f. To do so, the ingress Gateway (iGW) shapes the incoming bursts and injects packets in cycles, according to a *reserved* capacity that depends on r_f and the end-to-end (E2E) delay requirement D_f^{max} of each flow f. In such a way, it is possible to ensure a bounded end-to-end latency and a very limited (almost 0) jitter between the ingress and the egress gateways.

The main features of LDN are the following.

Traffic Shaping Each flow is declared to the network controller using an arrival rate r and a maximum burst size b. These two parameters are used to characterize the original arrival curve $A(t) = b + r \cdot t$ that indicates the maximum amount of traffic that can be sent over a period of time t.

As bandwidth is reserved in cycles for the accepted flows, they must be shaped at the iGW to let the arrival curve fit with the guaranteed service curve $S(t)$.

The shaping operation introduces a shaping delay $D_{shaping} = T\lceil\frac{b}{b'}\rceil$, where b is the incoming maximum burst size, b' the guaranteed one, and T is the cycle duration. The shaping parameter b' is adjusted by the controller in order to maximize flow acceptance while respecting end-to-end delay requirements of flows D^{max}.

Bandwidth Reservation Thanks to the definition of the shaping parameter b' and the uniform reservation of the capacity in each cycle, the iGW reserves for each flow f an amount of bandwidth equal to $\frac{b'_f}{T}$ for each cycle at every hop. As we can observe, some flows may undergo an excessively large amount of capacity waste due to this uniform reservation scheme (especially for small rate and/or bursty flows, i.e., $r << \frac{b'_f}{T}$). This is what motivated the advanced transmission patterns proposed in Sect. 4 for A-LDN.

Cyclic Mapping The delay is kept bounded, as there is a strict mapping between the sending cycle at the parent node and the outgoing cycle at the child node (i.e., the cycle X in Node A and the cycle Y in Node B in Fig. 1). Similarly to CSQF, each node is using 3 queues at each outgoing port (i.e., cycle Y, Y+1, and Y+2 in Fig. 1) and the strict mapping between neighbor nodes compensates variations in processing and propagation delay, relaxing the need for strict

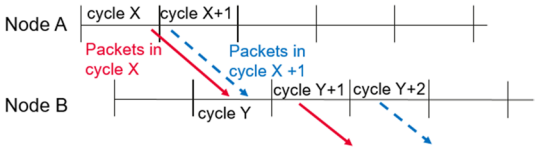


Fig. 1 Cycle mapping between adjacent nodes. The packets in cycle X of node A will be transmitted in cycle Y+1 of node B, while those transmitted in cycle X+1 of node A will be mapped into cycle Y+2 of node B, as the reception of these packets is done during cycles Y and Y+1

time synchronization between neighbor devices. Only the cycle duration must be guaranteed everywhere in the network. In this way, only the transmission at the first and last nodes will have an impact on the jitter. Once learned, the mapping between neighbor nodes is static. As this mapping only depends on neighboring nodes and cycles, it allows for a low complexity forwarding at core nodes. This mechanism is stateless as the core nodes do not store any per-flow states to implement the mapping, making it feasible for large-scale networks.

Deterministic Guarantees The worst case E2E delay can be calculated and used to enforce the E2E latency constraints of flows. It includes shaping, propagation, and forwarding/queuing delay.

An upper bound of the end-to-end delay for flow f can be computed as:

$$D_f^{\text{E2E}} = D_f^{\text{shaping}} + D_f^{\text{forwarding}} + J_{\text{E2E}}$$

D_f^{shaping} refers to the previously described shaping delay, $D_f^{\text{forwarding}}$ is the minimum amount of time between the transmission time at iGW and the reception time at the egress gateway (eGW), and can be computed as $D_f^{\text{forwarding}} = \sum_{e \in p_f} \delta(e)$, with p_f the path for flow f, expressed as a list of edges e, and $\delta(e)$ the shift between the cycles, expressed in number of cycles of duration T, between the parent and child node for edge e. Referring to Fig. 1, the shift corresponds to the time interval between the transmission in cycle X at Node A and the forwarding in cycle Y in Node B. $\delta(e)$ takes into account, the processing delay of the node, the queuing delay (T), and the propagation delay between the nodes. As the shaping delay is fixed, and these terms are kept constant by design, the E2E performance is deterministic. We emphasize that D_f^{shaping} strictly depends on how shaping is performed and on the shaping factors used for a flow, while $D_f^{\text{forwarding}}$ strictly depends on the chosen routing path. The last term, J_{E2E}, represents the worst case jitter.

The jitter is measured as the maximum difference between the time a packet enters the LDN and the time it exits the LDN. Thanks to the mapping of input and output cycles at each hop, this maximum difference is bounded by $2T$, since it can only vary within the first and last hop. For more details on the mathematical derivation, the interested reader can refer to [1]. In order to accept a flow, we must enforce that $D_f^{\text{E2E}} \leq D_f^{max}$ for each accepted flow.

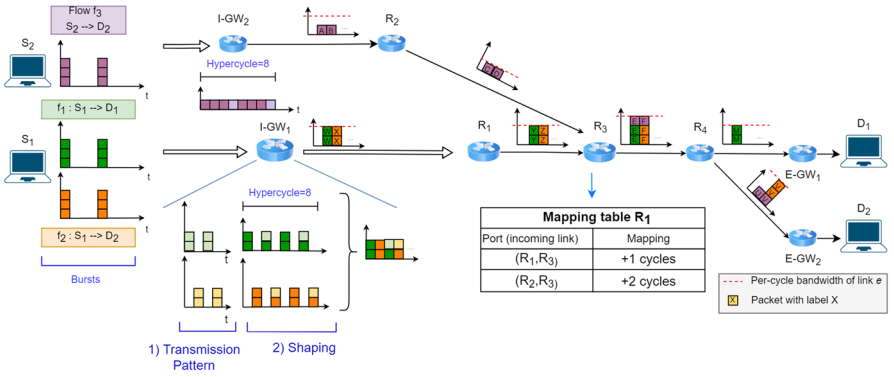


Fig. 2 Main features of A-LDN: injection of packet bursts into transmission patterns and shaping at iGW, and forwarding based on cycle mappings at core nodes. Three flows are shown: f_1, f_2 *and* f_3, having $b_1 = b_2 = b_3 = 3$ packets, and $r_1 = 1$ packets/cycle, $r_2 = r_3 = 0.75$ packets/cycle, being b_i the max burst size for the flow f_i expressed in packets, and r_i its arrival rate. First, iGW_1 injects packets into the pattern with a reservation of one burst every two cycles; then, the bursts are shaped with a resulting burst size of 2 packets per cycle for both f_1 and f_2, with a final combined pattern of two packets per cycle, transmittable through all the links of the paths. Similarly, the flow f_3 can transmit at most 1 packet in each reserved cycle. The transmission patterns selected for the three flows at the iGWs lead to an efficient use of the capacity at core links

4 Advanced LDN: Data Plane

In this section, we introduce the A-LDN architecture as an improvement of LDN [1]. We summarize its features with Fig. 2. A-LDN introduces a more advanced (virtual) reservation scheme that allows improving the performance of the network in terms of throughput, while satisfying end-to-end SLA requirements. This extension requires updates on both data plane and control plane. As for LDN, A-LDN does not rely on a reservation protocol throughout the network. The control plane virtually reserve bandwidth when deciding about shaping and routing parameters, i.e., no physical reservation is carried out in the nodes. In this way, by following the routing and transmission patterns provided by the controller, each accepted flow does not interfere with the other flows, ensuring that deterministic performance can be provided for each transmitted data packet.

The main novelty introduced by A-LDN, compared to LDN, is the introduction of *transmission patterns*, that required the adaptation of the shaping mechanism presented for LDN. A transmission pattern is a specific profile of resource reservation over different cycles, corresponding to a precise shaping of incoming traffic. We point out that, given a transmission patterns, its replicas over different starting cycles are considered as different transmission patterns, i.e., scheduled versions of the original ones.

