Exploiting Queue Information for Scalable Delay-Constrained Routing in Deterministic Networks

Jakob Miserez, Didier Colle, *Member, IEEE*, Mario Pickavet, *Senior Member, IEEE*, and Wouter Tavernier

***Abstract*—Next-generation Internet will require strict end-to-end delay guarantees to support upcoming latency-sensitive applications. The IEEE 802.1 Time-Sensitive Networking (TSN) standard has become the de-facto solution for Ethernet-based L2 networks to support applications with strict latency, jitter and packet loss requirements. The IETF DetNet Working Group tries to expand on TSN to support real-time applications over larger-scale L3 networks. This paper proposes control and routing strategies that provide latency guarantees in L3 networks without requiring time synchronization among nodes. The proposed strategies include a link-state routing protocol and several exploration-based protocols that exploit queue-level information and network calculus to provide latency guarantees. Additionally, the use of queueing delay budgets enables independence among flows, while enabling fine-grained routing. This allows to make better routing decisions and to support applications with diverse latency requirements. Moreover, traffic shaping is only required at the network ingress. The strategies are evaluated extensively and compared in a simulation environment in multiple large-scale scenarios, considering acceptance rate, network utilization, path dissemination time, control overhead, and memory consumption, as well as how these metrics evolve w.r.t. different network scales. Experimental results demonstrate that representative delay-constrained traffic demands can be accommodated adequately by queue-level link-state routing protocols only in smaller-scale networks. In larger-scale network scenarios, breadth-first exploration-based protocols are required to provide stable performance w.r.t. acceptance rate and path dissemination times at the cost of only linearly increasing control overhead and memory footprint.**

***Index Terms*—Deterministic networking, DetNet, time-sensitive networking, TSN, priority queueing, network calculus.**

Manuscript received 4 December 2023; revised 8 May 2024; accepted 22 July 2024. Date of publication 29 July 2024; date of current version 16 October 2024. This research is partially funded by the Flemish FWO SBO S003921N VERI-END.com (Verifiable and elastic end-to-end communication infrastructures for private professional environments) project, and by the European Commission through the Horizon Europe/JU SNS project Hexa-X-II (Grant Agreement no. 101095759). The associate editor coordinating the review of this article and approving it for publication was M. Tornatore. *(Corresponding author: Jakob Miserez.)*

The authors are with the IDLab and the Department of Information Technology, Ghent University - imec, 9052 Ghent, Belgium (e-mail: jakob.miserez@ugent.be; didier.colle@ugent.be; mario.pickavet@ugent.be; wouter.tavernier@ugent.be).

Digital Object Identifier 10.1109/TNSM.2024.3435769

I. Introduction

PROVIDING stringent latency and reliability guarantees has become paramount for future communication networks in order to support next-generation applications such as critical healthcare applications, autonomous automotive vehicles, and industrial automation [1]. The Time-Sensitive Networking (TSN) standard emerged to support such safety-critical, real-time applications in Ethernet-based LANs. Time-Aware Shaping (TAS) and Cyclic Queueing and Forwarding (CQF) are fundamental TSN mechanisms in support of achieving low-latency, low-jitter and zero packet loss. The common denominator between these mechanisms is their reliance on highly-accurate time synchronization between TSN switches, which can be achieved in small and highly-controllable L2 networks. However, applications such as smart energy grids, building automation systems, and industrial machine to machine (M2M) require larger L3 networks [2], and – due to its reliance on precise time synchronization – extending TSN beyond the LAN is not at all straightforward. Choosing the right control architecture is a pivotal decision, and the difficulty in guaranteeing precise time synchronization necessitates the exploration of alternative data planes, beyond TAS and CQF. Moreover, scaling up deterministic networks introduces challenges such as efficiently disseminating and managing routing information, computing optimal paths efficiently, efficient signaling, and configuring the network for maximum efficiency. This paper focuses on addressing these challenges, while also providing a comprehensive evaluation. The proposed solution consists of novel scalable routing protocols that exploit strict priority queueing (SPQ) and network calculus (NC) to bound end-to-end delay in dynamic large-scale deterministic networks. The use of queueing delay budgets enables independence between flows and fine-grained routing. This introduces a new level of flexibility to support diverse traffic requirements and it allows the system to make better routing decisions under contention. Moreover, no time synchronization or new hardware is needed, and traffic shaping is only required at the ingress of the network. Two distinct control architecture models are proposed and evaluated thoroughly: a link-state (LS) routing protocol inspired by OSPF-TE, and exploration-based routing protocols. The analysis and evaluation of these architectures aim to provide valuable insights into the practical design considerations for deterministic networking

in large-scale networks. This paper improves and extends [3], providing a more in-depth overview of its positioning within TSN and DetNet, a deeper analysis of the control architectures and foundations, and exhaustive evaluation under different regimes, focusing on routing performance, scalability aspects and assessing resource requirements. The remainder of this paper is organized as follows. Section II provides an overview of related work. Sections III and IV lay out the data plane and control plane architectures of the proposed DetNet solutions, which includes a detailed overview of how the latency guarantees are provided. Then, Section V explains the experimental setup and provides a detailed discussion of the simulation results, after which this paper is concluded in Section VI.

II. Related Work

Time-Sensitive Networking (TSN) is a set of standards that are designed to ensure reliable transmission and predictable end-to-end latency over switched Ethernet networks [1]. TSN is standardized in the IEEE 802.1 TSN Task Group and includes a wide range of features such as time synchronization, traffic shaping, filtering and policing, and path selection and network traffic scheduling algorithms. IEEE 802.1Qbv Time-Aware Shaping (TAS) relies on controlled gates to explicitly open and block queues for transmission on periodic timescales. TAS is one of the primary technologies to bound the queueing delay of scheduled (time-triggered) traffic, in the presence of other traffic. TAS heavily relies on precise time synchronization between nodes, which is provided through the IEEE 802.1AS generalized Precision Time Protocol (gPTP) [4]. A variation on TAS is IEEE 802.1Qch Cyclic Queueing and Forwarding (CQF). With CQF, TSN switches synchronize their frame transmissions in a cyclic manner, enabling deterministic communication. A significant amount of effort has been put in researching routing and scheduling mechanisms for TSN [5]. Control architectures in TSN are typically inspired by software-defined networking (SDN), which involves a centralized network controller (CNC) that manages all switches (and potentially the end devices) in the network. While this architecture works well in small-scale L2 networks, this design has scalability issues [6], and, with the CNC being a single point of failure, it is also subject to security risks. However, TSN also specifies protocols that allow to establish a path and reserve resources along that path without a CNC, such as the Stream Reservation Protocol (SRP) [7]. SRP is essentially a signaling protocol along a spanning tree that allows device registrations and resource reservations. This means that when a flow is requested, the paths to the destinations are already decided by the spanning tree, and SRP will merely reserve resources along the way. SRP is therefore not a routing protocol that can make advanced routing decisions. The Resource Allocation Protocol (RAP) [8] is being developed as a scalable successor of SRP, while also supporting more TSN features.

The IETF Deterministic Networking (DetNet) Working Group (WG) focuses on data paths over L2 bridged and L3 routed segments, providing deterministic bounds on latency, jitter, and packet loss. The DetNet WG specifically addresses L3 aspects, while also collaborating with the IEEE 802.1 TSN Task Group to define a common architecture for both L2 and L3. Three enablers have been specified for deterministic networking in L3 networks: service protection, resource reservation, and explicit routing [9]. While service protection deals with network failures, resource reservation and explicit routing address two of the DetNet requirements: bounded latency and packet loss. Proper resource reservation ensures that no contention-related packet losses can occur, and explicit routing makes sure that flows do not suffer from temporary interruptions of converging routing protocols.

Multi-Protocol Label Switching Traffic Engineering (MPLS-TE) [10] is proposed as a candidate to support explicit routing [11]. MPLS [12] enables IP packets to be routed over predefined paths, and forwarded using a label that is added to the packet, instead of the destination IP address. This flexibility makes MPLS very useful and efficient for explicit routing in DetNet context, as flows can simply be routed along different paths by using different labels. Besides MPLS, other techniques can also be used to support explicit routing, including IP-based forwarding [13] and source routing [14]. The Resource Reservation Protocol - Traffic Engineering (RSVP-TE) [15] can be used in combination with MPLS to setup explicit label-switched paths and reserve resources (e.g., bandwidth) along the path. The proposed DetNet control architectures for the combination of MPLS and RSVP-TE (and similar setups) use a path computation element (PCE) that computes the explicit path for the flow and initiates the resource reservation using RSVP-TE [16]. This path computation element can be the first node of the flow path, a CNC, or any other network element. Without a CNC, the PCE typically requires to maintain a global view of the network, including the available resources at every node. For this purpose, a routing protocol with traffic engineering capabilities like OSPF-TE [17] can be leveraged. Since L3 networks are deployed on a larger scale, they need mechanisms that require less precise time synchronization, or even no time synchronization at all [18]. This has given rise to a wide range of novel data planes that fit these requirements, such as Tagged Cyclic and Queueing [19] and Stateless Fair Queueing [20]. The general bounded queueing latency model used by these data planes is proposed in [21], and it is based on network calculus principles.

