Routing protocols exploiting queue information for deterministic networks

Jakob Miserez*, Gourav Prateek Sharma†, Wouter Tavernier*
*IDLab, Department of Information Technology, UGent - IMEC
{jakob.miserez,wouter.tavernier}@UGent.be
†EECS, KTH Royal Institute of Technology
{gpsharma@kth.se}

***Abstract*—Time-Sensitive Networking (TSN) provides mechanisms for strictly controlling network latency, jitter, and packet loss in LAN environments. These technologies enable various applications, including industrial automation and professional audio/video bridging. Current TSN network architectures heavily rely on a centralized control model inspired by Software-Defined Networking (SDN). While this approach works well for small-scale L2 TSN networks, it is not feasible for larger-scale L3 deterministic networks involving multiple network segments and routers. This paper proposes and evaluates distributed mechanisms for Deterministic Networking (DetNet) that enable: i) the discovery of priority queue information using a link-state protocol or an exploration-based approach, ii) the planning of network paths using network calculus based on available queue information, and iii) a distributed signaling mechanism for configuring the data plane of the deterministic network. Simulations demonstrate the potential of these protocols in large-scale networks and validate their bounded delay and packet loss.**

***Index Terms*—Deterministic networking, routing protocols, Quality-of-Service, QoS, real-time applications**

I. Introduction

Traditional IP-based networks provide best-effort end-to-end connectivity to their users. In practice, this leads to end-to-end latencies which are at best in the order of tens of milliseconds [1]. In addition, no guarantees are made regarding packet loss, jitter, or throughput. While this is acceptable for traditional Internet applications, critical applications that demand such guarantees cannot be trusted to those networks. Examples of such critical applications include industrial applications, the tactile Internet, and the one-way fronthaul in wireless cellular networks, which require worst-case latencies of respectively a few milliseconds, 1 ms, and in the order of 100 μs [1]. Time-Sensitive Networking (TSN) technologies and mechanisms have been proposed in L2 networks to accommodate such tight latency requirements. As documented in section II, existing TSN work focuses on LAN environments involving different data plane shapers, flow control mechanisms, and scheduling in the context of a control architecture that relies on a Centralized Network Controller (CNC), which is responsible for discovering each node's queue status, scheduling TSN paths and instructing the data plane nodes. Although a centralized control architecture might be suitable for L2 TSN networks with relatively small scale, such an approach is less feasible over multiple L2 bridged and L3 routed segments. A scalable control architecture suggests a more distributed approach where all nodes are jointly responsible for: i) setting up the control network, ii) disseminating node-level routing/queue information, and iii) participating in the calculation, scheduling and signaling of time-sensitive paths satisfying latency, bandwidth, and packet loss requirements. Such an architecture aims to reduce signaling overhead and the dependency on a single CNC, which has the possibility to become either a performance bottleneck or the target of a hardware failure or attack. This paper aims to investigate an alternative distributed routing approach enabling deterministic networking by exploiting fine-grained queue information, proposing three novel distributed routing schemes that build on top of related work in link-state routing, network calculus, and exploration-based routing protocols.

The remainder of this paper is organized as follows. Section II provides an overview of related work that inspired the contributions of this paper. Section III describes the basic building blocks of the protocols, after which the proposed protocols are explained in sections IV and V. The simulation setup is presented in section VI followed by the evaluation of the protocols. Finally, this paper is concluded in section VII, where a few pointers for future work are presented.

II. Related work

Van Bemten et al. [2] proposed Chameleon, a demand-aware cloud network that provides predictable latency and high utilization in multi-tenant datacenters. Chameleon builds on network calculus [3] to compute worst-case delay and memory bounds, and the queue-level topology, which is a network abstraction that explicitly accounts for priority queueing in an output port. Building on the observation that packets going through queues of different priority experience vastly different delays, a physical link is now abstracted to several virtual links, one for each queue. Every virtual link is assigned a worst-case queueing delay, i.e., a budget, with higher priority queues having lower budgets. Chameleon is implemented using a logically centralized controller. Whenever a new flow is requested, a feasible path is computed using the LARAC algorithm [4]. An admission control component based on

978-1-6654-7598-3/23/$31.00 ©2023 IEEE

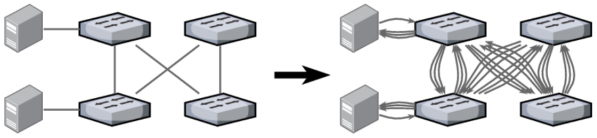


Fig. 1: An illustration showing a network and its queue-level representation [2].

network calculus within the path computation makes sure that a flow is never embedded in a queue such that it would violate the queue's delay budget or the budgets of lower priority queues. The buffer requirements are also taken into account. Chameleon comes with a reconfiguration system, which can re-embed flows to make place for other flows if necessary. The queue-level topology allows Chameleon to perform fine-grained routing while guaranteeing bounded delay, bandwidth, and packet delivery to applications.

