Towards Large-Scale Deterministic IP Networks

†Bingyang Liu, †Shoushou Ren, †Chuang Wang, *Vincent Angilella,
*Paolo Medagliani, *Sebastien Martin, *Jeremie Leguay
*Huawei Technologies, Paris Research Center.
†Huawei Technologies, Beijing Research Center.

***Abstract*—Deterministic performance is a key enabler for 5G networking. In this context, we present a highly scalable Large-scale Deterministic Network (LDN) architecture providing end-to-end latency and bounded jitter guarantees in IP networks. At the data plane, flows are first shaped at ingress gateways using gate-control queues, achieving a very fine granularity compared to existing state of the art solutions. Inside the network, traffic is scheduled using an asynchronous cyclic queuing mechanism that can be implemented in real devices as it requires only 3 FIFO queues. The data plane relies on standard IP routing and a quasi-static mapping table to deterministically aggregate and forward packets over cycles with a low complexity in $O(1)$. For the control plane, we present an advanced column generation algorithm to quickly take admission control decisions in large-scale networks. For a set of flows, it determines acceptance and selects the best shaping and routing policy.**

Through a proof-of-concept implementation and simulations, we show that our LDN architecture can guarantee end-to-end latency and bounded jitter. We also demonstrate that our advanced control plane algorithm brings an improvement up to 40% in terms of accepted traffic over classical routing.

***Index Terms*—Deterministic networks, Large-scale IP networks, Bounded delay, Bounded jitter.**

I. Introduction

The 5th generation of networks is paving the road for latency-sensitive network services to enable a wide-range of Internet applications like factory automation, connected vehicles and smart grids [1]. Traditional Internet Protocol (IP) services allow reliable data delivery, but they cannot provide strict Quality of Service (QoS) guarantees. Certain classes of service can get preferential treatment but performance remains statistical. To frame the development of new technologies for deterministic IP networks, the IETF [2] and the ITU-T [3] 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) [4], 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. A common approach to bound latency and jitter at each hop for High Priority (HP) traffic is to use Cyclic Queuing and Forwarding (CQF) [5], which considers 2 queues on each port, opened and closed alternatively in a cyclic fashion. At any given time, one queue is for transmission and the other one is for reception. However, CQF is suitable for small networks with time synchronized nodes and short transmission links as a packet, sent in a given cycle, must be received during the same cycle and retransmitted at the next cycle.

To improve the scalability of TSN solutions, the IETF DetNet (Deterministic Networking) [6] group has been working on a Deterministic Networking (DN) architecture, where solutions such as Large-scale Deterministic Network (LDN) [7] and Cycle Specified Queuing and Forwarding (CSQF) [8] have been proposed to specify how traffic should be scheduled and forwarded. Even if they both operate at layer 3, CSQF requires complex per-packet scheduling policies [9], as each packet can have a custom forwarding cycle at each intermediate node, while LDN considers flow-level processing only at ingress nodes and it operates on flow aggregates at core nodes. In the rest of this paper, we build upon the LDN proposal as it is the most scalable.

To provide E2E latency and bounded jitter guarantees, several challenges must be addressed. As traffic bursts are common in IP networks due to their multiplexing nature [10], the elasticity of traffic needs to be controlled at the ingress, at the price of an extra latency depending on the burst size. Also, because IP networks usually contain a very large number of devices and users, a low complexity and scalable solution is required, to avoid core routers and switches to maintain per-flow states and achieve low complexity inside devices.

To address the aforementioned challenges, we present a complete LDN architecture, that extends current work at IETF [7], to guarantee deterministic E2E latency and bounded jitter for HP flows. At the data plane level, we consider a new fine-grained shaping method that scatters incoming bursts at ingress nodes and makes flows fit into the assigned cycles inside the deterministic network. The use of gate-control queues yields a 1.5 KB granularity for shaping, 10 times better than current state of the art solutions. At intermediate core nodes, the data plane relies on standard IP forwarding and schedules traffic using a cyclic queuing mechanism with 3 queues that does not require time synchronization between nodes. At each hop, a quasi-static mapping table is used to forward packets from incoming to outgoing cycles based on a label carried in the header of packets. Since core routers only use the carried label for packet scheduling, LDN is free of per-flow or per-packet orchestration at both data plane and control plane. To derive latency and jitter bounds, we present a theoretical performance analysis of this asynchronous scheduling and forwarding mechanism. Through a Proof-of-Concept (PoC) implementation on real devices and simulation results, we

ISBN 978-3-903176-39-3© 2021 IFIP

2021 IFIP Networking Conference (IFIP Networking) | 978-3-9031-7639-3/21/$31.00 ©2021110.23919/IFIPNETWORKING52078.2021.9472798

verify that performance can be guaranteed even in large-scale networks. At the control plane level, we introduce a centralized architecture and an efficient Column Generation and Randomized Rounding (CGRR) algorithm to decide the best shaping and routing parameters for a batch of incoming traffic requests. Thanks to a numerical evaluation, we show that our control plane algorithm can accept 40% more traffic than traditional routing, while ensuring deterministic E2E requirements. Overall, we demonstrate that our LDN implementation is highly scalable and suitable for real networks.

The rest of this paper is organized as follows. In Sec. II, we present some related works. Sec. III introduces the system design. The data plane and the control plane are presented in Sec. IV and Sec. V, respectively. Finally, Sec. VI describes performance results, before concluding in Sec. VII.

II. Related Work

Traditional data plane scheduling methods mainly include Round Robin (RR) [11], [12] and Weighted Fair Queuing (WFQ) [13], [14] schedulers. In RR-based methods, such as WRR and DRR, queues are scheduled one after another. At each round, an amount of data based on each queue's weight or deficit is scheduled. Besides being useful to prioritize traffic, these methods can be used to provide deterministic guarantees as latency bounds can be computed with network calculus [15]. However, these solutions yield highly conservative E2E latency bounds. They also suffer from a large jitter due to poor flow isolation. Methods like WFQ, inspired from Generalized Processor Sharing (GPS) [13], can provide better E2E latency bounds. However, they usually require devices to maintain per-flow states, preventing their use in large networks.