Network calculus (NC) [22] is a set of mathematical results that allows to compute worst-case delay and buffer bounds in computer networks. In the context of TSN, NC is one of the main tools to analyze or bound end-to-end queueing delays. The work in [23] provides analysis for multiple AVB traffic classes in TSN. The authors of [24] analyze CBS+SPQ TSN networks in the context of industrial automation. A routing and scheduling method for integrating scheduled and AVB traffic is given in [25]. In [26], NC is used to derive minimum bandwidth reservations in CBS-based TSN networks, given a static set of flows and fixed routes. NC has also been used to bound queueing delays (and therefore end-to-end latency) in datacenter environments. In [27], NC is applied in an offline VM placement algorithm in order to guarantee bounded end-to-end delay between VMs, assuming the network nodes

use first-in first-out (FIFO) packet scheduling. The work in [28] builds further on this idea by exploiting NC and SPQ in an SDN-based solution called Chameleon. The main idea of this work is that packets experience vastly different queueing delays, depending on the priority they are assigned at a node. This observation leads to a network abstraction called the queue-level topology, which replaces a unidirectional link by several new links, one for every queue. By assigning queueing delay budgets to every queue and bounding the queueing delays using NC, every queue-level topology link has a bounded delay, therefore bounding the end-to-end delay of a path. Since the protocols proposed in this paper reuse the same foundations, these principles are explained in detail in Section III.

Song et al. [29] proposed a fully distributed routing protocol that allowed to find multi-constrained paths for Quality-of-Service (QoS) by using a minimum of local state only, which served as the main inspiration for the exploration-based protocols proposed in this paper. The idea is to send exploration packets (probes) into the network. When a router receives a probe, it will first perform a link forwarding test to see which links have enough resources (e.g., bandwidth) left, and then forward a probe in parallel through all links that pass the test, while simultaneously reserving resources. This process goes on until a probe reaches the destination. Besides probes, other packets are also used for signaling success (confirmations) and failure (prunes). An optimization of the signaling mechanism (fast-prune) is also proposed to avoid over-reservation.

Although delay-constrained routing using NC is a well-researched topic, the use of queueing delay budgets in this paper enables independence between flows, and provides additional levels of flexibility to support diverse application requirements and it enables fine-grained QoS routing, while not relying on time synchronization or new hardware. The proposed routing protocols can be deployed in large-scale networks, supporting diverse and dynamic traffic flows. Moreover, while many data plane and control plane solutions have been proposed in DetNet context, little of this work is actually thoroughly evaluated in large-scale scenarios. The focus of this paper is therefore to provide DetNet solutions based on priority queueing and network calculus, and to evaluate these solutions thoroughly in large-scale scenarios.

III. Data Plane Architecture

A. Strict Priority Queueing (SPQ)

With priority queueing, every outgoing packet is classified into a certain priority. The packet then waits to be transmitted at a queue corresponding to that priority. In the SPQ regime, packets in the highest priority queue are always transmitted before any lower priority queues are serviced. Once the highest priority queue is empty, the next lower priority queue is serviced, and so on. Lower priority queues can therefore experience high queueing delays if higher priority queues are consistently busy. This form of packet scheduling is called 'strict' since it will always respect the priority hierarchy, potentially starving the lower priorities. A packet scheduling method like weighted round-robin also makes use of priority queues, but takes a proportional approach in distributing bandwidth among different queues. This is in sharp contrast with the immediate prioritization that occurs with SPQ.

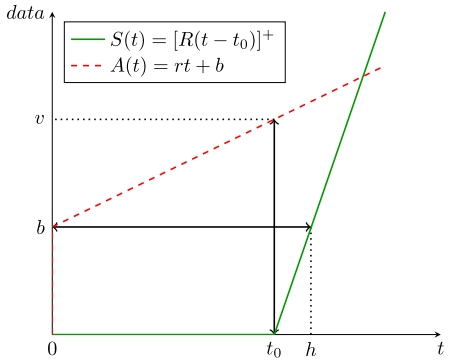


Fig. 1. Example rate-latency service curve and affine arrival curve. The worst-case queueing delay is given by the horizontal deviation h, while the buffer space requirement is given by the vertical deviation v.

B. Network Calculus (NC)

This section provides a short overview of the NC principles that are used in this work to enable deterministic networking. Using NC, bounds can be computed if an upper bound on the incoming traffic and a lower bound of the network (link) service are provided. Arrival curves model the former, while service curves model the latter. A common type of arrival curve, which is assumed in this paper, is the affine (leaky-bucket) arrival curve, characterized by a worst-case burst b, and a rate r:

$$\alpha_{b,r}(t) = \begin{cases} rt + b & \text{if } \ t > 0 \\ 0 & \text{otherwise} \end{cases}$$

The arrival curve constraint means that the amount of data sent by the flow during a time window of τ is always limited by $\alpha(\tau)$. If two traffic flows are bound by arrival curves $\alpha_1(t)$ and $\alpha_2(t)$ respectively, then their aggregate arrival curve $A(t) = \alpha_1(t) + \alpha_2(t)$ provides an upper bound on the aggregate traffic flow. For link service, the rate-latency service curve is used in the context of SPQ:

$$\beta_{R,t_0}(t) = [R(t - t_0)]^+ = \begin{cases} R(t - t_0) & \text{if } \ t > t_0 \\ 0 & \text{otherwise} \end{cases}$$

This type of service curve provides service (i.e., puts bits on the link) at a rate of R, but only after t_0 amount of time has passed. Bounds on both queueing delay and buffer space can be computed using these curves: the worst-case queueing delay corresponds with the horizontal deviation h between the aggregated arrival and service curve, while the worst-case buffer space requirement is given by the vertical deviation v. This is shown graphically in Fig. 1. With FIFO packet scheduling, the service curve of a link with bandwidth R is given by $\beta(t) = Rt$. In case of SPQ, the situation is more complicated. Consider a network link with a bandwidth of R bps that uses SPQ with three priorities, where the lowest priority queue (Q2) is used for best-effort traffic. Thus, it has a high priority queue (Q0), with aggregated arrival curve

$A_0(t)$, and a low priority queue (Q1), with aggregated arrival curve $A_1(t)$, for critical traffic. In case of non-preemptive priority queueing, Q0 should, in the worst-case, only wait on a single packet of the lower priority queues to complete its transmission. If the largest packet size in those queues is l, then the rate-latency service curve $\beta_0(t)$ for Q0 is given by

$$\beta_0(t) = [Rt - l]^+ \tag{1}$$

Traffic in Q1 must wait until all higher priority queues are empty, which means it can only receive service (in the worst-case) after Q0 has service (link rate) 'to spare' for it to be used. This is why the service curve $\beta_1(t)$ for Q1 is given by

$$\beta_1(t) = [Rt - A_0(t) - l]^+ \tag{2}$$

Note that l is still present in $\beta_1(t)$ to account for non-preemptive priority queueing, given that Q2 is used for best-effort traffic. Both Eq. (1) and (2) can be generalized to an arbitrary amount of queues. Bounds in Q0 can be computed using $A_0(t)$ and $\beta_0(t)$. For Q1, $A_1(t)$ and $\beta_1(t)$ should be used. Importantly, the arrival curve of a traffic flow changes when it goes through a priority queue (without additional shaping). This is because if packets are waiting before being transmitted, then new incoming data is accumulated in the worst-case. The most practical way to derive the new arrival curve $\alpha'(t)$ is as follows, with D the maximum queueing delay.

$$\alpha'(t) = \alpha(t + D) \tag{3}$$

For an affine arrival curve $\alpha_{b,r}(t)$, this results in an increase of the worst-case burst.

$$\alpha'(t) = \alpha_{b+rD,r}(t) \tag{4}$$

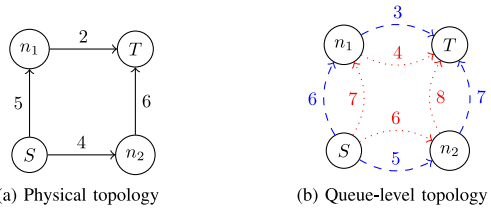
Additional details on NC can be found in [22], [30].