Time-Sensitive Networking (TSN) is a set of networking technologies and standards that are designed to ensure that time-critical data is transmitted reliably and with predictable latency over switched Ethernet networks [1]. TSN is standardized in the IEEE 802.1 TSN task group and includes features such as time synchronization, traffic shaping, filtering and policing mechanisms, and path selection and network traffic scheduling algorithms, which all help to ensure that time-critical data is transmitted reliably and efficiently over the network. For example, the Time-Aware Shaper (TAS), as standardized in IEEE 802.1Qbv, relies on controlled gates to explicitly open and block queues for transmission on periodic timescales of the order of tens to hundreds of nanoseconds. The TAS is one of the primary vehicles to protect scheduled traffic from other traffic in given nodes to bound its queueing delay. Time-triggered or scheduled traffic refers to periodic data streams with real-time requirements that are known in advance. Synchronization between nodes is provided through the Precision Time Protocol (PTP) protocol. Based on these requirements and the network topology, the CNC generates the periodic schedule for all switches. Once the schedule is generated, the CNC provides the individual switches with their shaping configuration. An extensive survey of the current state-of-the-art of TSN focusing on URLLC applications is given in [1]. A significant amount of TSN research has focused on the design of efficient scheduling mechanisms [5].

Whereas TSN technologies and standards focus on providing network guarantees in L2 LAN environments, the need for more deterministic networking in larger-scale routed L3 networks is acknowledged by the IETF deterministic networking (DetNet) working group. There is a wide range of applications ranging from Industrial Automation and Control Systems (IACSs) to electrical utilities [6] with strong and relatively similar needs for deterministic network services in larger-scale L3 routed networks. Such environments pose additional challenges beyond those targeted in traditional TSN standards, in particular regarding scalability, routing, path distribution, and signaling [6]. An IETF DetNet draft [7] is currently being developed to outline a more scalable control framework for larger-scale L3 networks. Initial assessments acknowledge scalability issues with SDN-focused architectures and see potential in the reuse of existing L3 routing protocols, such as OSPF-TE, and signaling protocols such as RSVP-TE. However, none of these protocols are currently aware of the fine-grained queue information, as required for time-sensitive applications. The remainder of this section gives a brief overview of relevant extensions of routing protocols to enhance the discovery and signaling of QoS parameters in L3 networks.

Guerin et al. [8] proposed path selection algorithms for QoS routes as extensions to the OSPFv2 protocol, focusing mainly on bandwidth requirements. One of the proposed path selection algorithms computes QoS paths on demand; the details are omitted here. However, this algorithm implies that every router should have an up-to-date view of the network, i.e., every router should know the available bandwidth of every link. Because continuously distributing this information is expensive, and periodic updates might result in missing important changes between update periods, a hybrid approach is presented, combining time-based (periodic) updates with event-based updates. Event-based updates are distributed immediately whenever the change (since the last update) of the available link bandwidth exceeds a certain threshold. The distribution of updates fits perfectly within the OSPF-TE extensions [9], which use opaque LSAs to distribute additional information.

Song et al. [10] proposed a flood-and-prune 'exploration' QoS routing protocol. This protocol finds feasible paths in the network in parallel by flooding a probe (connection request) and reserving resources at intermediate nodes. Before reserving resources at an output port and forwarding the probe, a QoS admission test is performed first. For example, if a probe carries a bandwidth requirement, it will only be forwarded through ports that have enough unreserved bandwidth. In addition to probes, two other types of packets are used: confirmation packets, which signal that a feasible path is found, and prune packets, which signal that no feasible path could be found. Both of these can be seen as a reply to a probe. The following rules capture the full working of the protocol:

1) A probe is forwarded through ports where the QoS admission test succeeds. A probe is never sent through the port from which it came (upstream).
2) When a probe reaches its destination, a confirmation packet is sent back.
3) When a confirmation packet is received at a node, it frees all unnecessary allocated resources at the other output ports. The confirmation packet is sent further upstream until the first node is reached.
4) If all output ports fail the QoS admission test, then a prune packet is sent back upstream.
5) If a node receives a prune, it frees the allocated resources at that port. If a node received a prune for all probes that it sent out, then it sends a prune packet upstream.