The IEEE 802.1 TSN task group has produced a series of solutions to provide deterministic transmissions. Unfortunately, they are limited to layer 2 and do not scale well to large IP networks due to issues such as time synchronization, high precision scheduling, short distance links, etc [16]. The IEEE 802.1Qav norm defines priority scheduling and Credit Based Shaper (CBS) to guarantee latency. However, all the nodes must maintain per-flow states to avoid burst formation. IEEE 802.1Qbv describes a TDM-like (Time Division Multiplexing) method, where each flow can only be transmitted in cycles at specific allocated time slots. It leverages on a time-based gate opening and closing at the output ports, operating at cycle-level, rather than slot-level. Thus, if a flow's slot is in the middle of a cycle, we need to control the multiplexing with other flows to guarantee the transmission time instant inside the proper cycle. This imposes constraints on the arrival time of each flow at packet-level and requires per-packet orchestration while doing path planning, considerably increasing the complexity to the control plane. IEEE 802.1Qch specifies the CQF method based on a 2-queues buffering to synchronize transmissions in a cyclic manner, with a per-hop bounded latency. 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.

To integrate TSN technologies into IP networks, the IETF DetNet working group has defined a general DN architecture [6]. LDN [7] and CSQF [8] drafts have been proposed for deterministic data scheduling and forwarding. The main differences are the following. In CSQF, (i) packet arrivals are required to be known in advance (i.e., time-triggered traffic), (ii) different packets of the same flow may be scheduled in different cycles, thus the nodes must maintain per-flow states, and (iii) a Segment Routing header is used to route traffic and schedule packets at each node, leading to a significant packet overhead. On the other hand, LDN introduces a 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 a (ii) cycle-based deterministic forwarding at core nodes in which no per-flow states are needed, making LDN scalable. Compared to the LDN draft [7], in this paper we detail how we implement efficient shaping and routing at data plane and control plane levels.

III. LDN Architecture overview

This section presents the high-level functionalities of the LDN architecture we implemented to guarantee E2E latency and bounded jitter. The next two sections (Sec. IV and Sec. V) will introduce the data and control planes in more details. For the sake of simplicity, we assume that all the devices are managed by the same network controller. We also assume that amongst all the traffic in the network, only a subset of IP flows are High Priority (HP) and need guaranteed performance, while the remaining flows are routed as low-priority Best Effort (BE) flows.

As shown in Fig. 1, the LDN network is composed by three elements: (i) user devices at the edge of the network that send and receive application traffic, (ii) edge and core network devices forwarding flows in the network, and (iii) a network controller. The control plane is implemented in the centralized controller and it orchestrates the configuration of network devices in charge of deterministic and end-to-end data transmissions. User traffic enters the LDN network via an Ingress Gateway (I-GW) and leaves it at an Egress Gateway (E-GW). As shown in Sec. IV-A, the I-GWs shape incoming traffic to cut down bursts and make them fit within transmission cycles along the paths. Network devices in the middle, referred to as *core nodes*, forward flow aggregates at each hop based on a cyclic division of time, to enforce E2E delay and jitter constraints. In this paper, we define the E2E latency of a packet as the time interval between the instant at which it enters into the I-GW and the instant at which it leaves the E-GW, and the jitter, as the maximum latency gap between consecutive packets of the same flow.

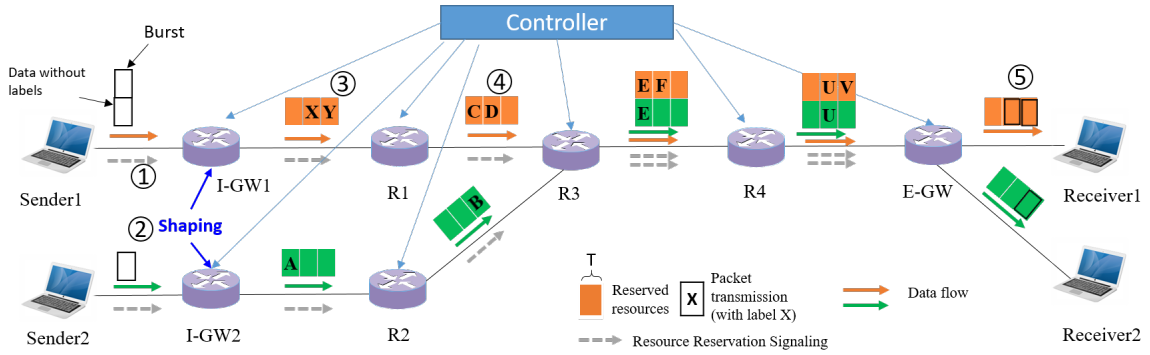


Fig. 1. Architecture overview of LDN. 1) Admission control is performed before senders 1 and 2 start emitting flows. 2) I-GWs do per flow shaping. For instance, I-GW 1 will scatter the received 2-packets burst into two consecutive cycles. 3) I-GWs mark HP packets with a local label and schedule packets according to their labels. 4) Core routers rewrite labels for each HP packet to identify the next transmission cycle. Packets from different upstream nodes can be mapped to the same label on a router (like for R3 and R4). 5) E-GWs strip the labels before forwarding packets to their destinations.

A. Data plane

Deterministic forwarding in the LDN is assured by dividing transmission time into cycles of duration T on each port of all the nodes in the network. The starting time of the cycles on a node is based on its local clock, i.e. network devices are not required to be time synchronized. As explained in more details in Sec. IV, the scheduler is based on 3 gate-control queues that are opened and closed in a round robin fashion.

At the ingress, I-GWs maintain per-flow states to control the shaping and forwarding of each incoming flow. As described later in Sec. IV-A, a fine-grained shaping method is used to control the bandwidth utilization of each flow in transmission cycles. When sending out a packet, the I-GW also adds a *label*, marking its sending cycle.

Core network devices forward each flow over a specific path and use a cycle-based scheduling at each hop to enforce delay and jitter bounds. Once a labeled packet is received, the forwarding engine updates the label to specify the transmission cycle to be used at the next hop. A *mapping table* is maintained on each device and it is used to map incoming cycles into outgoing cycles, i.e., incoming labels into outgoing labels (more details in Sec. IV-C). To forward packets, the outgoing port, i.e., the next hop, is determined in a standard and conventional manner via IP forwarding (i.e., using FIB entries). Once the port is chosen, the transmission cycle is selected, starting from the label in the received packet header, using the corresponding entry in the mapping table. The mappings are not specific to the flows as packets with the same label are sent out in the same cycle, even if they belong to different flows and come from different upstream nodes. As the propagation delay between neighbor nodes and the processing delay can be upper bounded using statistical information, the mapping between sending cycles of two neighbor nodes is almost static. This decoupling between scheduling and routing improves scalability. As last step, once the E-GW receives a packet, it schedules the transmission, sending the packet out in the corresponding cycle. Differently from core nodes, the E-GW only needs to (i) find the outgoing cycle from the mapping table and (ii) remove the header field carrying the label.