C. Queueing Delay Budgets

The computed worst-case end-to-end latency should be independent of other flows that might be added or deleted in the future [21]. To this end, every queue of priority q is assigned a fixed queueing delay budget Δ_q. The delay budgets are network-wide parameters that provide an upper bound on the maximum queueing delay that any flow can experience through a queue of that priority. In other words, the horizontal deviation in a queue cannot go beyond its pre-configured queueing delay budget. However, this must be explicitly enforced (cf. Section III-D). The use of queueing delay budgets naturally results in the queue-level topology. This is a network abstraction that explicitly accounts for the worst-case queueing delays, where a directed (physical) link is replaced by multiple directed links, one for every priority. Given a network topology $G = (N, E)$ where $N \subset \mathbb{N}$ is a finite set of nodes and E is a set of directed edges e_{ij} with $(i, j) \in N \times N \wedge i \neq j$. Every edge e_{ij} has a propagation delay denoted as $d_{e_{ij}}$. This basic topology is transformed to the queue-level topology $G' = (N, E')$ with E' a set of queue-level topology edges/links. E is transformed to E' by duplicating every edge Q times, with Q being the number of priority queues for real-time traffic (e.g., 8). This results in edges $e_{ij}^0, e_{ij}^1 \dots e_{ij}^{Q-1}$ for a physical link e_{ij}.

TABLE I
EXAMPLE FLOW CHARACTERISTICS

Characteristic	Value
Name	f_1
Source	S
Destination	T
Max. end-to-end latency	10
Arrival curve	$\alpha_{f_1}(t)$



(a) Physical topology (b) Queue-level topology

Fig. 2. Sample toy network featuring two queues with queueing delay budgets of 1 (dashed, blue) and 2 (dotted, red). (a) Physical topology with propagation delays only. (b) Queue-level topology with worst-case delay per queue-level topology link, obtained by summing the propagation delay and its budget.

The worst-case delay of a queue-level topology link e_{ij}^q is given by $d_{e_{ij}^q} = \Delta_q + d_{e_{ij}}$. The worst-case end-to-end delay of a path is then given by the sum of the worst-case delays of the queue-level topology links on the path. This highlights the power of queueing delay budgets: the worst-case delay of a flow along a path depends only on the propagation delays and the queueing delay budgets of the chosen links, given that the budgets are respected. Therefore, previous routing decisions will never have to be revisited when flows come and go. Moreover, a single flow can make use of different priorities along the way, as long as the worst-case end-to-end delay is respected. This is a lot more flexible than, e.g., DiffServ, where every flow is assigned a fixed priority on beforehand. A simple physical network, which utilizes 2 priority queues for real-time traffic and delay budgets of 1 and 2, and its corresponding queue-level topology is given in Fig. 2. An example flow in this network is given in Tab. I. As an example, the physical link e_{S,n_1} with propagation delay 5 is transformed to the queue-level topology links e_{S,n_1}^0 and e_{S,n_1}^1 with worst-case delays of respectively $5 + 1 = 6$ and $5 + 2 = 7$. This example also highlights the larger search space between nodes: in the physical topology, there are 2 paths between S and T, while in the queue-level topology, there are 8 possible paths with a variety of worst-case delays.

D. Resource Reservation and Admission Control

Resource reservation ensures that no contention-related delay violations or packet losses occur. Thus, when a flow f with arrival curve $\alpha_f(t)$ is assigned to a queue e_{ij}^q, e.g., by a routing algorithm, then resources must be reserved for the flow at that queue. Resource reservation in this context consists of adding the arrival curve of the flow $\alpha_f(t)$ to the aggregated arrival curve in the assigned queue $A_{e_{ij}^q}(t)$, and updating the service curves in the lower queues. If, in the new configuration, the queueing delay budgets and buffer sizes are respected, then the flow can be admitted. Whenever a flow is admitted to a link e_{ij}^q, its arrival curve changes according to

Eq. (3), using Δ_q as the maximum queueing delay:

$$\alpha'_f(t) = \alpha_f(t + \Delta_q) \tag{5}$$

Note that this admission control system always makes sure that the queueing delay budgets are never violated. Thus, the worst-case delay of a path for a flow, which is computed based on the queueing delay budgets, will never be violated regardless of other traffic flows in the network. Indeed, this means that routing decisions in the past will never have to be revisited.

E. Control Channel

Using the principles of Section III-D, resources can also be reserved for control traffic on every physical link, i.e., the control traffic on every link is modeled by an arrival curve. By assigning those resources to Q0, control traffic has high priority such that the control protocols can operate reliably without interfering with data traffic. The exact modeling of the control traffic is beyond the scope of this work.

IV. Control Plane Architecture

A. Link-State Routing Protocol

The link-state (LS) routing protocol proposed in this section is heavily inspired by OSPF-TE. OSPF-TE specifies mechanisms to build an extended view of the network at every router for traffic engineering purposes through opaque Link-State Advertisements (LSAs) [17]. These LSAs contain Type-Length-Value (TLV) triplets describing, e.g., reserved bandwidth at a link. The LS protocol leverages these principles to flood LSAs that contain the latest queue information: QLSAs. A QLSA specifically contains the propagation delay and bandwidth of the physical link, and the aggregated arrival curves of the flows installed at every queue-level topology link that belong to that physical link. By flooding QLSAs, every router builds the queue-level topology, including all arrival and service curves. Therefore, upon a flow request, the source router acts as a PCE and computes a full path for the flow. After computing this path, resources are reserved at every node along the path through a signaling and resource reservation mechanism. The next paragraphs provide additional details on the protocol.

1) Distributing QLSAs: Ideally, every router has a perfectly accurate view of the network topology and the available resources. However, this requires very frequent updates, which is not scalable. Therefore, as proposed in [31], a hybrid system of event-based and time-based updates is used to distribute QLSAs. The idea is to make a trade-off between accuracy and efficiency. Generally, time-based updates provide a reasonably accurate view of the network with low overhead. However, when certain flows require lots of resources, that assumption is not valid anymore. Therefore, event-based updates are used to account for rapid and significant changes of the available resources. They are triggered if the allocated (or freed) bandwidth (given by the aggregated arrival curves) at a physical link since the last update surpasses a certain threshold.

2) Routing Algorithm: The routing algorithm to compute a path for a new flow is based on Dijkstra's shortest-path algorithm, run on the queue-level topology using the worst-case delays as link costs. The idea is to keep the worst-case burst increase low by finding minimum delay paths. However, the basic algorithm is slightly adjusted to ensure that the resulting path has sufficient resources. This is done by taking the burst increase at every hop into account and by incorporating an admission control mechanism (based on Section III-D) into the algorithm. This ensures that the resulting path has sufficient resources for the flow.

3) Signaling & Resource Reservation: After a path has been computed for a new flow f at the source, a scheme similar to RSVP-TE is used for signaling and resource reservation at every node on the path. The idea is that a reservation message travels from the source to the destination along the computed path, carrying the flow arrival curve $\alpha_f(t)$ and the computed queue-level topology path. At every hop, resources are reserved at a queue based on the computed path and $\alpha_f(t)$ (as described in Section III-D). Importantly, $\alpha_f(t)$ must be updated in the reservation message according to Eq. (5) before relaying it to the next hop on the path. Note that this update is already taken into account by the routing algorithm. At the next hops, the same steps should be taken until the destination is reached, before signaling a confirmation message back to the source. However, as discussed in Section IV-A1, the view that every router has of the global queue-level topology contains inaccuracies. Therefore, a router could compute a path for a flow that does not have sufficient resources in reality. If insufficient resources are detected at a node along the path, this is signaled back to the source router, which may opt to compute a new path and try again. The router with insufficient resources will also send out a new QLSA, such that a future inaccuracy failure is less likely to happen. This can also be seen as an event-based update.

4) Example: Given the example network in Fig. 2, the operation of the routing and signaling and resource reservation principles of the protocol can be seen in Fig. 3. Consider a flow request for f_1 (cf. Tab. I). Given that QLSAs are already distributed, the source node acts as PCE and therefore computes a full path on the queue-level topology. In this case, the resulting path is $(e^0_{S,n_1}, e^0_{n_1,T})$ using the blue dashed queue-level topology links, resulting in a worst-case end-to-end delay of $6 + 3 = 9$. This computed path is the one with the lowest delay following the Dijkstra-based routing algorithm (cf. Section IV-A2). Then follows the signaling and resource reservation, as described in Section IV-A3, where a reservation message '*rsv_msg*' travels from the source to the destination along the path and resources are reserved locally. Finally, a confirmation message '*confirm_msg*' is signaled back to the source, since no intermediate failures happened.