4.1 Transmission Patterns

Thanks to the possibility of dispatching packets into the different GCQs at the iGWs, it is possible to reserve a different amount of bandwidth resources at each

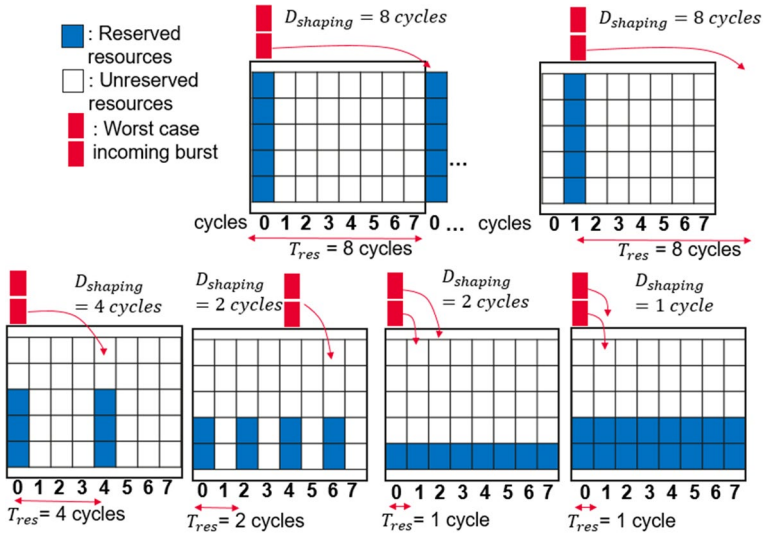


Fig. 3 Examples of transmission patterns over a hypercycle of $HC = 8$ cycles. Illustration of the shaping delay D_f^{shaping} for the regular and singular patterns considered. The two patterns at the top are considered to be different patterns, even if they are shifted and provide the same shaping delay and service rate. In LDN, only the two patterns on the bottom right were possible

cycle for each flow. As shown by Fig. 2, the incoming bursts are injected into transmission patterns selected by the controller, in order to efficiently combine the flows and maximize traffic acceptance in the network. For instance, Flow f_3 is mapped into a transmission pattern of 1 packet at every cycle, while f_1 and f_2 are mapped into a transmission pattern of 3 packets every 2 cycles.

To keep the implementation as simple as possible inside devices and the controller, we define a periodicity, referred to as *hypercycle* of length HC cycles, over which the transmission patterns repeat. As a consequence, a flow is no longer characterized by a single parameter b'_f, but rather by a set of HC values of the form: $\{b_f^1, b_f^2, ..., b_f^{HC}\}$; the maximum burst in cycle t_c for a flow f will be b_f^τ with $\tau = t_c \mod HC$. To ensure that a node can always free the transmitting queue at every cycle, the length of the hypercycle is smaller than the number of available GCQs at the iGWs. In addition, we consider a subset of all the potential transmission patterns with the following characteristics, as shown in Fig. 3:

- Regularity: non-zero values are separated with a constant pattern period T_{res}, i.e., $HC \mod T_{res} = 0$;
- Singularity: a single non-zero value is reserved and used at all cycles.

Simple and regular patterns can be described with $\{b'_f, T_{res}\}$, being, the b'_f unique non-zero value and T_{res} the constant pattern period.

Once selected at the iGW, the transmission pattern cannot be changed by core nodes. It can only be shifted by mappings (by $\delta(e)$) or scheduling at core nodes for

FRER. We remind that there is no specific flow-level reservation within core devices (e.g., via the RSVP protocol). The transmission pattern is used by the iGW to fit the incoming burst into the right queues and by the controller to make decisions about routing and flow acceptance for high-priority flows.

As bandwidth is now allocated at cycle granularity, it is possible to decide in which cycles of the hypercycle an incoming burst can be injected, so that it can be better combined with other incoming flows both at the iGW and at the core nodes. We point out that the same reservation profile over two different cycles of the hypercycle, as for the two patterns on the top of Fig. 3, are considered as two different reservation patterns (at most $HC - 1$ replicas are possible, one for each shift). In this work, we assume that the number of possible patterns is limited. This assumption makes sense as it reflects hardware limitations on possible shaping and scheduling configurations due, for instance, to processing speed or available memory.

4.2 Shaping

As for LDN, traffic comes in bursts, whose size may exceed the reserved resources over a cycle. The iGW maps packets of each burst in the selected transmission queues. If some packets cannot be fit in the immediately available resources, they must be reported to the next reserved cycles. This means that the iGW has, for each flow f, a service rate equal to b'_f/T_{res} that, over the period HC, should be larger than the arrival rate r_f to ensure deterministic performance and to guarantee the stability of the scheduler.

Differently from LDN [1], where the shaping delay only depends on the shaping factor applied to the flow, in A-LDN, this delay is impacted by the reservation pattern. As shown in Fig. 3, the shaping delay must take into account the worst-case scenario, i.e. when the burst arrives during a reserved cycle. In this case, referring to b'_f as the max allowed burst size or height of the reserved resource in a cycle of flow f, an incoming burst of size b_f must be split over $\lceil b_f/b'_f \rceil$ periods of the pattern. The shaping delay can then be expressed as $D_f^{\text{shaping}} = \lceil \frac{b_f}{b'_f} \rceil T_{\text{res}}$, where T_{res} is the pattern period (i.e., the distance between two blue blocks in Fig. 3). In order to guarantee the requested performance bounds for each flow, D_f^{shaping} must be accounted into the calculation of the E2E delay for each flow. We point out that D_f^{shaping} can be larger than the shaping delay of LDN, limiting potentially the available paths to reach the destination endpoint. We also point out that if a burst is larger than the maximum allowed burst size requested to the network, data packets can be either lost or transmitted over other cycles, introducing delay and jitter to the flow. Throughout this work, we will consider that each flow can be represented using an arrival curve.

4.3 Mapping Between Cycles

The mapping at core nodes between the output cycle of the parent node and the forwarding cycle of the child node is implemented in the same way in LDN and A-LDN (see Fig. 1). Packets have a 4-bits label to identify the cycle at which they

are transmitted by the parent node. This header is referred to as *Tagged CQF* in IETF draft [24]. Each forwarding router maintains an internal mapping table which specifies, for each parent node and label, which is the next transmission cycle to be used by this packet. We point out that we could consider the information on the outgoing port in the case of parallel links. For the sake of simplicity, here we neglect the port information for parallel links mapping. The label is updated at each hop according to the information stored in the mapping table at each hop, as shown in Fig. 2). In this way, LDN is able to maintain the transmission pattern along the path, and have a jitter smaller than $2T$. As this mechanism is stateless, i.e., it does not depend on the flows in the network, LDN and A-LDN can scale over large-scale networks.

The mapping only depends on the input port of a node and is characterized by a shift of $\delta(e)$ cycles applied on the transmission pattern, where e represents the link connected to the input port (see Fig. 2). The mapping table is filled at network startup, for instance via the transmission of PING messages between devices, and remains static as long as no network changes happen.

4.4 Use of a Pre-computed Set of Paths

In large telecommunication networks, the pre-computation of paths is a way to reduce the burden on the control plane algorithm during network planning or online traffic acceptance [25]. It can also be used to enforce good quality paths to maximize in-network bandwidth acceptance [26].

LDN and A-LDN do not escape this rule, as it may be of interest to leverage on a set of pre-computed paths for each demand. For instance, they can be related to optical constraints of the underlying physical network. Alternatively, the use of specific pre-computed paths can be required to enforce the use of specific network resources, for instance to traverse specific network functions such as monitoring probes or firewalls.

In general, pre-computed paths may come from underlying physical layers, such as available optical links, or from path computation routines in the network controller to enforce specific QoS requirements.

5 Frame Replication for Reliability

LDN and A-LDN provide both a solution for deterministic delay guarantees and almost 0 jitter. However, they are not reliable against failures in the network. Mechanisms such as IP Fast ReRoute [27] or Topology Independent Loop-free Alternate Fast Re-route (TI-LFA) [28] allow for a quick rerouting of flows in the case of failure. However, even if the reaction time is expected to be below 50ms, some packet losses can be experienced while the traffic is rerouted on a new path, making it not suitable for deterministic networking.