The main problem of this protocol is over-reservation due to

flooding. Therefore, it is extended with a fast-prune mechanism, which allows freeing resources faster, and works as follows:

- If a probe is not the first to arrive at a node, a prune is sent back immediately.
- When a confirmation packet is received at a node, a prune-forward packet is sent through every port where a response was not yet received.
- When a prune-forward packet is received, all resources are cleared and it is forwarded again through every port where a response was not yet received.

III. Building blocks

The route discovery protocols proposed in sections IV and V rely on common building blocks to guarantee delay and to ensure that the control traffic itself receives sufficient resources to enable adequate operation. This functionality is briefly described below.

A. Delay guarantees

The routing protocols are based on delay budgeting mechanisms used in Chameleon to guarantee worst-case delays. Every queue-level topology link is therefore assigned a budget, such that the worst-case delay d_l of a queue-level topology link l is now given by the sum of its budget and the propagation delay of the corresponding physical link. The worst-case (end-to-end) delay of a path of queue-level topology links is then simply computed as the sum of the worst-case delays of the links.

To keep the actual worst-case queueing delays below their budgets, some form of admission control and resource reservation is needed. Network calculus is used to keep track of the reserved resources via affine arrival curves (which model incoming traffic) and service curves (which model the link service), allowing it to compute worst-case queueing delays at every queue. By keeping these delays below their budgets at every queue, the queueing delays are effectively bounded.

A flow is characterized by a delay requirement d_f and an affine arrival curve, consisting of a burst and rate parameter. Per network calculus, the arrival curve of a flow changes after it is embedded in a queue, i.e., its burst increases. The exact formula to compute this burst increase can be found in [3].

B. Control channel

In a centralized context such as in [2], the control traffic does not influence the critical data. However, this is not the case in a distributed setting, where control traffic and data traffic use the same links, therefore influencing each other. This is why the routing protocols in this paper make use of an additional mechanism to overcome the problem of control traffic overhead. In particular, a limited amount of resources (burst and rate) will be reserved at every physical link between two routers at all times for the control traffic itself. This can be seen as a special flow: the control flow. By allocating the control flow to the highest-priority queue, the exchange of control messages (either for discovery or signaling) will not be hampered by congested data channels.

IV. Exploration-based QoS routing protocols

The basic principles of the exploration-based protocols presented in this section are based on the ideas of the flood-and-prune protocol in [10]. The difference is that resources for a flow are now reserved at a queue-level topology link, i.e., a queue, instead of at an output port. Thus, a QoS admission test will be performed on every queue-level topology link upon receiving a probe. The probes now carry flow parameters in the form of a delay requirement d_f, and burst and rate parameters.

Because an exploration-based protocol is only concerned with finding QoS paths, it is employed on top of a standard, non-QoS routing protocol such as OSPF. This protocol exposes information about the network, which can guide the exploration process. Useful information includes the distance to the destination hop, which will be used to prune paths and in the admission test.

As mentioned before, these protocols suffer from over-reservation because of their greedy nature. While this might not be a problem when resources are only reserved after a full path has been found, it adds extra complexity for certain edge cases. For example, a situation might arise where a path is needed for two flows. Under high network contention, two paths could be found that both make use of the same resources, such that a potentially inefficient reroute of one of the flows is needed. Exploration schemes focusing on re-optimizing such routes are left out for future work.

The remainder of this section describes the unique properties and innovative elements of the novel exploration-based protocols of this paper.

A. Path pruning

In large networks, many possible paths from a given source to a destination node exist. However, because of the problem of over-reservation, it is a good idea to limit the search space, such that a probe will only travel through a certain part of the network. The idea is that if the destination is h_0 hops away from the source (as per the underlying routing protocol), then the explored paths may not have a length that exceeds $\frac{4}{3}h_0+3$. This formula is an attempt to limit the search space to a reasonable size, assuming that longer paths probably waste too many resources, as the burst increases per hop. The added constant is used to ensure that there is still a reasonable search space for nearby destinations, e.g., a single hop away. This principle can be generalized to arbitrary link costs as well.

B. Admission test

A probe now carries the following additional statistics that are used by the admission test:

- The accumulated worst-case delay d_{acc} up until this point,
- An initial estimate of h_0 hops from the source to the destination, and
- The hop count h of the path up until the current node.

Based on these statistics, and an additional estimate by the underlying routing protocol that the destination is only h_{est} hops away if l is taken, the flow is allowed to be embedded at the queue-level topology link l if

1) $d_{acc} + d_l \leq d_f$, and
2) The flow can be embedded in l, and
3) A prune did not come from the physical link, and
4) $h + h_{est} \leq \frac{4}{3}h_0 + 3$ (path pruning), and
5) The probe did not come through the corresponding physical link, and
6) The flow is not already embedded in a queue-level topology link that belongs to the same physical link, and
7) The protocol chooses to do so.