B. Exploration-Based Routing Protocols

While an LS routing protocol relies on a global view of the network and its available resources, exploration-based routing protocols take a radically different approach. The main idea is to actively explore the network by sending exploration packets upon a flow request. When a router receives such an exploration packet (probe), it tries to forward the probe to the destination using only information inside the probe packet

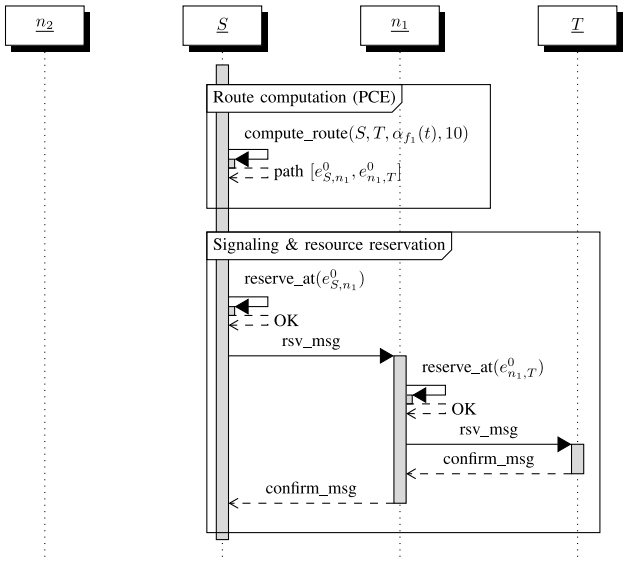


Fig. 3. Example link-state route computation and signaling upon a flow request for f_1 (cf. Tab. I), applied to the toy network in Fig. 2.

and router-local information. A forwarding test (FT) decides through which link(s) an incoming probe is forwarded, after which resources are reserved on-the-go. This fully distributed approach eliminates the need to maintain a global view of the network and the available resources. When a probe reaches the destination, a path from the source to this destination has been fully established, since resources are already reserved. Note that exploration-based protocols are only concerned with installing paths for specific flows. Therefore, the exploration mechanisms are employed on top of a best-effort routing protocol like OSPF or EIGRP. The proposed exploration-based protocols exploit the fact that this best-effort routing protocol provides valuable information, such as the cost (hop count) to the destination. This information can be used to guide and/or optimize the exploration process. Besides probes, two other types of packets are used for signaling purposes, both of which can be seen as a response to a probe. On the one hand, the prune packets signal that a path could not be found (due to, e.g., insufficient resources). Whenever a router receives a prune packet for every probe it has sent out, it knows that no path could be found through any of its links, so it returns a prune as well. On the other hand, the confirmation packets signal that a path was found. Once a probe reaches the destination, a confirmation packet is sent back. Upon reception of a confirmation, the router frees unnecessarily reserved resources, while also sending a confirmation back to the source. The following sections provide more information about the FT and different exploration flavors.

1) Forwarding Test (FT): In this case, the FT is specialized for guaranteeing queueing and end-to-end delay bounds, and contains some additional measures to try and keep the overhead low. Besides the flow parameters, a probe also carries additional statistics, as listed in Tab. II. The local information that router *i* disposes of is listed in Tab. III.

The queue-level topology link e^q_{ij} will now pass the FT if

1) $d_{acc} + d_{e^q_{ij}} \leq d_f$ (delay guarantee), and
2) The flow can be admitted to e^q_{ij} (cf. Section III-D), and
3) No prune was received earlier from e_{ji}, and
4) The probe did not come from e_{ji}, and
5) The flow is not already assigned to a queue-level topology link that also belongs to e_{ij}, and
6) $h + h_{e^q_{ij}} \leq H$ (path pruning), and
7) The protocol allows it.

Rule 5 specifies that only one queue-level topology link can be used per probe per physical link, which can be chosen heuristically. Here, the available queue with the lowest delay budget is chosen. Rule 6 tries to keep the overhead low by not exploring links that are too far away. Given the info of the best-effort routing protocol, this rule can cut off certain paths earlier, compared to the basic rule $h + 1 \leq H$. Since the worst-case burst increases at every hop (Eq. (5)), traversing long paths wastes lots of resources, so cutting these out is a good idea. Rule 7 provides flexibility to implement different exploration flavors. When a probe is to be forwarded through a link, its statistics are updated accordingly: h is increased by 1, d_{acc} is increased by $d_{e^q_{ij}}$, and $\alpha_f(t)$ is updated according to Eq. (5).

2) Depth-First (DF) Exploration: The idea of depth-first (DF) exploration is to only explore a single path at a time. Therefore, the FT needs to decide which link to explore first. In this case, the FT always prefers the link with the lowest cost (hop count) to the destination, which is provided by the best-effort routing protocol. If two links are tied for the lowest hop count, a decision is made randomly. Given the toy network of Fig. 2 and flow f_1 (cf. Tab. I), a simple example of DF exploration is depicted in Fig. 4. Note that only one link passes the FT, and that there is a tie for lowest hop count between the links to n_1 and n_2. In the example, the source router *S* makes a bad decision by forwarding the probe ('*probe_msg*') to intermediate node n_2 as n_2 cannot forward the probe without violating the maximum end-to-end delay. Therefore, n_2 sends a prune ('*prune_msg*') back to *S*. Since *S* receives a prune back, it frees up the previously reserved resources, runs the FT again, and forwards the probe now to n_1. After n_1 forwards the probe to the destination, a confirmation ('*confirm_msg*') is signaled back to the source.

TABLE II
EXPLORATION DATA: PROBE DATA

Name	Meaning
d_{acc}	Accumulated worst-case delay
H	Max. hop count
h	Accumulated hop count
src_f	Flow source
$dest_f$	Flow destination
d_f	Flow end-to-end delay requirement
$\alpha_f(t)$	Flow arrival curve

TABLE III
EXPLORATION DATA: ROUTER-LOCAL DATA

Name	Meaning
$d_{e^q_{ij}}$	Worst-case link delay of e^q_{ij}
$h_{e^q_{ij}}$	Min. hops to the flow destination through e^q_{ij}
$A_{e^q_{ij}}$	Aggregated arrival curve at e^q_{ij}
$B_{e^q_{ij}}$	Service curve at e^q_{ij}

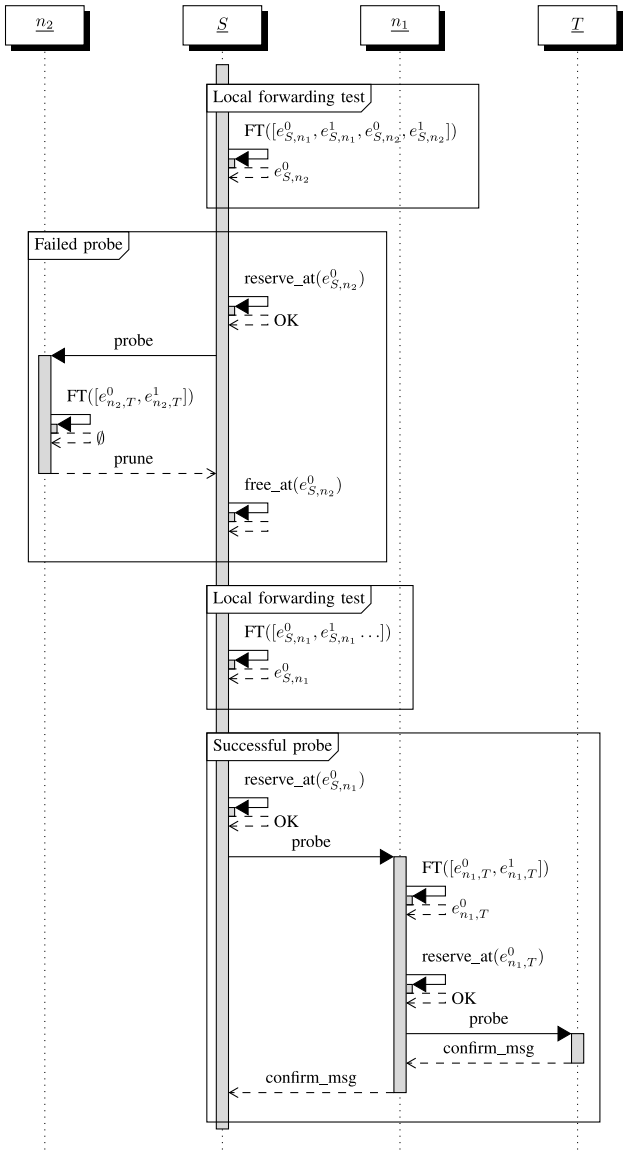


Fig. 4. Exploration-based depth-first routing process and signaling upon a flow request for f_1 (cf. Tab. I), applied to the toy network in Fig. 2.