For such a reason, in this work, we will consider a protection mechanism such as Frame Replication and Elimination for Reliability (FRER) [8], sending one or more packet replicas over different paths. Throughout this paper, we will consider only

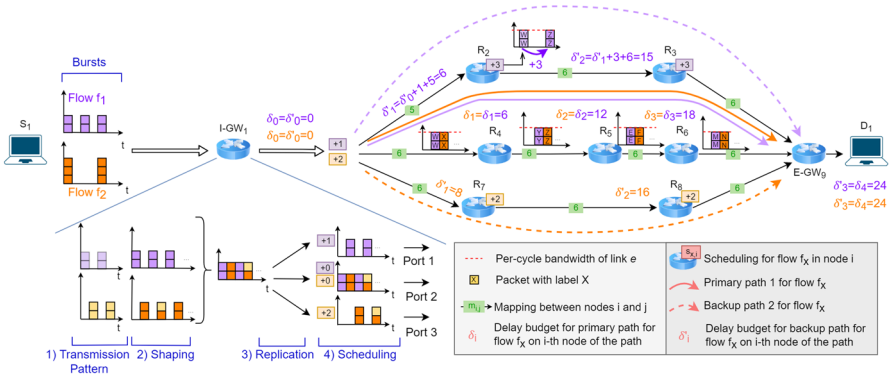


Fig. 4 Main features of FRER: 1) two incoming bursts of flow f_1 and f_2 are mapped into two different transmission patterns; 2) the incoming burst is shaped to fit into the assigned transmission pattern; 3) packets are replicated at the iGW to be sent over 2 different paths; 4) packets are scheduled for transmission over different ports. According to the path used by each replica, a different scheduling is added to the patterns to let the packets of a flow arrive at the same cycle at the destination

one replica of packets. According to the standard, the destination only keeps the first packet received, discarding all the other replicas. In particular, in the implementation for FRER, we require that the replicas of the packet are received at the same cycle at the eGW to avoid the use of dedicated buffers and guarantee 0 jitter even in the case of failures. Once received, packet duplicata are detected by checking the packet ID and the flow ID.

In order to implement FRER, as shown in Fig. 4, packets of a flow are first shaped and inserted in the transmission pattern selected by the network controller. Replicas are then created and inserted in the corresponding queues for transmission over different outgoing ports. As each replica follows a different path to reach the destination, and under the constraint that packets must arrive within the same cycle at the destination, an additional scheduling needs to be carried out at core nodes to shift transmission patterns, differently from A-LDN without FRER support. This extra complexity is required to get more scheduling flexibility and protect a larger set of flows. As mentioned in Sect. 5.2, the additional shifts at core nodes can be encoded into packet headers. In this way, combined with the fact that no buffer is required at the eGW to store packets, the network remains stateless, making the FRER solution feasible also for large-scale instances.

5.1 Transmission Patterns, Shaping, and Mapping

The injection of flows with shaping and transmission patterns for FRER is carried out in the same way as the shaping for A-LDN presented in Sect. 4.2. The iGW can spread packets of a burst over different cycles if they do not fit within the virtual resource reservation for a flow f carried out by the controller, as shown in Fig. 3. In order to keep the iGW simple, the same transmission pattern is applied for all the replicas of the FRER flows, in order to ensure that the replicas are received within the same cycle. Otherwise, with different transmission patterns, a packet could be

inserted in a given cycle depending on the arrival time, requiring much larger complexity to compensate this delay and guarantee 0 jitter. This means that all the replicas of the flow will experience, at least, a shaping delay $D_f^{\text{shaping}} = \lceil \frac{b_f}{b'_f} \rceil T_{\text{res}}$.

In order to maintain the core stateless, in FRER as well, we consider mapping between parents and children nodes. In the header of each packet, a tag is added by the transmitting node, in order to denote the cycle at which the packet has been sent. The information is consumed by the child node, which knows the static mapping between parents' transmission cycles and its own transmission cycle. Before forwarding the packet, the child node replaces the tag of the packet with its own tag indicating the cycle used for transmission.

5.2 Scheduling

The main difference between A-LDN and FRER lies in the way the packets are scheduled. In A-LDN without FRER, it is only possible to select different transmission patterns at the iGW (i.e., translated versions with the same traffic profile), in order to maximize the acceptance of flows. With FRER, instead, as packets must arrive at the same time at the destination, packets can be delayed at the core nodes (i.e., transmission patterns can be shifted), as indicated in Fig. 4. We point out that the delay within a node is rigidly applied to all the packets of a flow traversing the node. In this way, no additional jitter is introduced on the packets.

In order to keep the core of the network stateless, as it is for A-LDN, the information on the shift to be applied at core nodes is carried out by packets in the header, in the form of a tuple *(next hop, shift)*, necessary to identify in which hops the shift is applied, without storing this information inside the devices. For instance, this could be via an extension of Segment Routing (SR). The node, before forwarding the packet to the next hop, checks the shift to be applied and insert the packet in the corresponding queue for transmission. At this point, the node updates the tag inserted into the packet.

5.3 Path Selection for Reliability

Losses in the network can be related to different reasons, such as a node or a link failure, or temporary perturbations that can lead to packet drops. While mechanisms such as IP FRR introduce jitter and packet losses before reacting to a failure, FRER [8] guarantees, via the transmission of replicas of packets over multiple paths, that no packets are lost, and no jitter is introduced in the case of failure. We remind that, according to FRER, packets are duplicated at the iGW, sent over different paths towards the destination, under the constraint of being received within the same cycle by the eGW. This latter node only keeps the first received packet and discards all the other replicas received afterwards. We remind that packets can be received at the same cycle at the eGW as intermediary nodes shift the transmitted data accordingly.

The simplest version of FRER can be implemented using 1+1 protection, i.e., by finding a totally link-disjoint pair of paths from the source to the destination. An improvement of this model takes into account the probability of packet loss at

Table 1 Packet loss probability as a function of the number of hops for a constant $p_e^{\text{error}} = 10^{-7}$, and a target QoS of $P_f^{loss} \leq 10^{-6}$. We note that while the single path can guarantee such a reliability for a path up to 10 hops, FRER(assuming the two paths have the same number of hops) still guarantees the constraint for paths having a number of hops as high as 1000

	1 hop	10 hops	100 hops	1000 hops
Single path	10^{-7}	10^{-6}	10^{-5}	10^{-4}
FRER	10^{-13}	10^{-11}	10^{-9}	10^{-7}

each link e, referred to as p_e^{error}, in order to find a group of paths that respects an overall target probability.

In order to keep the mathematical model simple enough, we assume that the paths of the original packet and of the replica are totally disjoint. Assuming that only a backup path is computed for a demand, a packet is successfully received if both packets are received or either a packet is lost on the primary path and well received on the backup path or vice versa. By consequence, there is a packet loss if the packet is dropped on both primary and backup paths. Given a flow f and a path p_f for the original packet, a path p_f' for the replicated packet, we can compute the path success probability as $P_f^{success} = 1 - ((1 - P_{p_f}^{success}) \cdot (1 - P_{p_f'}^{success}))$, where $P_{p_f}^{success} = \prod_{e \in p_f} (1 - p_e^{\text{error}})$ [29].

To further motivate the need for FRER to guarantee a path reliability target, we provide here an example. We consider, for instance, a network having a $p_e^{\text{error}} = 10^{-7}$ that must guarantee, for each flow f, a target QoS of $P_f^{loss} \leq 10^{-6}$. Table 1 shows the packet loss probability for FRER as a function of the number of hops for a generic flow f. For the sake of comparison, we show in the same table the packet loss probability without FRER, i.e., with a single path. We remind that the probability of success for a single path is $P_{p_f}^{success} = \prod_{e \in p_f} (1 - p_e^{\text{error}})$, with the loss probability being $P_{p_f}^{error} = 1 - P_{p_f}^{success}$. Even though we are considering highly reliable links, we notice how a protection mechanism such as FRER is required for bigger networks characterized by a large number of hops.

5.4 Use of a Pre-computed Set of Paths

As mentioned in Sect. 4.4, a pathbook may be introduced before the routing is decided, due to operational constraints. In this configuration, the solution provided will use one of the paths (also referred to as path pairs) from the list available.

In order to guarantee the desired reliability for FRER, the paths can be totally link or node-disjoint, as normally done for 1+1 protection, or maximally link-disjoint if no totally link-disjoint paths can be found. Alternatively, the path pairs can be generated so that a threshold on the probability of error is met, as described in Sect. 5.3. In Sect. 7.1, we will detail how the pathbook is created to ensure disjointness and reliability.

6 Control Plane Algorithm

In this section, we present an efficient control plane algorithm, maximizing the total throughput of the accepted flows for the three different solutions presented in the previous sections, i.e. LDN, A-LDN, and FRER. The mathematical framework is general and can be used to solve the original LDN problem, support scheduling with transmission patterns at the iGW for A-LDN and ensure the target reliability with FRER.

First, we present a general Integer Linear Program (ILP) with an exponential number of variables, where each variable corresponds to a solution for a flow. In order to efficiently manage this problem, we solve a sub-problem, referred to as pricing [30]. In Sect. 6.2, we describe the pricing algorithms associated with the three variants of the considered problem, i.e. LDN, A-LDN, and FRER.