The first three criteria are self-explanatory: they ensure that the delay requirement of the flow is respected, that there are enough resources left on the link, and that probes are not forwarded through paths that previously proved not to meet necessary latency requirements, and are likely to fail again. The fourth and fifth criteria enforce respectively path pruning and that probes don't travel in loops. Additionally, the sixth rule is added to prevent an exponential explosion of the exploration packets that are sent into the network. Otherwise, there would be a probe for every possible queue combination on a single physical path. Finally, the last criterion provides freedom to the protocol on how to implement the exploration routine, as the protocol is now not forced to forward probes via all possible links at the same time. Once a link passes the admission test, the probe that is sent through it will be updated, i.e., its accumulated worst-case delay, burst, and hop count will be updated.

C. Link heuristics

There might be several queue-level topology links belonging to the same physical port that pass the admission test. However, a flow cannot be embedded at two queue-level topology links of the same physical output port. Therefore, a link heuristic is used by the protocol to select one out of the possible candidates. Several heuristics are proposed in the following subsections.

1) Random: Select a link randomly, in the hope that there will be a good balance between low-priority queues and high-priority queues.

2) Greedy: Always choose the link with the least delay, this keeps the burst consumption low at intermediate nodes but could waste resources for flows with stricter delay requirements.

3) Honest: Try to keep the delay consumption relatively equal across all intermediate nodes. For example, for a flow that has a delay requirement of 32ms, the honest heuristic will prefer a link with a delay of around 4ms, given that the accumulated delay is 16ms and the destination is 4 hops away.

D. Reacting to link failures

When a link failure is detected, a path that goes through that link is cleared, i.e, its resources are freed at every node from the broken link to the source, and from the broken link to the destination. This is accomplished by explicitly sending FREE packets along the path. The rerouting of a flow then starts at the source node again.

E. Breadth-first exploration

The breadth-first (BF) exploration variant is similar to the flood-and-prune protocol in [10], which searches all paths in parallel. An equivalent fast-prune technique is also used to counter over-reservation.

F. Depth-first exploration

The depth-first (DF) exploration variant will only explore one path at a time. This means that, upon receiving a probe, a single link must be chosen out of all possible links, which could belong to multiple physical ports. Therefore, it uses a port heuristic first, after which a link heuristic is used to embed the flow into one of the possible links of the chosen port. The selected port will be the one with the lowest cost to the final destination, and which contains at least one link that passes the admission test.

It is also possible to perform n-DF exploration, where the best n paths are explored in parallel. This is left for future work.

V. Link-state QoS routing protocols

The central idea of the link-state (LS) QoS routing protocol is that every router floods messages carrying the state of given links in the network topology. All routers receive these messages and collect them into a database that maintains a global view of the current state of the network. Whereas typical LS protocols (e.g., OSPF or IS-IS) focus only on physical topology link states, the considered LS protocols here also include queue states. This allows every router to compute a complete feasible path from source to destination on the queue-level topology upon a flow request. A new type of LSA, which contains up-to-date queue information, is introduced for this purpose: QoSLSAs. These are also flooded, such that every router maintains a QoSLSA database and a queue-level topology map. Note that these QoSLSAs also fit perfectly within the OSPF-TE extensions [9].

A. Computing constrained paths

Two algorithms based on Dijkstra's shortest-path algorithm [11] are used to compute paths on the queue-level topology that satisfy the delay constraints of flows. Firstly, simple paths with minimal delay can be computed, and, secondly, the LARAC algorithm [4] is also supported. However, these basic algorithms are slightly adjusted, as the computed path should have enough free resources. This is accomplished by keeping track of the flow parameters and updating those after visiting an edge, and by never visiting an edge if the flow cannot be embedded there.

B. Distributing queue states

Similar to [8], a hybrid system of event-based and time-based updates will be employed to make a trade-off between accuracy and efficiency. In particular, event-based updates will be distributed if the total allocated bandwidth at a link is more than 5% of the total link bandwidth since the last update.

The idea of the event-based updates is to account for rapid and significant changes of the available resources. When flows use a rather limited amount of resources, the time-based updates should provide a reasonably accurate view of the network, while keeping the overhead low. When a certain flow requires a lot of resources, that assumption is not valid anymore. The solution is to simply send an extra update.

It is also possible to send an update when the burst consumption in a queue changes significantly. This however requires a little more state management and is not yet incorporated into the protocols to keep the implementation simple.