3) Breadth-First (BF) Exploration: When using a breadth-first (BF) exploration scheme, a probe will be forwarded through all links that pass the FT in parallel. This means that lots of paths can be explored simultaneously, at the risk of over-reservation. A fast-prune mechanism, similar to [29], tries to counter this. It works as follows:

- If a probe arrives at a node while this node already received a probe for this flow, then a prune is sent back immediately.
- When a confirmation packet is received at a node, a prune-forward packet is sent through every link where a response was not yet received. Upon reception of a prune-forward packet, all resources are cleared and it is forwarded again through every link where a response was not yet received.

Given the example flow f_1 (cf. Tab I) in the toy network from Fig. 2, Fig. 5 shows a simple example of BF exploration. Here, multiple links can pass the FT and the source router S

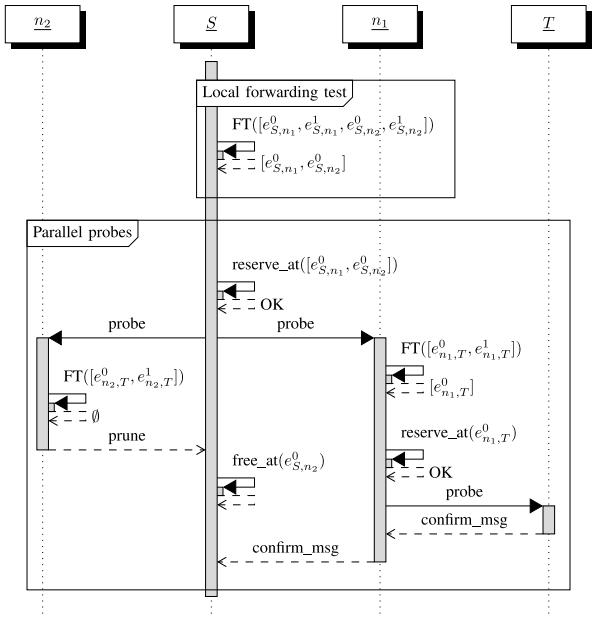


Fig. 5. Exploration-based breadth-first routing process and signaling upon a flow request for f_1 (cf. Tab. I), applied to the toy network in Fig. 2.

sends out two probes ('*probe_msg*') in parallel to intermediate nodes n_1 and n_2. Since n_2 responds again with a prune ('*prune_msg*'), S frees the previously reserved resources on the link to n_2. At the same time, n_1 explores further until the destination is reached. Finally, a confirmation ('*confirm_msg*') is sent back via n_1.

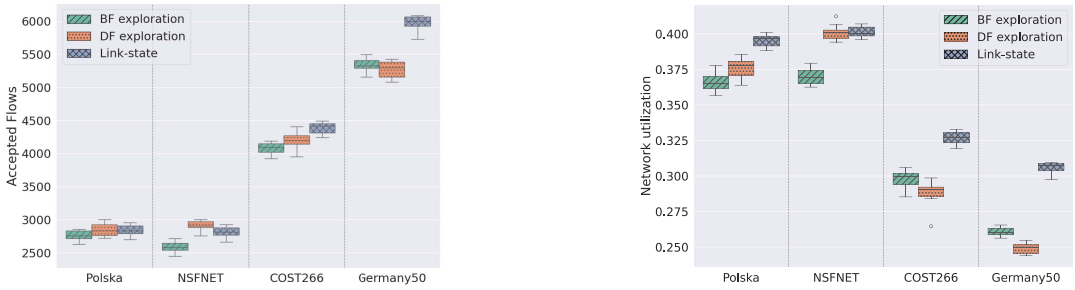
V. Evaluation

A. Experimental Setup

The routing protocols of this paper were evaluated in the OMNeT++ simulation environment [32], extended with the INET framework [33]. The basic experiments are performed on 4 networks: Polska, NSFNET, COST266, and Germany50, provided by SNDLib [34]. Their characteristics can be seen in Tab. IV. For the experiments, all of the links are bidirectional and have a bandwidth of 1 Gbps, and the propagation speed is set to 200,000 km/s. The routers are equipped with 8 priority queues for real-time traffic, and the delay budgets are set according to the 'Base' configuration, as listed in Tab. VI. Every priority queue has a fixed buffer size of 97kB. The experiments were conducted on a laptop using Ubuntu 20.04 with an Intel COREvPro i7 CPU. The C++ implementation of the protocols is available as open source [35]. The traffic characteristics can be seen in Tab. V. New flows are requested at a rate of 10 flows per 10ms during a period of 10s, resulting in a total of 10,000 requested flows. The source and destination of a flow are always chosen randomly out of the possible source-destination pairs in the network. The traffic characteristics are also uniformly distributed based on Tab. V. Unless stated otherwise, the event-based update threshold of the LS protocol is 10% of the link bandwidth.

B. Routing Performance

The routing performance of the different protocols is evaluated in terms of accepted flows and network utilization. The



(a) (b)

Fig. 6. Routing performance results. (a) In terms of accepted flows. (b) In terms of network utilization.

TABLE IV
BASELINE NETWORKS

Network name	Nodes	Links	Density
Polska	12	18	0.27
NSFNET	14	21	0.23
COST266	37	57	0.08
Germany50	50	88	0.07

TABLE V
TRAFFIC CHARACTERIZATION, BASED ON [28]

Application	Deadline	Burst	Rate
Database	[80-120] ms	[100-400] B	[300-550] Kbps
SCADA	[150-200] ms	[100-400] B	[150-550] Kbps
Production	[10-20] ms	[100-400] B	[100-500] Kbps
Control	[10-20] ms	[80-120] B	[1-100] Kbps
Video	[50-200] ms	[140-1800] B	[4-16] Mbps

TABLE VI
DELAY BUDGET CONFIGURATIONS (IN MS)

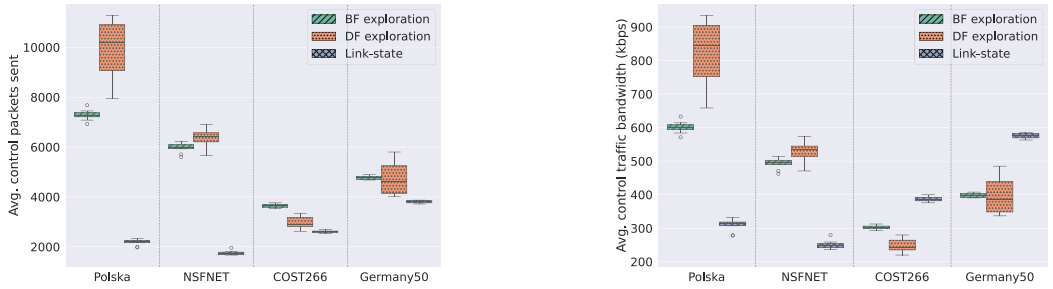
Name	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
Base	0.1	0.5	1	2	4	8	16	32
Chameleon	0.1	0.5	1	1.5	3	6	12	24
Const	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Exp	0.1	0.2	0.4	0.8	1.6	3.2	6.4	12.8
Gap	0.5	1.0	1.5	2	8	8.5	9	10
Increasing1	0.1	0.3	0.6	1	1.5	2.1	2.8	3.6
Increasing2	0.1	0.5	1.1	1.9	2.9	4.1	5.5	7.1
Simple1	0.5	1	1.5	2	2.5	3	3.5	4
Simple2	1	2	3	4	5	6	7	8
Simple3	0.5	2	3.5	5	6.5	8	9.5	11

network utilization is here defined as the amount of bandwidth that has been reserved for flows, divided by the total available bandwidth. While the number of accepted flows is the simplest metric for performance, network utilization is also interesting, as the admission control component makes sure that flows are admitted in a queue only if there are enough resources. Thus, the more bandwidth is reserved, the more efficiently queue resources are used, resulting in more critical traffic that can flow through the network. The results can be seen in Fig. 6. For both the number of accepted flows and network utilization, the LS protocol performs better than the exploration-based protocols in the bigger and less dense networks (COST266 and Germany50). This can be explained from the fact that the LS protocol is globally aware of the least-filled queues, given a reasonable accuracy. For the other networks, the number of accepted flows are somewhat similar, while the LS protocol typically has an edge regarding the network utilization. The smaller networks highlight the biggest problem of the BF exploration protocol: over-reservation. Since it explores (and reserves resources) in parallel, over-reservation is more likely to happen in a small, dense network with a limited amount of links.