6.1 Notations

Let's assume an input network characterized by:

- a directed graph $G = (V, E)$ representing the network topology, with V the set of routers and E the set of links.
- a packet loss probability $p_e^{error} \in [0, 1]$ for each arc $e \in E$
- a set of transmission cycles $\mathcal{C}$ with a hypercycle length HC.
- a cycle bandwidth capacity $C_{e,c}$ for each link $e \in E$ and for each $c \in \mathcal{C}$. For LDN, only one cycle is considered, as the reservation is identical on all cycles.
- the mapping $\delta(v, v')$ between adjacent routers over the link $(v, v') \in E$ (see Fig. 1). Going from v to v' introduces a constant shift of $\delta(v, v')$ cycles, modulo HC, to the transmission patterns decided at iGWs.

We denote the set of flows $\mathcal{F}$. Each flow $f \in \mathcal{F}$ is characterized by:

- a source node $s_f \in V$ and a destination node $t_f \in V$.
- an arrival curve $A_f(t) = b_f + r_f \cdot t$, where r_f is the arrival rate and b_f the maximum burst size;
- a maximum E2E delay requirement D_f^{max} (expressed in number of cycles).
- a maximum packet loss probability P_f^{loss} (considered in FRER only).

In the following, we denote by $\mathcal{S}_f$ the set of all feasible solutions for the flow $f \in \mathcal{F}$. For each flow $f \in \mathcal{F}$ and each solution $s \in \mathcal{S}_f$, $b(s, e, c)$ is the bandwidth utilization of the solution s on cycle c over the link e. Remark that a solution is defined for each variant of the problem.

6.2 Admission Control Problems

The goal is to maximize the throughput, computed as the sum of the arrival rates r_f for the flows accepted into the network. The different variants are expressed as follows.

LDN Problem Maximize the throughput of the accepted flows, while respecting the end-to-end delay constraint for each flow, and the capacity constraint on each arc. For each accepted flow, a feasible path and a related shaping parameter are provided.

A-LDN Problem Maximize the throughput of the accepted flows, while respecting the end-to-end delay constraint for each flow, and the capacity constraint on each arc and each cycle of the hypercycle. For each accepted flow, a feasible path, a transmission pattern, and a related shaping parameter are provided.

FRER Problem Maximize the throughput of the accepted flows, while respecting the end-to-end delay constraint for each flow, the capacity constraint on each arc and each cycle of the hypercycle, and imposing that the delay experienced on both paths, accounting for scheduling, is the same. The pair of paths must be either totally disjoint (1+1) or respect a target reliability requirement. For each accepted flow, a feasible pair of paths, a transmission pattern, a scheduling at each node, and a related shaping parameter are provided.

6.3 Mathematical Model

In this section, we propose a mathematical model that allows solving LDN, A-LDN, and FRER admission control problems. First, we present the generic model, and then we show how this model can handle the three variants of the problem.

The decision variables are as follows:

- $x_s \in \{0, 1\}$: Equals 1 iff the solution s is selected for the flow f, for $f \in \mathcal{F}$, $s \in \mathcal{S}_f$.

The LDN routing problems can be cast as the following linear program:

$$\begin{aligned} \max \sum_{f \in \mathcal{F}} \sum_{s \in \mathcal{S}_f} r_f x_s & \\ \sum_{s \in \mathcal{S}_f} x_s \leq 1 \quad & \forall f \in \mathcal{F} \end{aligned} \tag{1}$$

$$\sum_{f \in \mathcal{F}} \sum_{s \in \mathcal{S}_f} b(s, e, c) x_s \leq C_{e,c} \quad \forall e \in E, \forall c \in \mathcal{C} \tag{2}$$

Inequalities (1) make sure that, for each flow, only one solution is chosen. Constraints (2) ensure that the bandwidth required by all the flows going through an arc at a given cycle does not exceed the available bandwidth. Remark that for LDN, as

reservations are identical on all cycles, there is only one capacity constraint (2) for each arc.

A classical method to tackle an exponential number of variables is the Column Generation (CG) [31]. The formulation is solved as a Restricted Master Problem (RMP) with a limited subset of variables, which are relaxed to be greater than or equal to 0, instead of $\{0, 1\}$. In an iterative fashion, we only add the variables, also referred to as columns, which can improve the objective function. If no column can be added, then the linear relaxation is found. The algorithm to determine if a column must be added is referred to as pricing. It consists in finding a violated inequality in the dual of the RMP. If the dual constraint is violated, adding this column can improve the objective function of the RMP. Let $\alpha_{e,c}$ be the dual variable associated with the Inequality (2) for each link $e \in E$ and each cycle $c \in C$. For each flow $f \in \mathcal{F}$, the dual variable associated with the Inequality (1) is denoted as β_f. The dual model is equivalent to the following formulation:

$$\min \sum_{f \in \mathcal{F}} \beta_f + \sum_{e \in E} \sum_{c \in \mathcal{C}} C_{e,c} \alpha_{e,c} \tag{3}$$

$$\text{s.t. } \beta_f + \sum_{e \in s} \sum_{c \in \mathcal{C}} b(s, e, c) \alpha_{e,c} \geq r_f \qquad \forall f \in \mathcal{F}, s \in \mathcal{S}_f \tag{4}$$

At each iteration, the RMP provides the values $\alpha^*_{e,c}$ (resp. β^*_f) for each dual variables $\alpha_{e,c}$ (reps. β_f). For each flow $f \in \mathcal{F}$, the pricing problem consists in finding a solution $\bar{s} \in \mathcal{S}_f$ such that $\beta^*_f + \sum_{e \in s} \sum_{c \in \mathcal{C}} b(s, e, c) \alpha^*_{e,c} < r_f$.

Remark that the definition of a solution is relative to the considered problem variant. For each variant, we formally explain the solution definition as well as the associated pricing sub-problem.

6.4 Pricing Algorithms

In this section, we describe how to solve the pricing problem associated with each variant presented in Sect. 6.1. Furthermore, for each variant, we consider the problems both with and without a pathbook in input.

6.4.1 LDN

6.4.1.1 Solution definition A solution for one flow f is defined by a path and a shaping parameter. Note that, for a given path, the best shaping parameter is the smallest one verifying the end-to-end delay constraint. The set of solutions $\mathcal{S}_f$ is then defined as the set of paths for which a feasible shaping parameter exists.

The set of feasible shaping parameters SH_f for a flow f is found by looking at the number of cycles on which a maximum size burst is spread. For a number of cycles n, the shaping parameter associated is $b' = \left\lceil \frac{b}{n} \right\rceil$. If it verifies $b' \cdot n \geq b + n \cdot r$, then it

can absorb the maximum burst size and arrival rate, and is added into the set. The associated shaping delay is $n = \left\lceil \frac{b}{b'} \right\rceil$.

6.4.1.2 Pricing without pathbook For a shaping parameter b', we look for a path p minimizing the sum of dual costs $\sum_{e \in p} \alpha_e$ and respecting the delay constraint $\left\lceil \frac{b}{b'} \right\rceil + \sum_{e \in p} \delta e \leq D_f^{max}$. The only arcs allowed are the ones with enough capacity, meaning $b' \leq C_e$. As the pricing can be formulated as a Constrained Shortest Path (CSP) problem, we use the LARAC algorithm (see [32]) to solve it. Then, the column with the largest violation of the dual constraint (4) is added, if any.

6.4.1.3 Pricing with pathbook If the set of paths is known, then the column with the largest violation of the dual constraint (4) is found by looking at the dual costs for each path and each shaping parameter.

6.4.2 A-LDN

6.4.2.1 Solution definition A solution for a flow f is defined by a path and a transmission pattern, chosen within a set that depends on the set of feasible shaping parameters. For each possible shaping parameter b', which spreads a max burst size on $n = \left\lceil \frac{b}{b'} \right\rceil$ cycles, and each possible period T_{res} dividing the hypercycle (in number of cycles), we can have patterns of reservation height b' and period T_{res} iff

- $b' \cdot n \geq b + n \cdot r \cdot T_{res}$, which means that the transmission pattern will absorb the arrival curve;
- $n \cdot T_{res} \leq HC$, since a burst needs to be transmitted in one hypercycle.

For a given reservation height b' and period T_{res}, there are T_{res} possible patterns, since there are T possibilities for the first active cycle (see Fig. 3). This gives a set of transmission patterns, referred to as Π_f, of polynomial size. For each pattern $\pi \in \Pi_f$, its shaping delay is $n \cdot T_{res}$ and denoted as $D(\pi)$.