The efficient bandwidth allocation and aggregation of different flows within the transmission cycles is managed by the control plane. As bandwidth sharing is controlled by ingress shaping and routing, core devices do not need to maintain per-flow states, making LDN highly scalable to large IP networks.

B. Control plane

The control plane is in charge of admission control, shaping and routing for high-priority flows. As short delay and low bandwidth consumption are usually opposite objectives, a smart resource allocation needs to be performed by the controller to ensure that a maximum amount of traffic can be accepted in the network and that all accepted flows meet their requirements. When a batch of new flows has to be admitted into the network, the centralized controller decides which flows could be accepted, as well as the shaping parameter and the path of each accepted flow. In particular, for the decision of the optimal shaping parameter, the controller has to consider that a smaller shaping parameter decreases the amount of capacity used along the path while, on the other hand, it introduces a predictable additional shaping delay. The controller has to ensure that this extra delay is compatible with the end-to-end delay constraints of priority flows.

Once the admission control decision is made, the installation of each flow can be performed in a distributed manner using a resource reservation protocol like RSVP [17] or by the centralized controller communicating directly with devices.

IV. Data plane mechanisms

This section presents in more details the data plane mechanisms at gateways and core nodes.

A. Ingress shaping at I-GWs with gate-control queues

Every flow can be characterized by a traffic envelope, also referred to as arrival curve $A_f(t)$, using the network calculus terminology [18]. For any arbitrary time interval of duration t, the cumulative data $C_f(t)$ sent by flow f can be constrained as follows:

$$C_f(t) \leq A_f(t) = r_f t + b_f \quad (1)$$

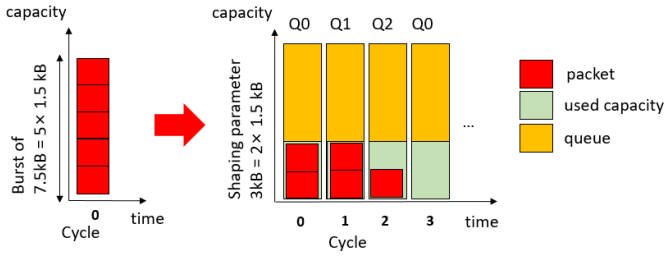


Fig. 2. Example of shaping of a burst of size $b_f = 7.5$ kB with a shaping parameter $b'_f = 3$ kB/cycle, 3 kB used on each cycle and shaping delay $3T$.

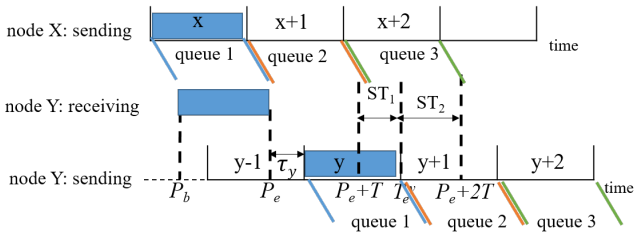


Fig. 3. Cyclic queue scheduling on core LDN nodes.

where b_f is the maximum burst size for f, r_f is the the arrival rate (upper bound average rate), and MPS_f is the maximum packet size for the flow. In our LDN implementation, these parameters are declared by the application and communicated to the network controller to perform admission control.

Fine-grained shaping. The shaping performed at I-GWs for a flow is meant to cut an incoming burst of size b_f into a smaller burst of size b'_f, referred to as *shaping parameter*. The objective is to make flow f fit into the network capacity. Typically, routers implement shaping in the Traffic Management (TM) module via a token bucket mechanism. However, due to the low frequency of network processors, a large volume of traffic has to be scheduled at each round. Thus, this method can only control burst sizes at coarse-grained (larger than 15 KB in most commodity routers) [19], [20].

To overcome this drawback, we propose a new shaping method based on multiple gate-control queues. Each I-GW divides time into cycles of duration T and each cycle is associated with a queue. A queue opens at the beginning of the corresponding cycle and closes at the end. Only when a queue is open, the packets inside can be transmitted. As shown in Fig. 2, a large burst will be scattered into different queues. With this shaping method, the burst after the I-GW can be as small as the Ethernet MTU packet size, i.e. 1.5KB.

Shaping delay. The incoming burst is spread over $\lceil \frac{b_f}{b'_f} \rceil$ contiguous cycles, provided that $b'_f \geq r_f T$. In this way, it is guaranteed that the flow f cannot send more than b'_f/MPS_f data packets in every cycle to core routers. This shaping introduces a maximum additional latency of $D_{shaping} = T\lceil \frac{b_f}{b'_f} \rceil$, which is the shaping delay experienced by the last packet of a maximum size burst. For instance, referring to the example in Fig. 2, a burst of size $b_f = 7.5$ kB is spread over 3 cycles considering a shaping factor $b'_f = 3$ kB/cycle, and the shaping delay of the last packet is no more than $3T$.

The set of considered shaping parameters for a flow f is $\mathcal{B}_f = \{MPS_f \lceil \frac{b_f}{nMPS_f} \rceil | n \in \mathbb{N}, \lceil \frac{b_f}{n} \rceil \geq r_f T\}$; where n is the number of cycles over which a burst is spread. Indeed, shaping parameters of the form $\frac{b_f}{n}$ are the ones using a smaller amount of capacity in each cycle for a shaping delay of nT.

In the control plane algorithm presented in Sec. V, $D_{shaping}$ is controlled by the shaping parameter to tackle the trade-off between the amount of required capacity and the E2E delay requirements. Indeed, the shaping parameter b'_f determines the capacity that a flow will occupy in each cycle and it also impacts the extra delay $D_{shaping}$ introduced by shaping.