Therefore, DF exploration is a more sensible choice in such networks. However, in the other networks, BF and DF exploration have similar results. Exploration-based protocols are also a bit more variable, while the performance of the LS protocol is more consistent. The reason for this is that the exploration-based protocols are more susceptible to the network conditions (background traffic, other requested flows, queueing delays, etc.) when a new flow is requested, since they require a lot of packet exchanges. However, since the control traffic is allocated to a high priority queue, as explained in Section III-E, the effects of background traffic are largely mitigated.

C. Control Overhead

Since both control architectures are fully distributed, every protocol sends control packets to operate properly. This section evaluates this control overhead in terms of packets sent, and in terms of bandwidth. The results regarding the number of control packets can be found in Fig. 7(a). Exploration-based protocols send significantly more packets into the network than the LS protocol. This could be expected, as lots of packet exchanges are inherent to their design. Since the LS protocol relies on flooding to distribute information, it also sends a fair amount of packets, although not as much as exploration-based protocols. Fig. 7(b) shows that the exploration-based protocols are actually more efficient in terms of bandwidth when looking at the bigger networks. While the exploration-based protocols send more packets, the data in those packets is relatively small compared to the packet data of QLSAs that include detailed link information, such as the available resources. The reservation messages of the link-state protocol also carry the full path of the flow in order to reserve resources along the path to the destination. Exploration packets only include the flow identification and some light-weight statistics, which really makes the difference here. Finally, note that exploration-based protocols pose a higher burden on links in smaller, denser networks, since links are more likely to be used for multiple exploration processes.



(a) (b)

Fig. 7. Control traffic overhead results per link. (a) In terms of average packets sent. (b) In terms of average bandwidth.

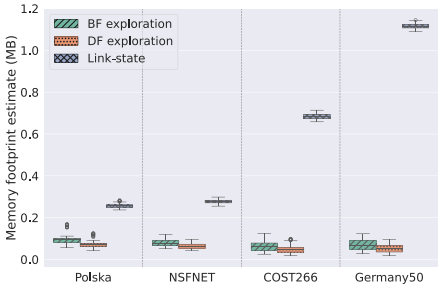


Fig. 8. Memory footprints estimations

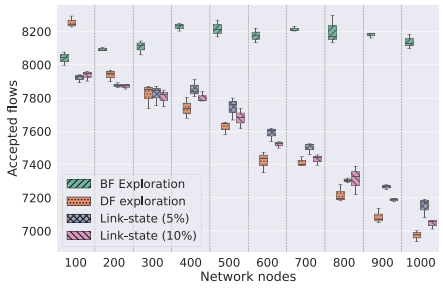


Fig. 10. Accepted flows for varying network sizes.

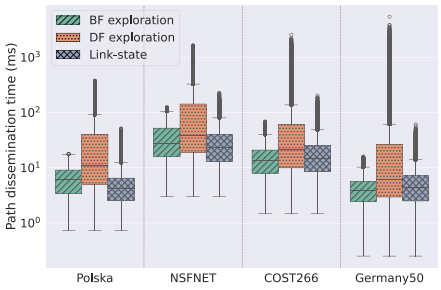


Fig. 9. Path dissemination times.

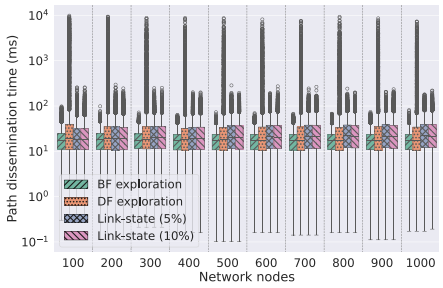


Fig. 11. Path dissemination times for varying network sizes.

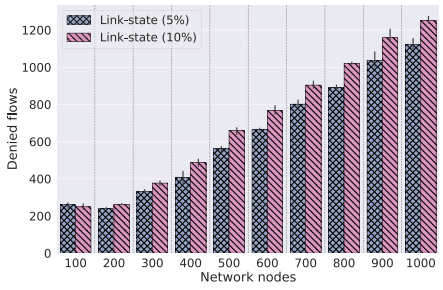


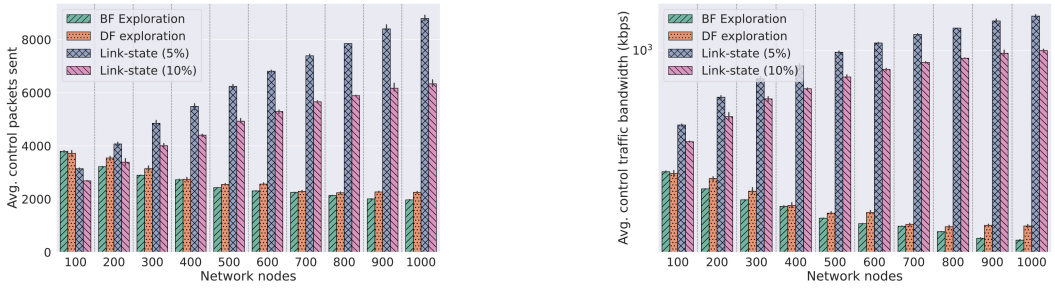
Fig. 12. Denied flows due to inaccuracy failures in the LS protocol for varying network sizes.

D. Memory Footprint

Memory footprint estimates are computed at runtime by traversing a set of fundamental data structures of the protocols. This set includes the state that is necessary to route flows. For the LS protocol, the queue-level topology view and QLSA database are counted. Exploration-based protocols do not maintain a queue-level topology, but require some state while exchanging exploration packets with neighbors and waiting for responses. This state overhead is also added to their memory footprint estimates. All of the protocols are built on top of a best-effort routing protocol, but the state overhead of this protocol is not counted. Fig. 8 shows the results of this experiment. For all networks, the LS protocol requires significantly more memory than the exploration-based protocols. This is of course due to the fact that the LS protocol maintains a global (big) view, while exploration-based protocols require only local state. The exploration-based protocols even manage to keep their memory overhead almost constant, independent of the network.

E. Path Dissemination Time

The path dissemination time is defined as the time between an application sending out of the flow request into the network and the response thereof, i.e., after the network has installed a full path or concluded that this was not possible. Fig. 9 shows



(a) (b)

Fig. 13. Control overhead per link for varying network sizes (a) In terms of average packets (b) In terms of average bandwidth.

similar performance for the BF exploration and LS protocols, but the DF exploration protocol performs not as well and has many more outliers. Since DF exploration only explores one path at a time, it can take a very long time to revisit a bad decision, which results in big outliers. The outliers of the LS protocol are a consequence of the retry mechanism to counter inaccuracies. In the worst-case, if there is a spike of inaccuracies, multiple retries can be triggered, contributing significantly to the path dissemination times. Finally, it can be observed that, although COST266 and NSFNET have fewer nodes than Germany50, the path dissemination times are higher in these networks. This is because they cover a larger geographical area, resulting in higher path dissemination times due to higher link delays.

F. Scalability

To evaluate the scalability of the protocols, several large-scale networks were generated using the Igen topology generator [36]. These networks range from 100 to 1000 nodes, all covering the geographical area of Australia. Every experiment is repeated 3 times. For the LS protocol, two event-based thresholds are compared: 5% and 10% of the link bandwidth. Fig. 10 depicts the number of accepted flows for varying network sizes. The main observation here is that the performance of the LS and DF exploration protocol is declining linearly. For the LS protocol, this is due to the inaccuracies of its global topology map. In a bigger network, it is harder to keep this up-to-date, resulting in more bad routing decisions. Although increasing the update frequency helps the routing performance, the number of denied flows due to inaccuracies still rises, which can be seen in Fig. 12. The reason is that the physical link propagation delays still incur significant delays to every update, resulting in inevitable inaccuracies. As depicted in Fig. 11, the DF exploration-based protocol suffers from high path dissemination times, which significantly impacts its performance. Again, since the DF exploration protocol only explores one path at a time, the more choices it can make in a bigger network, and if it makes a bad decision early on, then it can take a very long time to revisit this. Ultimately, this hurts its performance. This is in contrast with the BF protocol that seems to find a good balance between exploring multiple paths and not exploring too much. Fig. 11 depicts all path dissemination times for increasing network sizes. In general, the path dissemination

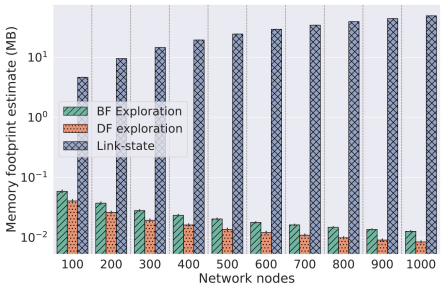


Fig. 14. Memory footprint estimates for varying network sizes.