6.4.2.2 Pricing without pathbook Let us consider a pattern π, with delay $D(\pi)$. For each flow $f \in \mathcal{F}$, let introduce an extended graph $G^f_{\text{A-LDN}} = (\bar{V}, \bar{E})$ of G where

- $\bar{V}$ is the set of couples (v, C) such that $v \in V$ and $C \in \{1, ..., D_f^{max}\}$, i.e., each node of the original graph is duplicated for each possible arriving cycle, where the arriving cycle is the sum of all mappings and shifts experienced along a path.
- $\bar{E} = \{((v, C), (v', C + \delta(v, v'))) | (v, v') \in E, c \in \mathcal{C} \text{ and } C + \delta(v, v') \leq D_f^{max}\}$, where $\delta(v, v')$ is the mapping between v and v' (see Sect. 5), i.e., the set of links between a couple (node, cycle).

For each link $((v, C), (v', C + \delta(v, v'))) \in \bar{E}$ we consider an associated weight $\sum_{c' \in \mathcal{C}} \pi((C + c')\%HC)\alpha^*_{e,c'}$ where $\pi(\bar{c})$ is the reserved capacity in the pattern π at cycle $\bar{c}$.

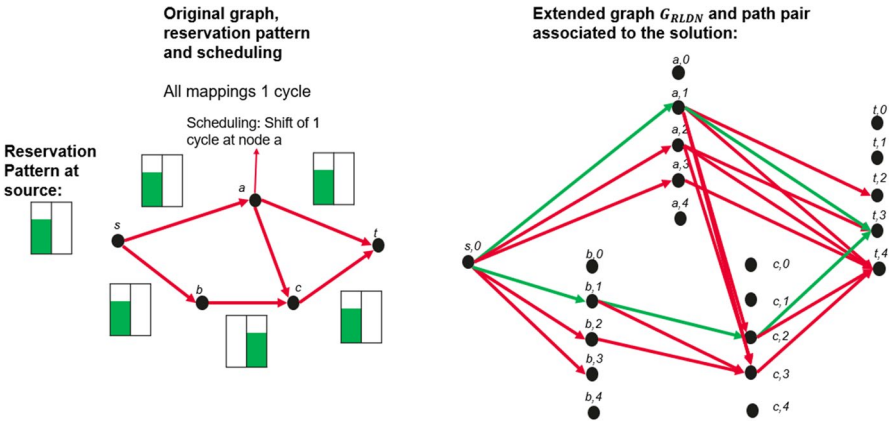


Fig. 5 Original graph and FRER solution, with the associated extended graph G_{FRER} and path pair. The graph G_{A-LDN} can be obtained by keeping only the arcs from G_{FRER} of the kind $(v, C), (v', C + \delta(v, v'))$, which require no scheduling

We then look for the shortest path which has as destination anyone of the nodes (t_f, C) of the extended graph. The end-to-end delay constraint is ensured by the set of possible destination nodes in the graph.

Remark that in practice, the extended graph does not need to be constructed for each flow and pattern. We use a dynamic programming algorithm which computes this shortest path. The same is true for extended graph FRER pricing algorithms.

6.4.3 Pricing with pathbook

If the set of paths is known, then the column with the largest violation of the dual constraint (4) is found by looking at the dual costs for each path and each transmission pattern.

6.4.4 FRER

6.4.4.1 Solution definition For a given path p, we associate a set of ordered integer $i \in \{0, .., HC\}$ of the same size. Each integer of this set gives the scheduling, i.e. the shift expressed in number of cycles, done in the node corresponding to the source of the link.

A solution is defined by a set of selected flows, and for each of these flows the algorithm provides:

- A transmission pattern
- A path pair verifying the node-disjointness and the reliability constraints
- A pair of scheduling sets to be applied on each intermediate nodes of each path

The set of possible transmission patterns in the same way as for A-LDN.

6.4.4.2 Pricing without pathbook Let us consider a pattern π. For each flow $f \in \mathcal{F}$, let's introduce an extended graph G^f_{FRER} based on $G^f_{\text{A-LDN}}$ where scheduled links are added. As shown in Fig. 5, scheduled links are defined as: $\bar{E} = \{(v, C), (v', i + C + \delta(v, v'))|(v, v') \in E, C\{1, .., D^{max}_f\}, i \in \mathcal{C}\}$, where $\delta(v, v')$ is the mapping between v and v' (see Sect. 5). The set of nodes is the same as for the A-LDN pricing. The set of arcs is larger, in order to account for the possibility of scheduling at each node in the graph.

For each link $((v, C), (v', i + C + \delta(v, v')) \in \bar{E}$ we consider an associated weight $\sum_{c' \in \mathcal{C}} \pi((C + i + c')\%HC)\alpha^*_{e,c'}$ where $\pi(\bar{c})$ is the burst size of the cycle $\bar{c}$. It represents the dual cost of the pattern if it uses arc (v, v') with a scheduling shift of i cycles, and it arrives at node v with a sum of mappings and scheduling of C cycles along the path in use.

Solving the pricing problem consists in finding, in this extended graph, two paths respecting the node-disjointness for the original graph, and the reliability constraint. The scheduling is given by the arcs used in the extended graph.

We could prove that the pricing problem is NP-complete, however we omit it here due to the lack of space.

6.4.4.3 Pricing with pathbook Let us consider a pattern π and a path pair (p, p') which verifies the node-disjointness and reliability constraint. In this case, for each $f \in \mathcal{F}$, we can consider a graph $G^f_{\text{FRER}}[p \cup p']$ where nodes not on p or p' are removed and all incident links of these nodes are removed.

A solution to the pricing is given by two paths arriving at the same possible destination node, one of the nodes $(t_f, C), C \leq D^{max}_f$. This is given by solving a min cost flow problem where all the links of the extended graph have a capacity of 1.

6.5 Advanced Solving Method

To solve the problem using the column generation, we proceed as follows:

1. Solve the RMP. This provides values of the dual variables.
2. Solve the pricing problem. For each flow, try to find a variable which can improve the objective. Insert into the RMP formulation the variable which violates (4) the most (if any). If no such variable is found for all flows, go to 3. Else, go to 1.
3. Set the domain of definition of all variables found to $\{0, 1\}$ and solve the formulation using an ILP solver.

This algorithm can be adapted for each variant (LDN, A-LDN or FRER) with or without pathbook. The only part of the algorithm that is specific to each problem variant is the pricing.

7 Numerical Results

In this section, we show through numerical results and packet-level simulations: 1) the compliance of LDN, A-LDN, and FRER to the end-to-end delay requirements, 2) the throughput improvement of A-LDN compared to the original LDN solution presented in [1], and 3) the performance comparison between A-LDN and FRER showing the trade-off between reliability and throughput, and the benefits of introducing a pathbook for FRER to reduce the complexity. In order to showcase these results, we developed three families of algorithms, whose utilization will be reflected by the name used in each plot. For the first approach, referred to as CGX, the different Column Generation (CG) routines presented in Sect. 6 are rounded up to an integer solution by solving an ILP on the solution of the relaxed version of the problem. The second family of solutions, denoted as PB, is based on the pre-computation of a set of paths for each demand. These paths must respect QoS requirements, such as latency, be totally link disjoint (1+1) or respect a maximum packet loss probability (P_f^{loss}). The last family, used only for the FRER 1+1 version of the problem and referred to as 1+1 CGX, is based on the generation in the CG loop, more precisely in the pricing, of a scheduling and a path for each demand.

In this section, we will show results on the LDN and A-LDN architecture when executing CGX without pathbook, and the exact solution (not using column generation and solved with CPLEX) with pathbook (referred to as PB-EX); concerning FRER, we will show results solving the problem with pathbook and CGX (referred to as 1+1 PB-CGX), and with path computation in the pricing of CGX (referred to as 1+1 CGX).

7.1 Pathbook Generation

The set of pre-computed paths used in the experimental results for LDN and A-LDN is defined as follows. We have a desired number of paths N_p from which we want to choose from for each demand. Using Yen's algorithm [33], we then find the N_p shortest paths, where the shortest means that the number of hops is minimized.