B. Scheduling with 3 queues

Cyclic scheduling. At each port, 3 FIFO queues are scheduled in a round-robin fashion. Each queue is active for a period T and follows a calendar to open and close its gate. At any given time, only one queue is open (scheduled to transmit traffic) and the other two queues are closed (backlogging traffic). As shown in [21], the packet processing complexity of this round robin mechanism is $O(1)$, making this approach suitable for large-scale networks. During each cycle, HP and BE traffic may coexist. To guarantee QoS performance, BE traffic is enqueued into other lower priority FIFO queues. In order to avoid starvation of BE flows, the HP traffic should not occupy the entire bandwidth of each cycle. One of the goals of the control plane algorithm is to ensure that only a portion of the capacity is reserved for HP flows, leaving the spare capacity to BE traffic.

Fig. 3 illustrates how the data plane works in a simple two-nodes topology where we assume that node X is the only upstream node of Y and Y is allowed sending TC_Y data at each cycle, where C_Y is the outgoing link capacity of Y and T the duration of cycles. Node Y receives the first bit of the packets sent at cycle x at the time instant P_b and it receives in the worst case the last bit of the packets at the time instant P_e, where, $P_e = P_b + T$. According to the mapping table (explained more in details in Sec. IV-C), these packets will be scheduled for transmission in cycle y. Similarly, the packets sent in cycle $x+1$ will arrive at node Y between P_e and P_e+T and will be scheduled in cycle $y+1$. In Sec. IV-C, we will see that node Y will maintain the cycle mapping information for each of its upstream nodes.

Number of queues. In the following, we explain why 3 queues are required to forward traffic. Let's now focus on the time interval $(P_e + T,\ P_e + 2T]$. We first consider the sub-interval $ST_1 = (P_e + T,\ T_e^y)$, where T_e^y is the end of cycle y. For any time instant during ST_1, there are always packets belonging to cycle x (labeled now with y) waiting to be sent in queue y. One extra queue (e.g., queue 1 in node Y in Fig. 3) is needed to cache packets sent in cycle x. Moreover, we can also see that, during ST_1, node X has already received all the packets belonging to cycle $x+1$ and received part of data of cycle $x+2$. Thus, two queues (e.g., queue 2 and queue 3 in node Y in Fig. 3) are needed to cache the packets belonging

to cycle $x+1$ (with the new label $y+1$) and cycle $x+2$ (with the new label $y+2$) before sending them out in cycles $y+1$ and $y+2$, respectively. At T_e^y, all the packets belonging to cycle x have been sent out and queue 1 is empty.

In the sub-interval $ST_2 = (T_e^y,\ P_e + 2T)$, queue 2 is open and scheduled to send packets, while queue 1 is still empty and queue 3 is receiving packets belonging to cycle $x+2$. At time $P_e + 2T$, queue 1 will be re-used to receive and cache packets belonging to cycle $x+3$, and so on. Thus, node Y needs at most 3 queues per interface to schedule and forward packets during the time period $(P_e + T,\ P_e + 2T]$. Only three gate-control FIFO queues are needed on each interface.

C. Forwarding with labels and IP routing

Mapping table. The mapping table maintained at each core node contains an entry for each upstream neighbor (i.e., in-port) to determine the corresponding output label (i.e, output cycle or queue) according to the carried label. These mappings are applied to all priority packets and labels are encoded into packet headers. Note that packets from different upstream nodes may be mapped into the same sending cycle and share the same queue, thus aggregation scheduling can be performed. All the packets at the same output port with the same new label will be scheduled into the same queue and, by consequence with the same label, in the same cycle, independently of where they come from. As explained before, the next hop is determined using standard IP routing. Besides, the control plane algorithm makes sure that, at each cycle T and port k, the capacity constraint is met. In particular that the total used bandwidth R_k is smaller than the capacity C_k.

Fig. 3 illustrates how packets are forwarded. Let's assume that node X is an upstream neighbor of node Y and that it sends a packet marked with label x to identify the outgoing queue. When node Y receives this packet, it replaces the label x with a new local label y according to the local mapping table. Before forwarding the packet, node Y introduces a delay τ_Y to wait for the next cycle y.

Learning mappings. The learning of mappings at nodes is a critical point. Since there is no time synchronization in LDN, packets with label y after the label swapping may come from different upstream nodes at the same time. Thus, a constraint is that all the packets, expected to be transmitted in cycle y, must be received before the beginning of cycle y, otherwise node Y may not be able to send all the packets belonging to this cycle. This means that the queue x of node X should be mapped to the first available cycle after the end of the reception cycle at node Y, referred to as P_e. The mapping relationship between X and Y depends on the propagation delay of the link e between X and Y. In the case d_e changes for some reasons, a dynamic detection and mapping re-learning mechanism is required. Due to space limitation, the implementation details are not given in this paper. Once determined, the forwarding delay associated with mappings is provided to the control plane so that end-to-end QoS requirements of flows can be satisfied. From Fig. 3, it turns out that the overall delay l_e for a packet to reach the out-port of node Y is given by $l_e = d_e + \tau_Y + T$.

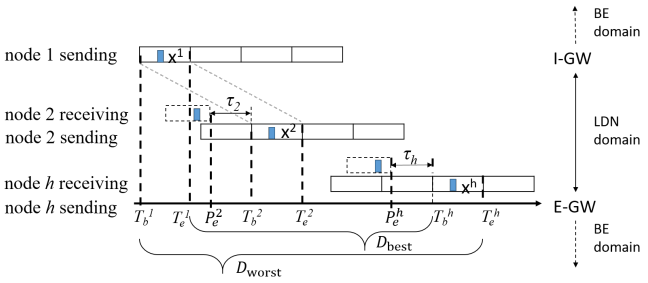


Fig. 4. End-to-end latency and jitter analysis.

D. Bounds on E2E latency and jitter

According to [5], the E2E delay and jitter bounds are $2hT$ and $2T$, respectively, where h is the number of nodes of the E2E path and T is the cycle duration. In this section, we are going to provide a more detailed evaluation of the worst and best queuing delay experienced by all the packets within a flow. We also provide some useful theoretical upper and lower bounds on the queuing delay that can experienced by all the flows in the network. Let's focus on a generic flow f with an h-hop path. The forwarding time for the packets belonging to a specific cycle x at the first hop is shown in Fig. 4. Since the mapping information is fixed between each pair of neighbors, all the sending cycles $\{x^1, \ldots, x^h\}$ along the path are fixed as well and known if x^1 is known.