C. Embedding flows

Due to the unavoidable inaccuracies of the queue-level topology map, a router might try to embed a flow in a path using resources that are already gone. Therefore, it needs to be checked at every intermediate router if the resources are still available. This is accomplished via forwarding an EMBED packet along the path, which contains the flow parameters and the full path. The situation where a flow cannot be embedded because of these inaccuracies is called a 'collision'. If such a collision occurs, it is signaled back to the source router, which may opt to compute a new path and try again. The router at which the collision took place will also send out a new QoSLSA, such that a collision is now less likely to happen at this node. When no collisions occurred up to the destination router, a confirmation is signaled back to the first router. When this confirmation reaches the source, the critical traffic is allowed to be sent out.

D. Reacting to link failures

Routers can detect link failures implicitly via standard LSAs. Because every router has full knowledge of all paths of the flows passing through it, resources can be freed immediately at intermediate nodes. When the first router in a path for a flow detects a link failure on the path, it will try to re-embed the flow.

VI. Evaluations

A. Simulation setup

The evaluation of the protocols is carried out in the simulation environment OMNeT++ [12] (v5.6.2), extended with a modified version of the INET framework [13]. Both this modified version and the protocol implementations are available as open source [14] [15]. All of the experiments were performed on a laptop with an Intel core i7 7500u processor and 8 GB RAM running on Ubuntu 20.04. All simulation experiments, except if stated otherwise, are performed on the large-scale COST266 network [16], which consists of 37 routers and 57 two-way links that span a large part of Europe. All links have a bandwidth of 1Gbps, and all routers have 8 priority queues for QoS traffic, with a buffer of 97 kB per priority queue. The budgets of the 8 queues are fixed at respectively 0.1 ms, 0.5 ms, 1 ms, 2 ms, 4 ms, 8 ms, 16 ms, and 32 ms. These are based on the budgets in [2], and the idea is that they cover both end-to-end delay requirements of around a few milliseconds to a few hundred milliseconds (delay diversity).

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
Random	[10-200] ms	[100-3000] B	[0.1-16] Mbps

TABLE I: The list of applications used for the evaluation of the proposed protocols.

The simulations make use of a flow requester. This is a simulation tool that will set up new flows into the network, i.e., it creates new applications that will send a flow request to their access router to establish a path. Tab. I, which is based on the applications in [2], depicts the different types of applications and their characteristics. Whenever a new application is created, its type and characteristics will be generated randomly using this table. The source and destination host are also chosen using a uniform random distribution, and the flow requester will create new applications at a rate of 10 flows per 10 ms, which corresponds to a rate of 1000 flows per second.

B. Network utilization

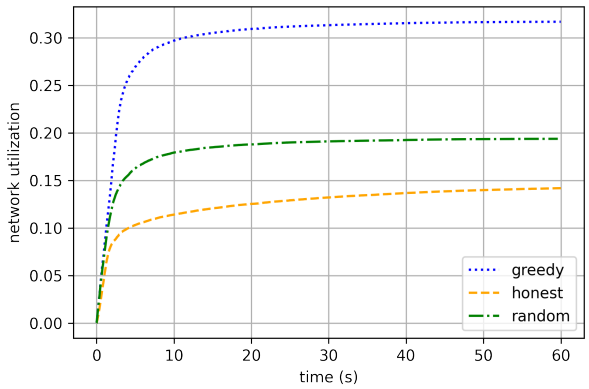
The network utilization of the critical traffic is measured by the total bandwidth that is allocated to it, divided by the sum of the bandwidth of all links in the network. This experiment takes 60 s, during which the flow requester creates new applications demanding a flow, and it is repeated 10 times to be statistically representative.

Network utilization is an interesting performance metric because it reflects the efficiency of the protocols regarding pathfinding. As mentioned earlier, the admission control component will make sure that flows are allowed to be embedded in a queue only if there are enough resources left. Thus, the more bandwidth a protocol can allocate to critical traffic, the more efficiently it uses the queue resources, resulting in more critical traffic that can flow through the network.