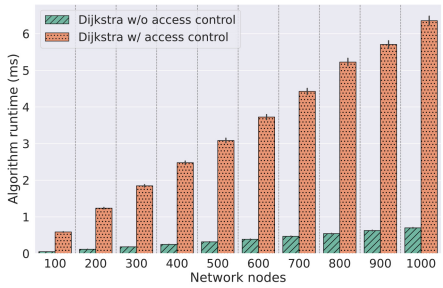
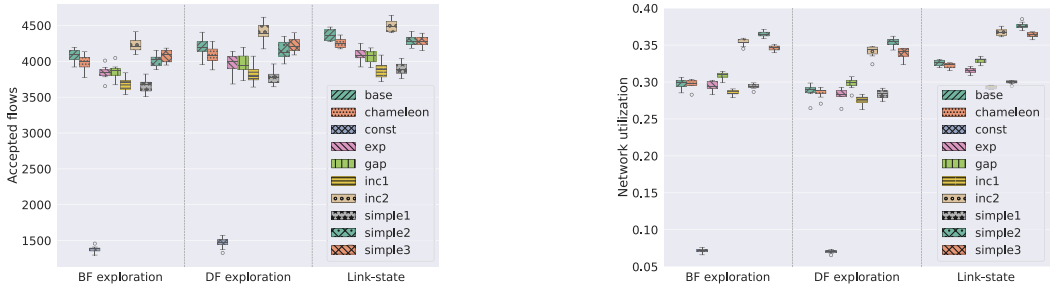


Fig. 15. Link-state routing algorithm runtimes for varying network sizes.

times don't go up for increasing network size, but heavy outliers can be expected, especially for the DF exploration-based protocol. The BF exploration variant is faster than the DF variant, which can be expected given its parallel exploration process. The outliers of the LS protocol are a direct cause of the inaccuracy mechanism that will retry to find a route when an inaccuracy failure happened. Fig. 13 depicts the control overhead both in packets and data sent into the network, averaged per link. It can be observed that the LS protocol sends significantly more packets for bigger topologies. This can be expected, since it relies on flooding to distribute queue state information. The same trend can also be seen by looking at the overhead in terms of bandwidth, as the LS protocol distributes notably more data across the network. This is again because QLSAs contain a lot more data than simple exploration packets. Moreover, increasing the number of event-based updates has the logical effect of introducing additional overhead. Interestingly, the control overhead of the DF exploration protocol is higher than the overhead of the BF variant, both in terms of packets and bandwidth. This means that the added efficiency



(a) (b)

Fig. 16. Delay budget configuration influence on routing performance for the COST266 network. (a) In terms of accepted flows. (b) In terms of network utilization.

measures (fast-prune and path prune) of the BF exploration-based protocol work well for countering overhead due to parallel exploration. Memory footprint estimates for increasing network sizes are shown in Fig. 14, which are derived in the same way as in Section V-D. Because the queue-level topology grows as the network topology becomes bigger, the LS protocol (regardless of its update frequency) requires significantly more and more memory as the scale of the network increases. For the exploration-based protocols, the memory footprints are even decreasing for bigger networks. This is due to the fact that 10,000 flows are requested in every network. A big contributor to the memory footprint of exploration-based protocols is the state that they need to keep between exploration packet exchanges (e.g., probes sent out, links not tried, flow parameters, etc.). For a bigger network and the same number of flows, it is less likely that a node needs to handle multiple flows at the same time, therefore the memory footprint decreases. Finally, Fig. 15 depicts the runtime performance of the LS routing algorithm, compared to the standard shortest-path algorithm (Dijkstra). As explained in Section IV-A2, the LS routing algorithm is a modified version of the standard algorithm that incorporates an admission control mechanism. It can be observed that this modification adds a lot of extra runtime. However, it does follow the trend of the standard algorithm. Since the admission control system needs to check every queue for violated bounds when expanding a node in the algorithm, a constant factor is expected to be added, since the number of queues is a small constant (e.g., 8). For bigger networks, the total runtime overhead reaches several milliseconds, which also contributes significantly to the path dissemination times of the LS protocol.

G. Delay Budget Configurations

The delay budgets are an integral factor in guaranteeing bounded end-to-end delay, which can be configured in different ways. Consider the delay budget of Q0. If set to $100\mu s$, very tight queueing delay bounds can be accommodated, which might be necessary for some flows. If this delay budget would be $500\mu s$, the worst-case queueing delay bound is higher, which could negatively impact certain flows. However, more resources are available in this queue for flows to reserve as per NC. Therefore, different configurations might influence overall routing performance. To this end, 10 different queueing delay budget configurations for 8 priority queues were evaluated regarding routing performance. The specific configurations are listed Tab. VI, and the results are depicted in Fig. 16. From this figure, significant differences between certain configurations are observed, as some configurations are clearly better than others. It is also noticeable that the same performance patterns arise per protocol, which could be a sign that there exist an 'optimal' configuration of the delay budgets for a given network and traffic matrix. Finding such an optimal configuration lies outside the scope of this work.

VI. Conclusion

This paper presented three different distributed control architectures exploiting queue state information and NC for deterministic networking in L3 networks. These architectures do not rely on time synchronization among involved nodes, which makes deployment in large-scale L3 networks more feasible compared to TSN mechanisms. The use of queueing delay budgets in this paper enables a flexible way to accommodate dynamic flows with diverse latency requirements in the network. Multiple architectures were proposed and evaluated: an LS protocol extended with queue-level information, as well as two exploration-based protocols. The LS protocol requires adequate and timely dissemination of network topology and queue-level information in order to calculate fulfilling delay-constrained paths. On top of that, a signaling mechanism is required to effectively reserve the resources in the network. This combination proved to provide adequate results w.r.t. acceptance rate and network utilization in smaller-scale networks. However, in larger-scale networks, it suffers from incoherent network views in nodes resulting from the QLSA flooding and associated lag. As a result, the acceptance rate decreases for larger-scale networks, while the cost in terms of control overhead and memory footprint (at least) linearly increases at steep rates. Exploration-based protocols produce different trade-offs. Their topology view is limited, resulting in a DF or BF exploration strategy from the initiating (source) node. In smaller-scale networks this comes at the cost of potentially selecting non-optimal paths satisfying the required delay constraints, resulting in sub-optimal acceptance rates and

network utilization, but with the benefit of very limited control and memory overhead. The true value of these protocols is that the overhead in terms control messaging and memory continues to scale very well (i.e., linear increase with minimal slope), while still providing adequate – near stable – performance in terms of acceptance rate, network utilization and path dissemination times in larger-scale networks. This makes them very well-suited for routing in larger L3-based deterministic networks.

REFERENCES

[1] A. Nasrallah et al., "Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 88–145, 1st Quart., 2019.

[2] E. Grossman, "Deterministic networking use cases," RFC 8578, IETF, May 2019. [Online]. Available: https://www.rfc-editor.org/info/rfc8578

[3] J. Miserez, G. P. Sharma, and W. Tavernier, "Routing protocols exploiting queue information for deterministic networks," in *Proc. 19th Int. Conf. Design Rel. Commun. Netw. (DRCN)*, 2023, pp. 1–8.

[4] *IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications*, IEEE Standard 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011), 2020.

[5] T. Stüber, L. Osswald, S. Lindner, and M. Menth, "A survey of scheduling in time-sensitive networking (TSN)," 2022, *arXiv:2211.10954*.

[6] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," *Comput. Netw.*, vol. 112, pp. 279–293, Jan. 2017.

[7] *IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks*, IEEE Standard 802.1Q-2022 (Revision of IEEE Std 802.1Q-2018), 2022.