Let T_b^i be the beginning of the cycle to be used for transmission according to the carried label and T_e^i be the end of the mapped cycles on hop i, where $1 \le i \le h$. Let $\tau_i \in [0, T)$ be the time interval between P_e^i and T_b^i, i.e., the beginning of the sending cycle on node i, where $2 \le i \le h$. From Fig. 4, the worst E2E queuing delay happens when a packet, sent at the first hop at T_b^1, is forwarded, at the last hop, at T_e^h, i.e., a packet transmitted at the beginning of the transmission cycle in the first hop is sent out by the E-GW as the last packet of the transmission cycle. It can be expressed as:

$$D_{\text{worst}} = T + \sum_{i=2}^{h} (\tau_i + T) \tag{2}$$

In the best case, a packet, sent at the first hop at time T_e^1, is sent out at the last hop at time T_b^h. Thus, the minimum end-to-end queuing delay can be expressed as

$$D_{\text{best}} = \sum_{i=2}^{h-1} (T + \tau_i) + \tau_h \tag{3}$$

For a specific flow, all the packets experience the same τ_i. Thus the jitter can be computed as

$$Jitter = D_{\text{worst}} - D_{\text{best}} = 2T \tag{4}$$

Note that the best and worst delay bounds experienced by each flow can vary as the real value of τ_i can be different flow-by-flow. Recalling that by construction $\tau_i < T$, the theoretical upper bound on the queuing delay can be computed assuming that there exists at least one flow with $\tau_i = T\ \forall i$. The

corresponding theoretical upper bound is $\Phi_{\text{up}} \leq (2h-1)T$. Vice versa, the theoretical lower bound can be computed considering that there exists one flow with $\tau_i = 0\ \forall i$, giving a lower bound of $\Phi_{\text{low}} \geq (h-2)T$.

V. Control plane algorithm

In this section, we present an efficient control plane algorithm to route a batch of incoming flows over a LDN.

A. System model

The admission control decisions for a batch of incoming traffic requests are taken by the controller which computes the best shaping parameters and routing paths to maximize the total amount of traffic accepted into the network. While admission control can be performed for single arrivals, incoming traffic requests can be processed by batch to further optimize resources. In this case, a maximum flow processing time can be enforced to trigger admission control.

Let's assume an input network characterized by:

- a graph $G = (V, E)$ representing the network topology;
- a delay l_e for each link $e = (u, v) \in E$ as introduced in Sec. IV-C;
- a bandwidth capacity C_e of each link $e \in E$. The cycle capacity, which is the maximum traffic that a link can transmit during one cycle duration, is given by $C_e T$.

We denote the set of incoming flows $\mathcal{F}$, and for each flow $f \in \mathcal{F}$ the controller gets in particular:

- the arrival curve $A_f(t) = r_f t + b_f$, where r_f is the arrival rate and b_f the maximum burst size;
- the maximum E2E delay requirement D_f.

The arrival curve $A_f(t)$ and the maximum delay D_f are typically given by client applications to the controller. For each accepted flow f, the control plane computes a shaping parameter b'_f and a routing path p in the graph G, going from the source to the destination of f. The sum $\lceil \frac{b_f}{b'_f} \rceil T + D_{\text{worst}} + \sum_{e \in p} d_e$ (respectively, the sum of the shaping delay, the worst case queuing delay in Eq. (2) from Sec. IV-D, and the propagation delay) must be smaller than the maximum E2E delay D_f given as input. As we have the upper bound delay l_e for each link, this inequality can be rewritten as $\lceil \frac{b_f}{b'_f} \rceil T + \sum l_e + T \leq D_f$ and considered as a constraint on top of the path-based formulation presented after. The goal of the admission control algorithm is to maximize the sum of the arrival rates of the accepted flows, i.e., $\sum_{f | f \text{ accepted}} r_f$, since this is equivalent to maximizing the total throughput of the network. Alternative objectives, out of scope for this paper, could ensure a good load balancing of flows into the network or enforce a more resilient routing in case of failures.

B. Mathematical model

For each flow $f \in \mathcal{F}$, we denote as $\mathcal{P}_f$ the set of feasible paths, corresponding to the paths between the source and the destination of f with an end-to-end delay satisfying its E2E delay constraint on path p. Given a flow $f \in \mathcal{F}$ and a path $p \in \mathcal{P}_f$, the shaping parameter $b'(f,p)$ considered is the smallest one meeting the E2E delay and arrival rate constraints of the flow, $b'(f,p) = \min\{b' | \lceil \frac{b_f}{b'} \rceil T + T + \sum_{e \in p} l_e \leq D_f \& b' \geq r_f\}$. If $b'(f,p)$ is larger than the minimum available capacity on path p, the path is removed from $\mathcal{P}_f$. Another path must be generated to accommodate flow f. For $f \in \mathcal{F}, p \in \mathcal{P}_f$, the variable $\mathbf{z_{f,p}} \in \{0,1\}$ equals 1 iff flow f is selected, and routed on path p.

The optimization problem can be formulated as follows.

$$\max \sum_{f \in \mathcal{F}} \sum_{p \in \mathcal{P}_f} r_f \mathbf{z_{f,p}}$$

$$\sum_{p \in \mathcal{P}_f} \mathbf{z_{f,p}} \leq 1 \qquad \forall f \in \mathcal{F}, \quad (5)$$

$$\sum_{f \in \mathcal{F}} \sum_{p \in \mathcal{P}_f | e \in p} b'(f,p) \mathbf{z_{f,p}} \leq C_e T \qquad \forall e \in E, \quad (6)$$

$$\mathbf{z_{f,p}} \in \{0,1\}$$

Inequalities (5) ensure that at most one path is associated to each flow, while inequalities (6) ensure that the capacity required by the flows passing through a link e does not exceed the amount of data that can be transmitted over a cycle, i.e., $C_e T$. The delay and rate constraints are ensured by the set of variables introduced in the model. The difficulty of this model lies in the large set of paths (potentially exponential) to be generated to solve the problem. If $b'_f = r_f T$ and $D_f = \infty$ for all $f \in \mathcal{F}$, the delay constraints disappear and the only possible shaping parameter is b_f. Thus, the problem reduces to a multi-commodity flow, which is strongly NP-hard [22].