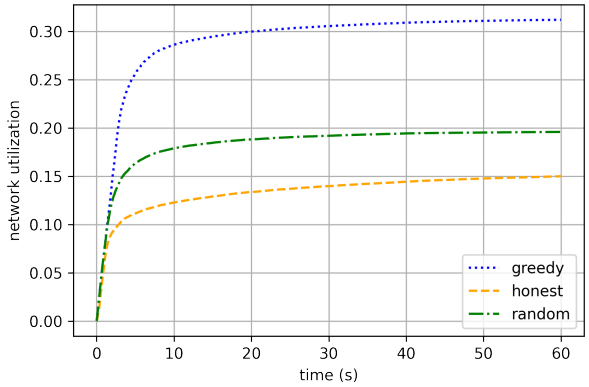
For the exploration protocols in Fig. 2a and 2b, it is immediately clear that the greedy link heuristic outperforms the others. This can be explained by the fact that choosing a low-delay link will keep the burst consumption low at intermediate nodes, therefore leaving more resources open for future flows. For the LS QoS protocols, Fig. 2c illustrates that the min-delay algorithm outperforms the LARAC algorithm. Similarly, the min-delay algorithm will keep the burst consumption low in the intermediate nodes, while the LARAC algorithm might differ from this.

C. Delay and reliability

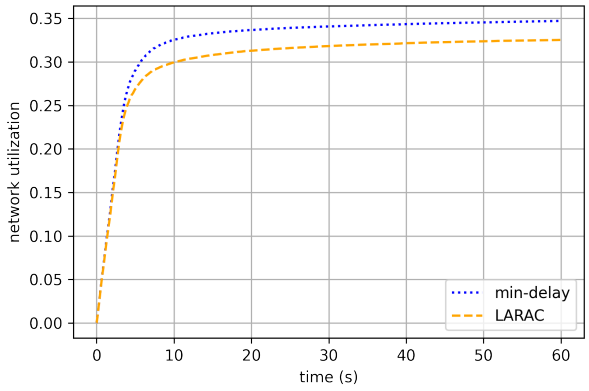
The delay and reliability guarantees of all protocols build on the same foundations and are evaluated as follows. Flows are created in the network for a duration of 60 s, and they are embedded using the LS min-delay protocol. Based on the results of section VI-B, most of the available resources will be allocated to critical traffic after 60 s, i.e., network contention



(a) BF exploration protocol



(b) DF exploration protocol



(c) LS QoS protocol

Fig. 2: Experimental results comparing network utilization for different protocols.

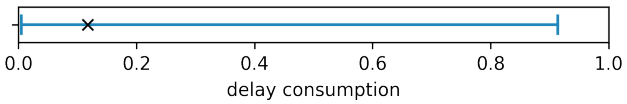


Fig. 3: Experienced end-to-end delay consumption range and mean

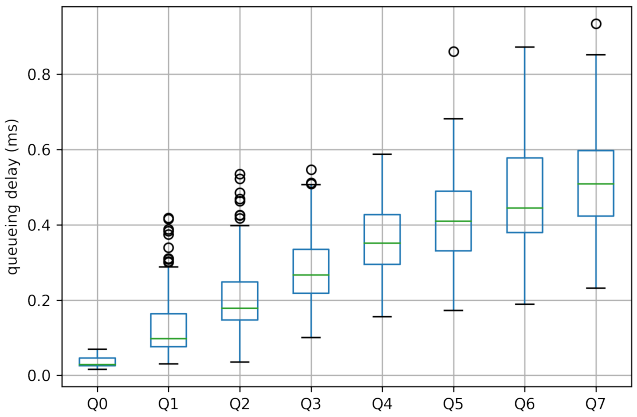


Fig. 4: Experienced worst-case queueing delays, ordered from high priority (QO) to low priority (Q7)

will be high. After this initial period, each of the accepted flows continues to send its traffic for a full period of 120s. While this period does not capture the full lifetime of a realistic flow, it should be enough to make flows experience heavy or even worst-case queueing delays in the presence of other critical traffic.

All experiments successfully demonstrated the absence of packet loss. The experienced end-to-end delay divided by the delay requirement of the applications (delay consumption) is depicted in Fig. 3. A delay consumption greater than 1 implies a missed deadline, which never occurred. This means that the system of network calculus and admission control can indeed provide guaranteed delay bounds and reliability. Another observation is that most of the time, the time-critical packets arrive much sooner than their imposed deadline. Indeed, since the only guarantees are worst-case delay bounds, and a worst-case scenario over the whole path is quite rare, packets will typically arrive much sooner. When looking at the worst-case queueing delays in Fig. 4, it is clear that the delay budgets are respected at all times.

D. Link failure

The link failure handling of the protocols is rated by their ability to re-embed the flows, and by the number of packets lost. There will always be a transitional period (rerouting window) between the link going down and the full confirmation of a new path, during which packets will get lost. Note that a critical packet that does not have a full path is considered lost here. Three different scenarios are evaluated 5 times: a random link failure at a low network load (1), at a medium network

Protocol	Reroute success rate	Packets lost per flow
LS Min-Delay	73%	170
Greedy BF	70%	528
Greedy DF	92%	27

TABLE II: Failure handling performances for scenario 1

Protocol	Reroute success rate	Packets lost per flow
LS Min-Delay	20%	33
Greedy BF	30%	286
Greedy DF	33%	19

TABLE III: Failure handling performances for scenario 2

load (2), and high network load (3). These network loads are based on the results of section VI-B, i.e., the scenarios differ regarding the network utilization of the critical traffic and they range from around 17% to 25% to 30%. The results are given in tables II, III, and IV.