For FRER, a routing strategy for a demand is not defined by a path but by a pair of node-disjoint paths verifying:

- *Node disjointness*: The two paths should have only the source and destination nodes in common.
- *Reliability*: The pair of paths should respect the reliability constraint, described in Sect. 5.3, $P_f^{success} = 1 - ((1 - P_{p_f}^{success}) \cdot (1 - P_{p'_f}^{success})) \geq 1 - P_f^{loss}$, where P_f^{loss} is given, p_f and p'_f are the two paths from the pair, $P_{p_f}^{success} = \prod_{e \in p_f} P_e^{success}$, and $P_{p'_f}^{success} = \prod_{e \in p'_f} P_e^{success}$. A packet is transmitted successfully on a path if no packet loss is experienced on all the traversed arcs, and it is transmitted successfully over a pair of paths if it is transmitted successfully on at least one of the paths.

In order to generate N_p path pairs, we first list the N_p most reliable paths, using Yen's algorithm, where the cost of an arc e is $-\ln(P^{success}(e))$ (which is positive, since $P_e^{success} \in [0, 1]$). Then, for each path in the list, we try to find a complementary path such that:

- the two paths are disjoint. This is done by forbidding the nodes of the primary path.
- the two paths are reliable enough. This is done by using as a cost, for each arc e, $-\ln(P^{success}(e))$.

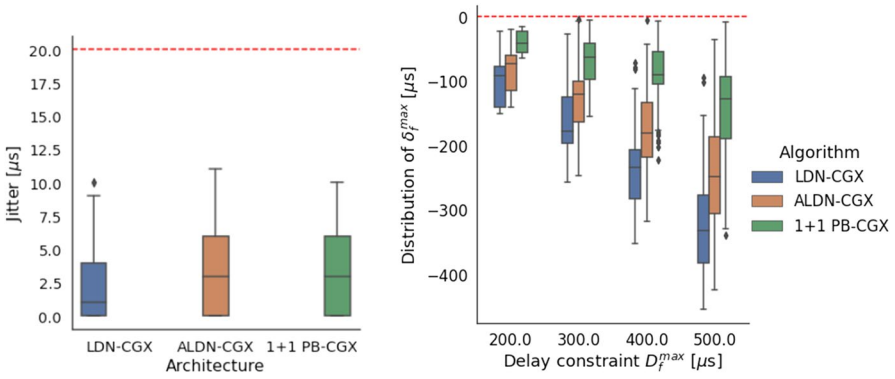
This complementary path can be found using a constrained shortest path algorithm, considering the end-to-end delay as a constraint. The path pair is then checked, and if it meets the reliability and QoS constraints, it is added into the list of possible path pairs for the flow.

7.2 Simulation Environment

We have extended the INET framework [34] in the OMNeT++ network simulator [9] to implement both LDN and A-LDN. On top of this latter, we have built two additional modules, one responsible for creating packet replicas over multiple paths and the other one in charge of scheduling packets at core nodes, in order to support FRER.

In the following, we first solve the control plane problem by running the algorithm within an optimizer, and then we execute network simulations to collect network traces to verify traffic acceptance and performance bounds. This framework allowed us to evaluate the correctness of both simulator and optimizer results through extensive tests, and extract statistics over multiple simulations. To achieve so, we built two different scenarios, leveraging on the NetworkX Python library:

- a scenario based on the AS 1239 ISP topology of Sprint [35], composed of $\|N\| = 315$ nodes and $\|E\| = 1944$ links. Every link has a propagation delay $\sim \mathcal{U}[10, 19]\ \mu s$ and a link capacity of 2.4Gbps. We consider up to 3000 flows following arrival curves described by parameters $r \sim \mathcal{U}[40, 600]$ Mbps and $b \sim \mathcal{U}[1000, 7500]$ B. For each flow we assign a fixed packet size $\sim \mathcal{U}\{1000, 1500\}$ B. Our goal is to show the improvement of A-LDN over LDN in terms of accepted traffic as a function of the number of flows n_f. For the sake of simplicity, we assume that all the flows have the same E2E delay constraint $D_f^{max} = 500\mu s$. We set this constraint to be sufficiently tight to get flow rejection if only shortest path resource reservation is carried out without caring about the other flows in the network;
- a random scenario, composed of 20 different random topologies, used to showcase the trade-off between reliability and throughput when introducing FRER. Each topology is characterized by a number of nodes $\|N\| \in \{32, 64, 128, 256\}$, with 5 instances for each topology size. We consider up to 600, 1200, 2400, 4800 flows for topologies with $\|N\| = 32, 64, 128, 256$, respectively. Flows are defined



(a) End-to-end jitter **(b)** End-to-end δ_f^{max}

Fig. 6 Box plots showing the E2E constraint satisfaction on the random topology. Fig. 6a shows the per-flow end-to-end jitter experienced in the simulator on the three architectures, when running the CGX algorithms (for FRER, with pathbook). Fig. 6b shows the distribution of δ_f^{max} over the flows (difference between the maximum experienced E2E delay in the simulator and D_f^{max}) for the same cases. All distributions lie below the red-dashed line, corresponding to the target constraint

as arrival curves with $r \sim \mathcal{U}[0.8, 1.2]$ Gbps and $b \sim \mathcal{U}[1000, 5000]$ B, and are generated as CBR applications with fixed packet size set to 1000 B. Every link has a propagation delay $\sim \mathcal{U}[10, 19]\ \mu s$ and a link capacity randomly drawn in the set $\{2.4, 4.8, 6, 12\}$ Gbps. We consider four different E2E delay constraints $D_f^{max} \in \{200, 300, 400, 500\}\mu s$, and we automatically assign them to the flows according to the number of hops between source and destination of the flow. Our goal is to measure the end-to-end delay and jitter for each flow, in order to prove that A-LDN and FRER still allow respecting both E2E delay and jitter bounds, and to show the impacts in terms of reliability and performance of FRER.

For the sake of simplicity, we consider, for all the simulations, a constant link packet error rate $PER = 5 \cdot 10^{-3}$. For FRER P_f^{loss}, we consider a maximum per-flow error rate $P_f^{loss} = 5 \cdot 10^{-3}$, constant over the flows. For the pathbook algorithms, we manually tuned the number of parallel paths parameter N_p and set it to $N_p = 5$.

We limit the execution time required to compute the optimization to 10 min for each instance. All simulations are characterized by $T = 10\,\mu s$ and $HC = 32$ cycles. Also, nodes have constant processing delay set to $22\mu s$. The nodes are not synchronized between them, i.e., the starting time of a cycle within a node may differ from the one in another node by a shift $\sim \mathcal{U}[1, 10)\mu s$. The OMNeT simulations last 20000 cycles.[1] Throughout this work, in each case we show a box plot, we remind that the marks denote the 95th, the 75th, the 50th, the 25th, and the 5th percentiles. The

[1] We manually set the simulation duration to achieve a 95% confidence interval of having a maximum error of 0.001 when estimating the packet error rate in the random scenario.

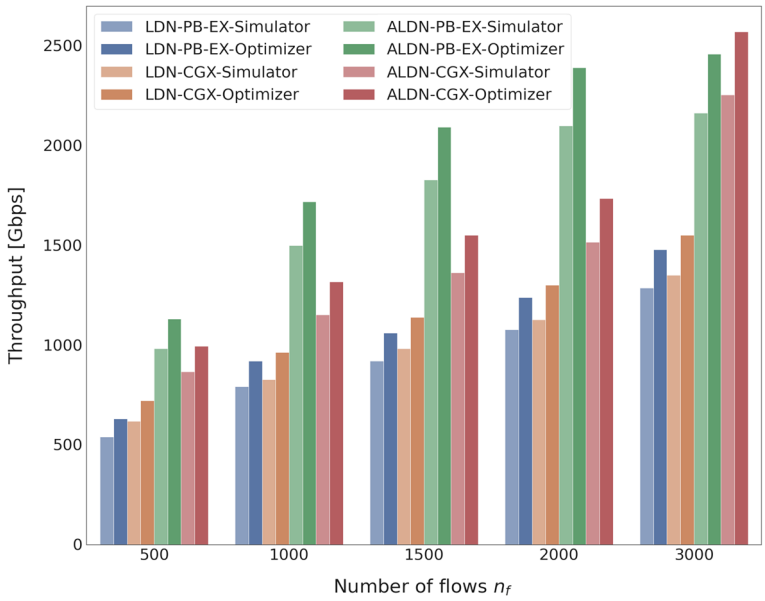


Fig. 7 Bar plots showing the between LDN and A-LDN architectures on the PB-EX and CGX algorithms in terms of measured (simulator) and estimated (optimizer) throughput, as a function of the number of flows n_f in the simulation

optimizer and the simulator are executed on an Ubuntu server with 500 Gb of RAM and up to 16 cores available for the parallelization.