C. Column Generation with Randomized Rounding (CGRR)

Algorithm 1 Column Generation and Randomized Rounding

Require: Set of flows $\mathcal{F}$, graph G, capacities C_e for $e \in E$.
  calculate solution of the RMP and dual variables values
  while Columns added last loop iteration or first iteration **do**
    Calculate solution of the RMP and dual variables values
    for all flow $f \in \mathcal{F}$ **do**
      Path $p^* \leftarrow \emptyset$.
      Solve pricing problem for f
      if path found such that $u_f^F + \sum_{e \in p^*} b'(f,p^*) u_e^E < r_{f^*}$ **then**
        add the variable z_{f,p^*} to the RMP
      end if
    end for
  end while
  base solution $\leftarrow \{(f,p) | z_{f,p} = 1\}$, best solution $\leftarrow$ base solution
  while max number of rounding steps not reached **do**
    current solution $\leftarrow$ base solution
    for $i \in \{0, .., N\}$ where $N = |\{(f,p) | 0 < z_{f,p} < 1\}|$ **do**
      $\mathbb{P}(f,p) \leftarrow \frac{z_{f,p}}{\sum_{f,p | z_{f,p} \neq 0} z_{f,p}}$ we add weights to each fractional variable
      $(f^i, p^i) \leftarrow$ random couple (f,p) such that $0 < z_{f,p} < 1$
      if (f^i, p^i) can be added to the current solution **then**
        current solution = current solution $\cup \{(f^i, p^i)\}$.
      end if
    end for
    update best solution found by rounding
  end while
  return Best Solution

A classical approach for solving this problem is the Column Generation (CG) algorithm [23]. The basic idea behind this method is to consider only a subset of "useful" relaxed variables (i.e., without integrality constraint), forming a basis, referred to as Restricted Master Problem (RMP) or "primal problem", that can be solved using generic linear solvers. For each flow $f \in \mathcal{F}$, a new variable $\mathbf{z_{f,p}}$, i.e., new column of the basis, is computed in the pricing phase via a constrained shortest path in a graph with dual costs on the links. The RMP and the pricing are iteratively solved until the pricing cannot find any new variable improving the RMP. At this point, the CG routine ends as the relaxed solution is optimal. The overall CGRR algorithm is presented in Alg. 1.

When the CG has converged to an optimal solution, the variables are fractional and need to be rounded to integers. For a fast resolution of this problem, we leverage on the Randomized Rounding (RR) algorithm. The variables equal to 1 at the end of the RMP are added to the solution as already integer. The other variables with a fractional value have a probability of being picked proportional to this value. The RR has the advantage of being easily parallelized, improving the efficiency of the research of solution.

Solving the pricing problem consists in finding a variable which can improve the objective [23]. Each constraint of the primal problem is associated with a variable of the dual problem and each variable of the primal problem is associated to a constraint of the dual problem. Thus, by solving the RMP over a subset of variables and looking for additional variables that can improve the objective, it is possible to identify the violated constraints in the following dual problem:

$$\min \sum_{f \in \mathcal{F}} \mathbf{u_f^F} + \sum_{e \in E} C_e T \mathbf{u_e^E}$$
$$\mathbf{z_{f,p}} : \mathbf{u_f^F} + \sum_{e \in E | e \in p} b'(f,p) \mathbf{u_e^E} \geq r_f \quad \forall f \in \mathcal{F}, \forall p \in \mathcal{P}_f$$

where $\mathbf{u_f^F} \geq 0$ are the dual variable associated with each flow $f \in \mathcal{F}$ and $\mathbf{u_e^E} \geq 0$ is the dual variable associated with each capacity constraint over $e \in E$.

In order to improve the RMP objective and identify the violated dual contraints, we must look for a flow $f^* \in \mathcal{F}$ and a path $p^* \in \mathcal{P}_f$ such that

$$\mathbf{u_{f^*}^F} + \sum_{e \in p^*} b'(f^*, p^*) \mathbf{u_e^E} < r_{f^*}$$

This problem can be solved as follows: for each flow f, the set of possible shaping parameters is $\mathcal{B}_f = \{r_f T, .., b_f\}$ (see Sec. IV). For each shaping parameter $b' \in \mathcal{B}_f$, we solve a Constrained Shortest Path (CSP) over the graph G, trying to minimize $\sum_{e \in p^*} \mathbf{u_e^E}$. The cost of each link in this CSP is given by the dual variables and the constraint corresponds to the delay constraint. Only the arcs with enough capacity for the shaping parameter can be used. We use the LARAC algorithm [24] for an efficient CSP implementation. At each iteration of the pricing problem at most one variable per flow, i.e., the one with the smallest shaping parameter, is added.

TABLE I
TRAFFIC PATTERNS IN THE POC TEST.

Parameter	HP flows	BE flows
Rate (Mbps)	600	600
Burst size (KB)	1.5	192

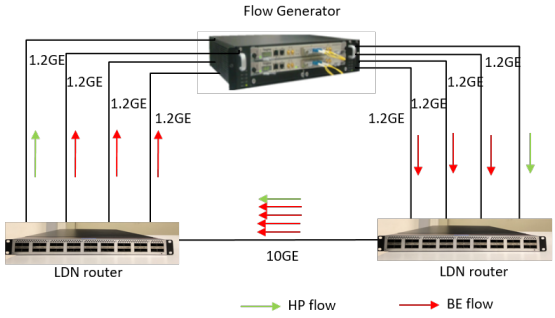


Fig. 5. Test topology for the PoC implementation.

VI. SIMULATION RESULTS

We now verify with a PoC implementation and simulations that our LDN architecture can guarantee the E2E delay and bounded jitter. We also show that our control plane algorithm CGRR can efficiently maximize the accepted traffic.

A. Proof-of-Concept implementation

We have implemented this LDN architecture on a mainstream commodity router of Huawei. All functionalities including shaping, traffic classification, label swapping, gate-control queues have been developed. The duration of a cycle was set to $T = 10\mu s$ and the test topology is shown in Fig. 5. A flow generator is used to generate flows and sample E2E latency of packets. We considered 1 HP flow and 3 BE flows with the characteristics given in Table I. The experimental results are presented in Fig. 6, where the red line and blue line represent the E2E latency upper and lower bounds, respectively. The largest value of the worst case E2E queuing delay is $67.046\mu s$, while the smallest value of the best case delay is $49.886\mu s$. By consequence, the largest jitter experienced by the flows is $67.046 - 49.886 = 17.16\mu s$, smaller than $2T$ as expected from the results in Sec. IV-D.