The exploration protocols are significantly more successful in rerouting broken flows. This could be because there is a peak of rerouting requests upon a link failure, resulting in a performance drop for the link-state protocol due to the increased inaccuracy of the queue-level topology map. This is not a problem for exploration-based protocols, which always make use of the latest local information at every router. On the other hand, the BF exploration protocol loses a lot of packets compared to the other protocols, which might indicate that it takes more time to reroute flows.

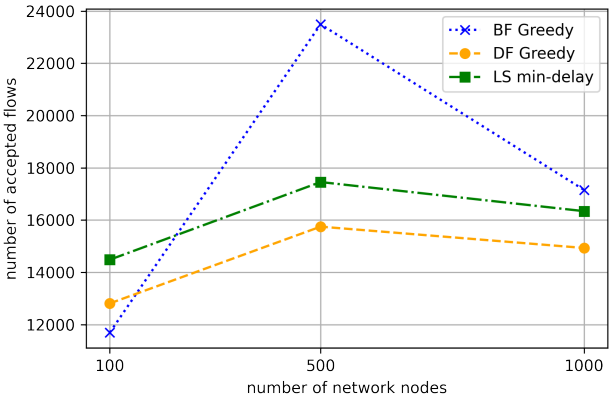
E. Scalability

The scalability of the protocols is evaluated by the number of accepted flows and the flow embedding times. This experiment is performed by setting up flows in three large-scale networks for a duration of 25 s. These three networks are generated using IGEN [17], and consist of 100, 500, and 1000 nodes each. These networks cover the same geographical area but differ in the number of nodes and links. This means that a network with more nodes is more connected, and has a larger search space between two given nodes. The results are depicted in Fig. 5.

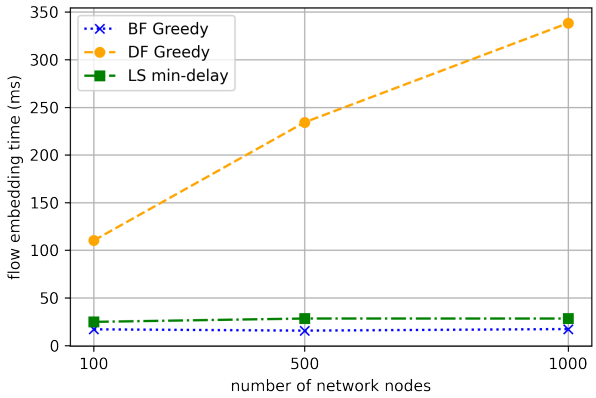
It might be observed that the DF exploration protocol does not scale well to larger networks, as its flow embedding times become excessively long. On the other hand, the flow embedding times of the BF and LS protocols remain almost constant. Fig. 5a illustrates that the BF exploration protocol makes the best use of the added capacity in the network when the number of nodes increases. This is probably due to its parallel capabilities, such that it can explore more and better paths in a smaller amount of time than the DF variant. Moreover, unlike the LS protocol, the BF exploration protocol does not suffer from state management overhead

Protocol	Reroute success rate	Packets lost per flow
LS Min-Delay	5%	5
Greedy BF	20%	285
Greedy DF	21%	36

TABLE IV: Failure handling performances for scenario 3



(a) Number of accepted flows per protocol



(b) Mean flow embedding time per protocol

Fig. 5: Experimental results comparing scalability of different protocols.

and uncertainty, which become greater if the network grows. However, a higher number of nodes does not necessarily imply a higher number of flows, as observed in Fig. 5a. This could be because of the longer paths in bigger networks. Since the flow's burst increases at every hop per network calculus, more burst is needed for a single flow compared to a smaller network. The result is that fewer flows can be embedded into the network when this increased burst becomes too much.

F. Link-state vs. exploration-based

Based on the simulation experiments presented in this section, none of the protocols stands out as the best overall. However, the LS protocol consistently performs well across all aspects, whereas the performance of the exploration protocols is more variable. This consistency, coupled with the fact that it is based on the well-established principles of standard LS protocols, makes the LS protocol the most promising among the presented protocols.