7.3 Satisfaction of E2E Constraints

We start by presenting validation results on jitter and delay bounds on the random topology. In Fig. 6, each box plot represents the distribution over all the flows for the measured maximum end-to-end jitter and delay in the three architectures. We used the CG routine without pathbook for LDN and A-LDN, and the CG routing with a pathbook for FRER to reduce computational time. In all cases, the rounding is realized via an ILP. Referring to Fig. 6a, all the flows have a maximum jitter that lies below the constraint of $2T = 20\mu s$, proving that the architectures provide bounded jitter. In Fig. 6b, we group the flows by their delay constraint D_f^{max}, and show the distribution of δ_f^{max}, i.e., the difference between the maximum experienced E2E delay in the simulator, and D_f^{max}. The plot shows that the E2E latency constraint is satisfied for each flow. Similarly to results presented in [1], the simulations confirm that the introduction of transmission patterns in A-LDN and FRER does not impact performance guarantees. The E2E delays for A-LDN and FRER are closer to the threshold, as the packets can be delayed by transmission patterns and scheduling (for FRER only), even if the QoS is respected.

7.4 LDN and A-LDN Comparison

In Fig. 7, we compare the performance of LDN and A-LDN, in terms of accepted throughput, as a function of the number of flows n_f in the Sprint topology for both CGX and PB-EX. Our goal is to quantify the gain when using the advanced scheduling of A-LDN and the loss when using a pathbook. The global network throughput is obtained as the sum of average rates for accepted flows by the optimizer and compared against packet measurements over all the accepted flows in the simulator. The running time required by the optimizer to solve the planning problem is shown in Sect. 7.6.

We first observe that the mathematical model presented in Sect. 6.1 well captures the LDN behavior, as the optimizer performance is very close to the one of the simulator. The slightly positive difference between the optimizer prediction and the simulations measurements are mainly explained by approximations when implementing Constant BitRAte (CBR) applications in the simulation. The CGX algorithm performs well for both LDN (where it slightly improves PB-EX) and A-LDN, while reducing noticeably the running time, as shown in Sect. 7.6, especially for the A-LDN case; for $n_f = 3000$, not only the complexity is highly reduced, but CGX improves the performances of PB-EX for A-LDN. As expect, the use of a pathbook affects the accepted throughput as routing options are restricted. However, the loss is reasonable when compared to the gain in execution times (Sect. 7.6). We also notice that A-LDN outperforms LDN in terms of throughput with an improvement as high as 67% for the CGX algorithm, for $n_f = 3000$, justifying the use of transmission patterns at the iGWs.

7.5 Reliability: Throughput Tradeoff

In this section, we present the improvement, in terms of reliability, of FRER 1+1 and FRER P_f^{loss} over A-LDN, by showing the results in the random topology. We run 20 simulation instances for A-LDN, FRER 1+1 and FRER P_f^{loss}. We choose for A-LDN the CGX algorithm as the best solution in terms of both complexity and throughput, while for FRER 1+1 we consider both the 1+1 PB-CGX and the 1+1-CGX solution. Last, we solve FRER P_f^{loss} with PB-CGX.

For each simulation and for each flow $f \in \mathcal{F}$, we compute the packet delivery rate $PER_f = N_f^{delivered}/N_f^{packets}$, being $N_f^{delivered}$ the number of delivered packets for flow f, and $N_f^{packets}$ the total amount of packets generated for flow f experienced in OMNeT++, and calculate the minimum packet delivery rate $PDR = min(PER_1, ..., PER_{\mathcal{F}})$ for each simulation. Additionally, for each simulation of the FRER architectures, we measure the experienced throughput in the simulator, and we compute the gain of A-LDN over these solutions, computed as $G_i = \frac{THR_{i,A} - THR_{i,F}}{THR_{i,F}}$, being $THR_{i,A}$ the throughput experienced in the i_{th} simulation on the A-LDN architecture, and $THR_{i,F}$ the throughput on FRER.

In Fig. 8a and 8b we represent the tradeoff between the throughput loss and the reliability improvement when comparing A-LDN to FRER. Fig. 8a shows the distribution of the gain of A-LDN over FRER 1+1 and FRER P_f^{loss} as a function of the

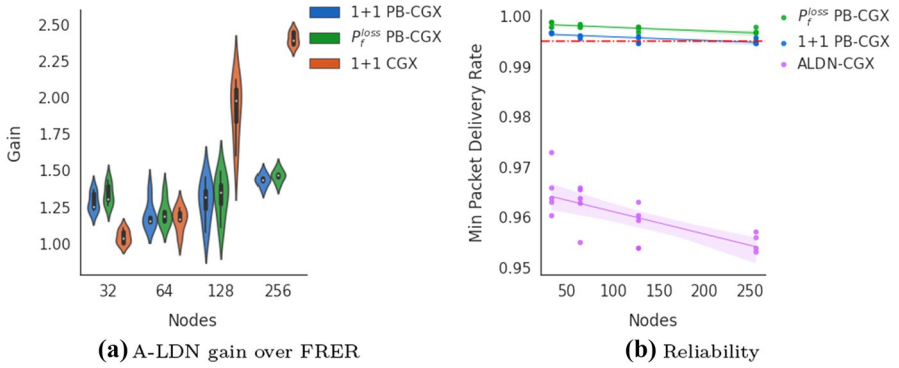


Fig. 8 Violin plots of A-LDN gain over FRER for different topology sizes, expressed as number of nodes, in 8a. Scatter plot of minimum packet delivery rate (PDR) among the flows for A-LDN, FRER 1+1 and FRER P_f^{loss}, compared to the target constraint (red dashed line), in 8b; each architecture is associated to a straight line with a faded area, representing the linear regression and its confidence interval

topology size, expressed in terms of number of nodes, through violin plots. First, we notice how, in the FRER 1+1 architecture, the pathbook scales well when increasing the topology size, especially if compared to the 1+1 CGX, with a huge improvement particularly experienced in bigger networks, due to an evident improvement in the complexity of the problem, as further described in Sect. 7.6. The slight increase of the gain experienced also in the pathbook solutions for the biggest network (256 nodes) is motivated by the time limits, fixed to 10 min, regardless from the topology size. We also not A-LDN architecture accepts, on average, more than the double of the traffic flowing in the FRER architecture. We explain this by reminding that FRER doubles the traffic injected for each demand.

Fig. 8b depicts the reliability improvement led by the introduction of the backup path and, in addition, by the reliability requirement. First, we notice higher variance in the packet delivery rates for simulations in A-LDN, as this index depends much more on the topology as it does in FRER. The noticeable increase in reliability is proved especially for bigger networks, as FRER is able to scale very well over the topology size. The introduction of the P_f^{loss} reliability constraint allows forcing the selection of the correct paths in order to satisfy the requirement, while maintaining almost similar performances in terms of throughput. This is particularly visible in Fig. 8b for bigger topologies (i.e, 128 and 256 nodes), as in some simulations we experienced a packet delivery rate lying below the QoS requirements for FRER 1+1, fixed by the introduction of the P_f^{loss} constraint.

7.6 Execution Times

Ultimately, in this section we present the execution times for all the aforementioned algorithms in the ISP topology, as a function of the number of flows N_f. For a fair comparison between the algorithms, we performed simulations with

Table 2 Running time on LDN, A-LDN and FRER 1+1 for the presented algorithms. Both modeling and optimization time are taken into account, with the latter being limited to 10 min

	LDN		A-LDN		FRER	
N_f	CGX	PB-EX	CGX	PB-EX	1+1 PB-CGX	1+1 CGX
500	4s	9s	380s	330s	140s	840s
1000	7s	13s	400s	913s	165s	980s
1500	12s	22s	420s	1250s	260s	1540s
2000	13s	24s	550s	2050s	440s	1790s
3000	18s	37s	630s	3800s	600s	3900s

the FRER architecture in the first scenario, achieving similar results in terms of performance as those presented in Sect. 7.5. When measuring the complexity of the models in terms of execution time, we also consider, in addition to the time required to perform the optimization (limited to 10 min), the model creation, that especially for the exact algorithms, has a noticeable impact on the overall computation time. Table 2 summarizes this effort. It is clear how PB-EX introduces higher execution times compared to CGX, particularly for the A-LDN architecture, justifying the choice of the column generation to guarantee scalability at the cost of a possible slight decrease in performance, as shown in Sect. 7.4. In FRER, 1+1 PB-CGX outperforms 1+1 CGX in terms of scalability, as anticipated in Sect. 7.5, motivating the use of the pathbook for large topologies.