In Fig. 6, we also show results of the case where routers do not have LDN capabilities, in the same experimental setting. As no performance can be guaranteed, the traffic in this case is considered as BE. The green line and yellow line represent the worst and best delays for the HP flows, respectively. We can see that, the latency upper bound is nearly $980\mu s$ and lower bound is $19\mu s$, with a resulting jitter of $961\mu s$. This is because common routers forward packets as BE traffic and the bursts of background flows have a larger impact on the latency of the HP flow. These experimental results have shown that LDN can provide guaranteed latency and jitter in practice.

B. Simulation setup

In order to assess the scalability of the LDN architecture, we use OMNeT++ to simulate its performance within a real ISP topology (AS 1239 of Sprint), composed of 315 routers and

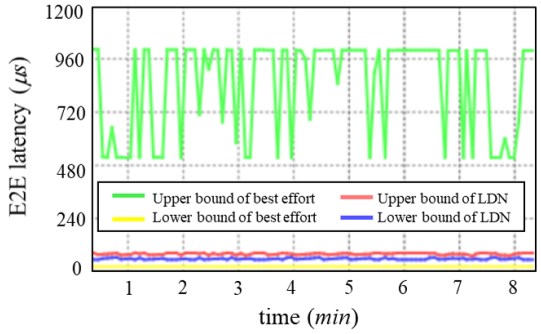


Fig. 6. Test results over the PoC implementation.

1944 unidirectional links, each with capacity $C_e = 10Gb$ [25]. Each router is directly connected to a host. The cycle duration T, in this case, is set to $T = 25\mu s$. We consider 22062 randomly generated flows between different pairs of hosts. The path length distribution of flows is presented in Table VI-B. Each flow sends one packet at a random time during a cycle. Three classes of flows are considered: (i) 8997 flows have packets with fixed $MPS = 64$ Bytes, (ii) 4342 flows have packets with fixed $MPS = 1500$ Bytes, and (iii) the remaining 8723 flows generate packets with MPS randomly generated in the interval $[64, 1500]$ Bytes. The routing is chosen using the OSPF shortest path.

The simulated E2E queuing latency is shown in Fig. 7(a). We can find that the measured E2E queuing latency lies between the D_{worst} and D_{best} bounds presented in Sec. IV-D. We point out that the Max and Min values in Fig. 7(a) represent the worst and best delay of all the flows with the same path length in the simulation. For the sake of comparison, in the same graph we also show the theoretical upper and lower bounds Φ_{up} and Φ_{low} introduced in Sec. IV-D. We can see that the measured latency is indeed in between these bounds. We also calculate the jitter of each flow independently and depict the statistics for all flows in Fig. 7 (b). We can see that the jitter of each single flow is less than the theoretical upper bound ($50\mu s$) and the average jitter of all flows with different hops is much smaller than the upper bound. For example, for the 7-hop flows, the lower jitter bound is 0 and average jitter among all the flows is $0.277\mu s$. Besides, the upper jitter bound of the majority of flows is also very small (smaller than $0.3\mu s$), while only 11 flows are with an upper bound above $3\mu s$. The corresponding Cumulative Distribution Function (CDF) for the jitter is shown in Fig. 7(c).

C. Admission control

In order to evaluate the efficiency of the proposed CGRR admission control algorithm, we produce numerical results considering the same instance as in Sec. VI-B, i.e., the same network, propagation delays, demand set, and MPS distribution. The maximum burst size b_f for each flow is randomly chosen between 1 and 10 MPSs over one cycle, and the arrival rates r_f are chosen between 256kb/s and 60 Mb/s, both with a uniform distribution. The maximum E2E delays for each demand is set to the same value, varying this value at each trial between $100\mu s$ and $1000\mu s$. Similarly, all the links have the same capacity, fixed, at each trial, to a value between 20 and 100 Gb/s. In order to get a realistic evaluation of the E2E delay, a random shift between cycles due to asynchronism between the nodes is also added. As a benchmark of the proposed CGRR admission control algorithm, we consider a Shortest Path First (SPF) routing algorithm using as link costs standard OSPF metrics, i.e. $10^8/C_e$. Differently from CGRR, where the flows of the batch are processed together, in OSPF routing, we consider one flow at a time. For each flow, once the path is computed, we choose the smallest shaping parameter that can verify the delay and arrival rate constraints. If the capacity constraint is respected on each link, the flow is accepted.

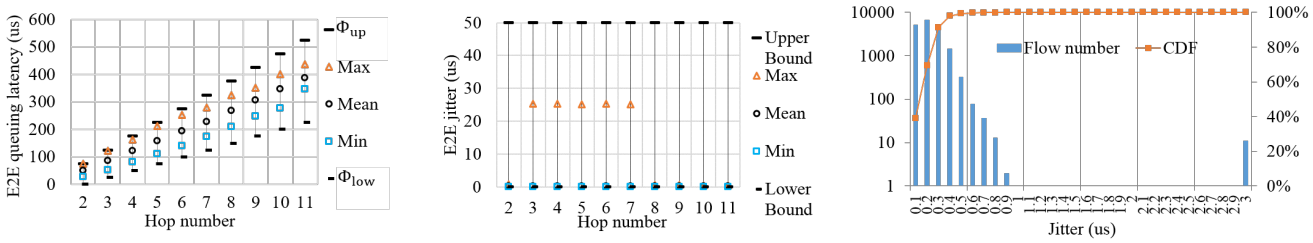
Fig. 8(a) shows the proportion of traffic accepted into the network as a function of the tightness of the delay constraint and the link capacity. The tighter the delay constraint, the lower the accepted traffic, since for many demands it is no longer possible to compute a feasible path. Even with a loose delay constraint and a large capacity, the OSPF routing fails to accept all demands. Vice versa, the CGRR routing is quickly able to accept all the traffic into the network due to a better distribution of traffic and a better choice of the shaping parameter. With a bandwidth of 40 Gb/s and a realistic max E2E delay constraint of 400 μs, all the LDN traffic can be accepted into the network. The gap in terms of accepted traffic between CGRR and OSPF, defined as $\frac{\text{CGRR - OSPF}}{\text{CGRR}}$, is presented in Fig. 8(b). When the link capacity is small, the gap gets higher, up to 40%, since the CGRR is able to carry out better decisions in terms of routing and shaping. This gap reduces as the OSPF algorithm manages to accept more demands, i.e., for larger values of the delay constraints and link capacities. However, the CGRR is slower than OSPF, as shown in Fig. 8(c). In the worst case, CGRR takes up to 80s to solve the hardest instances, i.e., those with the smallest capacity. On the other hand, OSPF runs in at most a few seconds, no matter the link capacity or the E2E delay constraint.