VII. Conclusions

This paper has presented three novel routing protocols that can guarantee bounded end-to-end delay for real-time applications without experiencing packet loss by exploiting network-wide queue information. All of the proposed protocols are parameterizable to suit different scenarios and can be easily extended with new heuristics and path computation algorithms. Experimental results demonstrate the potential of these protocols in large-scale networks enabling real-time and reliable networking.

The performed work and proposed mechanisms also create opportunities for several enhancements and future research. The delay budgets of the priority queues are crucial in guaranteeing delay bounds. However, they put severe restrictions on the burst length and bandwidth that can be reserved in every queue. Future research might focus on investigating the impact of those budgets in combination with the application requirements to compute optimal delay budgets. Other interesting topics include evaluating more complex heuristics, and hybrid protocols combining the strengths of the LS QoS routing protocol with an exploration-based QoS routing protocol, which could further improve the scalability of the LS routing protocol.

VIII. Acknowledgments

This research was funded by the Flemish FWO SBO S003921N VERI-END.com (Verifiable and elastic end-to-end communication infrastructures for private professional environments) project, by the FWO, Belgium project under grant agreement #G055619N and the Flemish Government, Belgium under the "Onderzoeksprogramma Artificiele Intelligentie (AI) Vlaanderen".

References

[1] Ahmed Nasrallah et al. "Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research". In: *IEEE Communications Surveys & Tutorials* 21.1 (2018), pp. 88–145.

[2] Amaury Van Bemten et al. "Chameleon: predictable latency and high utilization with queue-aware and adaptive source routing". In: *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*. 2020, pp. 451–465.

[3] Amaury Van Bemten and Wolfgang Kellerer. "Network calculus: A comprehensive guide". In: (2016).

[4] Alpar Juttner et al. "Lagrange relaxation based method for the QoS routing problem". In: *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*. Vol. 2. IEEE. 2001, pp. 859–868.

[5] Thomas Stüber et al. *A Survey of Scheduling in Time-Sensitive Networking (TSN)*. 2022. DOI: 10.48550/ARXIV.2211.10954. URL: https://arxiv.org/abs/2211.10954.

[6] Norman Finn and Pascal Thubert. *Deterministic Networking Problem Statement*. RFC 8557. May 2019. DOI: 10.17487/RFC8557. URL: https://www.rfc-editor.org/info/rfc8557.

[7] Andrew G. Malis et al. *Deterministic Networking (DetNet) Controller Plane Framework*. Internet-Draft draft-ietf-detnet-controller-plane-framework-02. Work in Progress. Internet Engineering Task Force, June 2022. 18 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-detnet-controller-plane-framework/02/.

[8] Roch A Guerin, Ariel Orda, and Douglas Williams. "QoS routing mechanisms and OSPF extensions". In: *GLOBECOM 97. IEEE Global Telecommunications Conference. Conference Record*. Vol. 3. IEEE. 1997, pp. 1903–1908.

[9] Derek M. Yeung, Dave Katz, and Kireeti Kompella. *Traffic Engineering (TE) Extensions to OSPF Version 2*. RFC 3630. Oct. 2003. DOI: 10.17487/RFC3630. URL: https://www.rfc-editor.org/info/rfc3630.

[10] Jun Song, Hung Keng Pung, and Lillykutty Jacob. "A multi-constrained distributed QoS routing algorithm". In: *Proceedings IEEE International Conference on Networks 2000 (ICON 2000). Networking Trends and Challenges in the New Millennium*. IEEE. 2000, pp. 165–171.

[11] Edsger W Dijkstra et al. "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1 (1959), pp. 269–271.

[12] Andras Varga. "OMNeT++". In: *Modeling and tools for network simulation*. Springer, 2010, pp. 35–59.

[13] *INET Framework: An open-source OMNeT++ model suite for wired, wireless and mobile networks*. https://inet.omnetpp.org. Accessed: 2022-05-19.

[14] Jakob Miserez. *Routing protocols exploiting queue information for deterministic networks - INET*. URL: https://github.com/jakobmiserez/inet.

[15] Jakob Miserez. *Routing protocols exploiting queue information for deterministic networks - implementation*. URL: https://github.com/jakobmiserez/thesis.

[16] S. Orlowski et al. "SNDlib 1.0–Survivable Network Design Library". English. In: *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*. http://sndlib.zib.de, extended version accepted in Networks, 2009. Apr. 2007. URL: http://www.zib.de/orlowski/Paper/OrlowskiPioroTomaszewskiWessaely2007-SNDlib-INOC.pdf.gz.

[17] Bruno Quoitin et al. "IGen: Generation of router-level Internet topologies through network design heuristics". In: *2009 21st International Teletraffic Congress*. IEEE. 2009, pp. 1–8.