[8] "IEEE P802.1Qdd—Resource allocation protocol," 2023. [Online]. Available: https://1.ieee802.org/TSN/802-1qdd/

[9] N. Finn, P. Thubert, B. Varga, and J. Farkas, "Deterministic networking architecture," RFC 8655, IETF, Oct. 2019. [Online]. Available: https://www.rfc-editor.org/info/rfc8655

[10] C. Srinivasan, T. Nadeau, and A. Viswanathan, "Multiprotocol label switching (MPLS) traffic engineering (TE) management information base (MIB)," RFC 3812, IETF, Jun. 2004. [Online]. Available: https://www.rfc-editor.org/info/rfc3812

[11] B. Varga, J. Farkas, L. Berger, A. G. Malis, S. Bryant, and J. Korhonen, "Deterministic networking (DetNet) data plane: MPLS," RFC 8964, IETF, Jan. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8964

[12] A. Viswanathan, E. C. Rosen, and R. Callon, "Multiprotocol label switching architecture," RFC 3031, IETF, Jan. 2001. [Online]. Available: https://www.rfc-editor.org/info/rfc3031

[13] B. Varga, J. Farkas, L. Berger, D. Fedyk, and S. Bryant, "Deterministic networking (DetNet) data plane: IP," RFC 8939, IETF, Nov. 2020. [Online]. Available: https://www.rfc-editor.org/info/rfc8939

[14] X. Wang, J. Dai, J. Liu, and F. Zhang, "DetNet data plane: IEEE 802.1 time sensitive networking over SRv6," Internet Eng. Task Force, Internet-Draft draft-wang-detnet-tsn-over-srv6-07, Jun. 2023, Work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-wang-detnet-tsn-over-srv6/07/

[15] D. O. Awduche, L. Berger, D.-H. Gan, T. Li, D. V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP tunnels," RFC 3209, IETF, Dec. 2001. [Online]. Available: https://www.rfc-editor.org/info/rfc3209

[16] A. G. Malis, X. Geng, M. Chen, F. Qin, B. Varga, and C. J. Bernardos, "Deterministic networking (DetNet) controller plane framework," Internet Eng. Task Force, Internet-Draft draft-ietf-detnet-controller-plane-framework-05, Sep. 2023, Work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-detnet-controller-plane-framework/05/

[17] D. M. Yeung, D. Katz, and K. Kompella, "Traffic engineering (TE) extensions to OSPF version 2," RFC 3630, IETF, Oct. 2003. [Online]. Available: https://www.rfc-editor.org/info/rfc3630

[18] P. Liu et al., "Requirements for scaling deterministic networks," Internet Eng. Task Force, Internet-Draft draft-ietf-detnet-scaling-requirements-04, Oct. 2023, Work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-detnet-scaling-requirements/04/

[19] T. Eckert, S. Bryant, A. G. Malis, and G. Li, "Deterministic networking (DetNet) data plane—Tagged cyclic queuing and forwarding (TCQF) for bounded latency with low jitter in large scale DetNets," Internet Eng. Task Force, Internet-Draft draft-eckert-detnet-tcqf-04, Jul. 2023, Work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-eckert-detnet-tcqf/04/

[20] J. Joung, J. Kwon, J.-D. Ryoo, and T. Cheung, "Scalable flow isolation with work conserving stateless core fair queuing for deterministic networking," *IEEE Access*, vol. 11, pp. 105225–105247, 2023.

[21] N. Finn, J.-Y. L. Boudec, E. Mohammadpour, J. Zhang, and B. Varga, "Deterministic networking (DetNet) bounded latency," RFC 9320, IETF, Nov. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9320

[22] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Berlin, Germany: Springer, 2001.

[23] L. Zhao, P. Pop, Z. Zheng, H. Daigmorte, and M. Boyer, "Latency analysis of multiple classes of AVB traffic in TSN with standard credit behavior using network calculus," *IEEE Trans. Ind. Electron.*, vol. 68, no. 10, pp. 10291–10302, Oct. 2021.

[24] J. Zhang, L. Chen, T. Wang, and X. Wang, "Analysis of TSN for industrial automation based on network calculus," in *Proc. 24th IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, 2019, pp. 240–247.

[25] A. Berisa et al., "AVB-aware routing and scheduling for critical traffic in time-sensitive networks with preemption," in *Proc. 30th Int. Conf. Real-Time Netw. Syst.*, 2022, pp. 207–218.

[26] L. Zhao, Y. Yan, and X. Zhou, "Minimum bandwidth reservation for CBS in TSN with real-time QoS guarantees," *IEEE Trans. Ind. Informat.*, vol. 20, no. 4, pp. 6187–6198, Apr. 2024.

[27] K. Jang, J. Sherry, H. Ballani, and T. Moncaster, "Silo: Predictable message latency in the cloud," in *Proc. ACM Conf. Special Interest Group Data Commun.*, 2015, pp. 435–448.

[28] A. Van Bemten et al., "Chameleon: Predictable latency and high utilization with queue-aware and adaptive source routing," in *Proc. 16th Int. Conf. Emerg. Netw. Exp. Technol.*, 2020, pp. 451–465.

[29] J. Song, H. K. Pung, and L. Jacob, "A multi-constrained distributed QoS routing algorithm," in *Proc. IEEE Int. Conf. Netw. Netw. Trends Challenges New Millennium*, 2000, pp. 165–171.

[30] A. Van Bemten and W. Kellerer, "Network calculus: A comprehensive guide," Lehrstuhl fur Kommunikationsnetze, Technische Universitat München, München, Germany, Rep. 201603, 2016.

[31] R. A. Guerin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions," in *Proc. IEEE Global Telecommun. Conf. Conf. Rec.*, vol. 3, 1997, pp. 1903–1908.

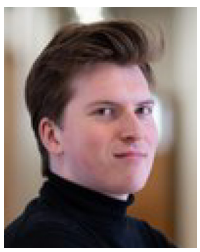
[32] A. Varga, "OMNeT++," in *Modeling and Tools for Network Simulation*. Berlin, Germany: Springer, 2010, pp. 35–59.

[33] "INET framework: An open-source OMNeT++ model suite for wired, wireless and mobile networks," Accessed: Nov. 1, 2023. [Online]. Available: https://inet.omnetpp.org

[34] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0—Survivable network design library," in *Proc. 3rd Int. Netw. Optim. Conf. (INOC)*, Apr. 2007, pp. 276–286. [Online]. Available: http://www.zib.de/orlowski/Paper/OrlowskiPioroTomaszewskiWessaely2007-SNDlib-INOC.pdf.gz

[35] J. Miserez, "Routing protocols exploiting queue information for deterministic networks—Implementation," [Online]. Available: https://github.com/jakobmiserez/thesis

[36] B. Quoitin, V. Van den Schrieck, P. François, and O. Bonaventure, "IGen: Generation of router-level Internet topologies through network design heuristics," in *Proc. 21st Int. Teletraffic Congr.*, 2009, pp. 1–8.



Jakob Miserez received the M.Sc. degree in computer science from Ghent University, Ghent, Belgium, in 2022.

He is currently pursuing the Ph.D. degree in computer science engineering with the IDLab, Ghent University—imec. His research interests include network calculus, algorithms and (routing) protocols for time-sensitive, and deterministic networking.

Didier Colle (Member, IEEE) received the M.Sc. and Ph.D. degrees in electrotechnical engineering from Ghent University, Belgium, in 1997 and 2002, respectively.

He has been a Senior Full Professor with Ghent University since 2022. He was an Associate Professor with Ghent University in 2011, where he has been a Full Professor since 2014. He is co-responsible for the research cluster on network modeling, design, and evaluation (NetMoDeL) inside the imec IDLab Research Group. His research is mainly conducted inside international (mainly European), national, and bilateral research projects together with the industry. This research has been published in more than 500 international journal and conference articles, and has resulted in more than 20 Ph.D. degrees. This research cluster deals with fixed internet architectures and optical networks, green ICT, the design of network algorithms, and techno-economic studies.

Mario Pickavet (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in electrical engineering from Ghent University, Ghent, Belgium, in 1996 and 1999, respectively.

Since 2000, he has been a Professor with Ghent University, where he is teaching courses on discrete mathematics and network modeling. He is coleading the research cluster on network modeling, design, and evaluation (NetMoDeL). In this context, he is currently involved in several European and national projects. He has authored or co-authored about 500 international publications, both in journals and proceedings of conferences. He is the co-author of the book *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. His main research interests are fixed Internet architectures and optical networks, green ICT, and the design of network algorithms. He is a holder of a Bronze Medal at the International Mathematical Olympiad, Sweden, in 1991.

Wouter Tavernier received the B.S. and M.S. degrees in computer science and the Ph.D. degree in reliable routing and switching from Ghent University, Belgium, in 2002 and 2012, respectively.

In 2006, he joined the Internet-Based Communications Networks Group (which became part of IDLab in October 2016) of Ghent University as a Researcher of Carrier Ethernet. He is currently employed as a Professor with Ghent University, where he teaches courses on computer networks. This work is performed in the context of European projects, such as H2020 5G-CHAMPION, NGPAAS, SONATANFV, 5G TANGO, and HEXA-X-II. This research has been published in more than 100 scientific publications. His current research interests focus on performance and resource optimization aspects of network function virtualization, and deterministic networking.