VII. Conclusion

This paper presented a realistic implementation of a Large-scale Deterministic Network (LDNs) to provide deterministic E2E latency and bounded jitter in large-scale IP networks. At the data plane, high scalability is achieved considering an asynchronous cyclic scheduling with 3 queues and a low complexity forwarding in $O(1)$. To ensure deterministic performance, the traffic is shaped at I-GWs using gate-control queues that allow achieving a granularity of 1.5kB, improving traditional methods. Detailed analysis shows that LDN can guarantee a bounded E2E queuing delay of $(2h-1)T$ for an h-hop flow and a $2T$ jitter bound. We also presented CGRR, a control plane algorithm for admission control to maximize the accepted traffic by computing the best feasible shaping parameters and paths for a batch of incoming flows. Besides, we also verified the performance of LDN through a PoC implementation and simulations. We showed that the proposed CGRR algorithm outperforms traditional OSPF routing by up

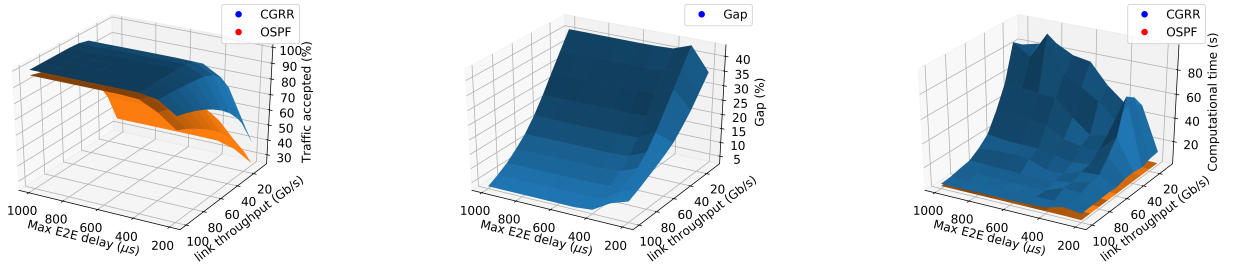
TABLE II
FLOW DISTRIBUTION OVER PATHS.

Path length (hops)	2	3	4	5	6	7	8	9	10	11
Number of flows	1239	4824	6953	5776	2156	558	295	162	88	11



(a) E2E queuing latency (μs). (b) E2E jitter (μs). (c) Jitter distribution among all flows.

Fig. 7. Performance results for a large-scale deterministic IP network simulated with OMNet++.



(a) Accepted traffic (%). (b) Gap between CGRR and OSPF. (c) Computational time of CGRR and OSPF (s).

Fig. 8. Admission control results for the CGRR algorithm and OSPF routing.

to 40% in term of traffic acceptance. Our results confirm that LDN is suitable large-scale IP networks.

REFERENCES

[1] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5g: Ran, core network and caching solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, 2018.

[2] E. Grossman, "Deterministic Networking Use Cases," RFC 8578, May 2019. [Online]. Available: https://rfc-editor.org/rfc/rfc8578.txt

[3] R. Li, "Towards a new internet for the year 2030 and beyond," *Proceedings 3rd Annual ITU IMT-2020/5G Workshop Demo Day*, 07 2018.

[4] A. Nasrallah, V. Balasubramanian, A. S. Thyagaturu, M. Reisslein, and H. Elbakoury, "Cyclic queuing and forwarding for large scale deterministic networks: A survey," *ArXiv*, vol. abs/1905.08478, 2019.

[5] "IEEE Standard for Local and metropolitan area networks: Cyclic Queuing and Forwarding," *IEEE 802.1Qch-2017*, pp. 1–30, June 2017.

[6] "Deterministic Networking Architecture," RFC 8655, Oct. 2019.

[7] L. Qiang, X. Geng, B. Liu, T. Eckert, L. Geng, and G. Li, "Large-Scale Deterministic IP Network," IETF Draft draft-qiang-detnet-large-scale-detnet-05, Sep. 2019.

[8] M. Chen, X. Geng, and Z. Li, "Segment Routing (SR) Based Bounded Latency," Internet Engineering Task Force, Internet-Draft draft-chen-detnet-sr-based-bounded-latency-00, Oct. 2018.

[9] J. Krolikowski, S. Martin, P. Medagliani, J. Leguay, S. Chen, X. Chang, and X. Geng, "Joint routing and scheduling for large-scale deterministic ip networks," *Elsevier Computer Communication*, 2020.

[10] J. Walrand and P. P. Varaiya, *High-performance communication networks*. Morgan Kaufmann, 2000.

[11] H. Shimonishi, M. Yoshida, R. Fan, and H. Suzuki, "An improvement of weighted round robin cell scheduling in atm networks," in *Proc. IEEE GLOBECOM*, 1997.

[12] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on networking*, vol. 4, no. 3, pp. 375–385, 1996.

[13] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM transactions on networking*, vol. 1, no. 3, pp. 344–357, 1993.

[14] J. C. Bennett and H. Zhang, "WF/sup 2/Q: worst-case fair weighted fair queueing," in *Proc. IEEE INFOCOM*, 1996.

[15] M. Fidler, "Survey of deterministic and stochastic service curve models in the network calculus," *IEEE Communications surveys & tutorials*, vol. 12, no. 1, pp. 59–86, 2010.

[16] I. . W. Group, "IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks," Tech. Rep., 2018.

[17] R. T. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification," RFC 2205, Sep. 1997.

[18] J.-Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. Springer Science & Business Media, 2001, vol. 2050.

[19] https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/ixp42x-product-line-network-processors-datasheet.pdf.

[20] https://www.mellanox.com/related-docs/prod_multi_core/PB_TILE-Gx36.pdf.

[21] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 375–385, 1996.

[22] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *Symposium on Foundations of Computer Science*, 1975.

[23] W. E. Wilhelm, "A technical review of column generation in integer programming," *Optimization and Engineering*, vol. 2, no. 2, pp. 159–200, 2001.

[24] G. Y. Handler and I. Zang, "A dual algorithm for the constrained shortest path problem," *Networks*, vol. 10, no. 4, pp. 293–309, 1980.

[25] https://research.cs.washington.edu/networking/rocketfuel/.