8 Conclusion

In this paper, we have presented an extension of the Large-scale Deterministic IP Networking (LDN), referred to as Advanced LDN (A-LDN). Beside being shaped at the iGW as in LDN, in A-LDN different transmission patterns can be chosen in order to schedule traffic and better exploit the available capacity. We also described how we can implement FRER on top of A-LDN: by adding the possibility of scheduling inside the nodes, we can enforce that the replicas of the packets are received within the same cycle. In this way, we ensure that the network is stateless and can scale properly.

In order to maximize the acceptance of traffic in the network, we presented different declinations of the column generation routine used in the control plane, with and without a set of pre-computed paths. Results show that the A-LDN improves LDN by 67% in terms of accepted traffic, and FRER provides higher reliability at the price of larger bandwidth utilization. The use of pre-computed set of paths allows reducing the running time while keeping acceptable performance for traffic acceptance.

Future research direction could lead to considering distributed solutions that run with limited or even without support from the controller. In this case, attention should be paid to avoid over-reserving link capacity as the information shared between adjacent nodes, required to compute and set up the paths and the

transmission patterns, may be outdated or missing. Another research direction would be to integrate more accurate probability models for protection in the case where paths are only partially disjoint.

References

1. Liu, B., Ren, S., Wang, C., Angilella, V., Medagliani, P., Martin, S., Leguay, J.: Towards Large-Scale Deterministic IP Networks. In: IFIP Networking (2021)
2. Parvez, I., Rahmati, A., Guvenc, I., Sarwat, A.I., Dai, H.: A survey on low latency towards 5G: RAN, core network and caching solutions. IEEE Commun. Surv. Tutor. **20**, 4 (2018)
3. Grossman, E.: Deterministic Networking Use Cases. RFC Editor (2019). https://doi.org/10.17487/RFC8578. https://rfc-editor.org/rfc/rfc8578.txt
4. Li, R.: Towards a New Internet for the Year 2030 and Beyond. Proc. 3rd Annual ITU IMT-2020/5G Workshop Demo Day (2018)
5. Nasrallah, A., Balasubramanian, V., Thyagaturu, A.S., Reisslein, M., Elbakoury, H.: Cyclic Queuing and Forwarding for Large Scale Deterministic Networks: A Survey. ArXiv **abs/1905.08478** (2019)
6. Deterministic Networking Architecture. RFC 8655 (2019)
7. Qiang, L., Geng, X., Liu, B., Eckert, T., Geng, L., Li, G.: Large-Scale Deterministic IP Network. IETF Draft draft-qiang-detnet-large-scale-detnet-05 (September 2019)
8. IEEE Standard for Local and metropolitan area networks–Frame Replication and Elimination for Reliability. IEEE Std 802.1CB-2017, 1–102 (2017). https://doi.org/10.1109/IEEESTD.2017.8091139
9. Varga, A., Hornig, R.: An overview of the omnet++ simulation environment. In: Proc. Simutools (2008)
10. Quan, W., Yan, J., Jiang, X., Sun, Z.: On-line traffic scheduling optimization in ieee 802.1 qch based time-sensitive networks. In: Prox. IEEE HPCC (2020)
11. Máté, M., Simon, C., Maliosz, M.: Asynchronous Time-Aware Shaper for Time-Sensitive Networking. In: 2021 17th International Conference on Network and Service Management (CNSM), pp. 565–571 (2021). https://doi.org/10.23919/CNSM52442.2021.9615545
12. Finn, N.: Introduction to time-sensitive networking. IEEE Commun. Stand. Mag. **2**(2), 22–28 (2018). https://doi.org/10.1109/MCOMSTD.2018.1700076
13. Chen, M., Geng, X., Li, Z.: Segment Routing (SR) Based Bounded Latency. Internet-Draft draft-chen-detnet-sr-based-bounded-latency-00, Internet Engineering Task Force (October 2018)
14. Krolikowski, J., Martin, S., Medagliani, P., Leguay, J., Chen, S., Chang, X., Geng, X.: Joint routing and scheduling for large-scale deterministic ip networks. Comput. Commun. **165**, 33–42 (2021)
15. Huang, Y., Wang, S., Feng, T., Wang, J., Huang, T., Huo, R., Liu, Y.: Towards network-wide scheduling for cyclic traffic in ip-based deterministic networks. In: 2021 4th International Conference on Hot Information-Centric Networking (HotICN), pp. 117–122 (2021). IEEE
16. Ergenç, D., Fischer, M.: On the reliability of ieee 802.1cb frer. In: IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, pp. 1–10 (2021). https://doi.org/10.1109/INFOCOM42981.2021.9488750
17. Casazza, M., Ceselli, A.: Optimization algorithms for resilient path selection in networks. Comput. Oper. Res. **128**, 105191 (2021)
18. Oki, E., Matsuura, N., Shiomoto, K., Yamanaka, N.: A disjoint path selection scheme with shared risk link groups in gmpls networks. IEEE Commun. Lett. **6**(9), 406–408 (2002)
19. Silva, J., Gomes, T., Fernandes, L., Simoes, C., Craveirinha, J.: An heuristic for maximally srlg-disjoint path pairs calculation. In: 2011 3rd International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), pp. 1–8 (2011). IEEE
20. Nasrallah, A., Balasubramanian, V., Thyagaturu, A., Reisslein, M., ElBakoury, H.: Large scale deterministic networking: A simulation evaluation. arXiv preprint arXiv:1910.00162 (2019)
21. Addanki, V., Iannone, L.: Moving a step forward in the quest for deterministic networks (detnet). In: IEEE/IFIP Networking (2020)

22. Ergenç, D., Fischer, M.: Implementation and orchestration of ieee 802.1 cb frer in omnet++. In: 2021 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1–6 (2021). IEEE
23. Le Boudec, J.-Y., Thiran, P. (eds.): Network Calculus, pp. 3–81. Springer, Berlin (2001)
24. Eckert, T., Bryant, S., Malis, A.G.: Deterministic Networking (DetNet) Data Plane - MPLS TC Tagging for Cyclic Queuing and Forwarding (MPLS-TC TCQF). Internet-Draft draft-eckert-detnet-mpls-tc-tcqf-01, Internet Engineering Task Force (October 2021)
25. Siachalou, S., Georgiadis, L.: Algorithms for precomputing constrained widest paths and multicast trees. IEEE/ACM Trans. Netw. **13**(5), 1174–1187 (2005). https://doi.org/10.1109/TNET.2005.857117
26. Acharya, S., Chang, Y.J., Gupta, B., Risbood, P., Srivastava, A.: Precomputing high quality routes for bandwidth guaranteed traffic. In: IEEE Global Telecommunications Conference, 2004. GLOBECOM '04., vol. 2, pp. 1202–12072 (2004). https://doi.org/10.1109/GLOCOM.2004.1378146
27. Shand, M., Bryant, S.: IP Fast Reroute Framework. RFC Editor (2010). https://doi.org/10.17487/RFC5714. https://www.rfc-editor.org/info/rfc5714
28. Litkowski, S., Bashandy, A., Filsfils, C., Francois, P., Decraene, B., Voyer, D.: Topology Independent Fast Reroute using Segment Routing. Internet-Draft draft-ietf-rtgwg-segment-routing-ti-lfa-08, Internet Engineering Task Force (January 2022). Work in Progress. https://datatracker.ietf.org/doc/draft-ietf-rtgwg-segment-routing-ti-lfa/08/
29. Finn, N.: Failure rates and P802.1CB (2013)
30. Desaulniers, G., Desrosiers, J., Solomon, M.M.: Column Generation. Cahiers du GERAD. Springer, Berlin (2005)
31. Wilhelm, W.E.: A technical review of column generation in integer programming. Optim. Eng. **2**(2), 159–200 (2001)
32. Juttner, A., Szviatovski, B., Mecs, I., Rajko, Z.: Lagrange relaxation based method for the qos routing problem. In: Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society, vol. 2, pp. 859–8682 (2001). https://doi.org/10.1109/INFCOM.2001.916277
33. Yen, J.Y.: An algorithm for finding the shortest routes from all source nodes to a given destination in general networks. Q. Appl. Math. **27**(4), 526–530 (1970)
34. Mészáros, L., Varga, A., Kirsche, M.: In: Virdis, A., Kirsche, M. (eds.) INET Framework, pp. 55–106. Springer, Cham (2019)
35. https://research.cs.washington.edu/networking/rocketfuel/