Real Time Local Re-Routing to limit Queuing Delay exploiting SRv6 and Extensible In-Band Processing

Marco Polverini, Davide Aureli, Antonio Cianfrani, Francesco G. Lavacca, Marco Listanti

Department of Information engineering, Electronics and Telecommunications (DIET)
University of Roma "Sapienza" - Via Eudossiana 18, 00184 Roma, Italy
Telephone: +39 0644585371, e-mail: name.surname@uniroma1.it

***Abstract*—In this paper we introduce *QLR*, a per-router control agent that aims at reducing the occupancy of the local buffers by performing re-routing operations. The Segment Routing architecture is exploited to manage the uncoordinated selection of re-routing performed by different nodes, thus avoiding the creation of routing loops, while the Extensible In-band Processing is used to allow the network nodes to have a detailed and updated view of the wide network status. Data and control plane programmability are considered to define a prototype implementation of *QLR* that allows for the execution of a preliminary performance evaluation and proof-of-concept. From the conducted experiments has emerged that *QLR* can effectively reduce the maximum queue occupancy and end-to-end delay up to 43% and 63%, respectively.**

***Index Terms*—Segment Routing, Extensible In band Processing, Programmable Networks, Routing Optimization**

I. Introduction

The advent of data and control plane programmability paradigms [1] has paved the way to the definition of new tools and procedures to operate, manage and monitor the network infrastructures more efficiently than ever before. In particular, data plane programming models have allowed the introduction of IN-band network Telemetry (INT) strategies [2], which are able to overcome the limits of the classical centralized monitoring frameworks, such as the scalability, low granularity and frequency of the statistics updates. At the same time, the opportunities for the definition of flexible and sophisticated customized control procedures [3] have widely increased, thanks to the control plane programmability.

As for the network capabilities, also user requirements have dramatically changed with respect to the past. Ultra Reliable and Low Latency Communications (URLLC), massive Machine Type Communications (mMTC) and enhanced Mobile Broadband (eMBB) are some examples of use cases [4] where the Quality of Service (QoS) requirements are pushed further ahead with respect to those of the classical Internet applications. Thus, new challenges emerge for Internet Service Providers (ISPs) to introduce advanced control mechanisms to deal with such heterogeneous and stringent service requirements.

Given the availability of these new network programming models, in this work we do a preliminary investigation about the possibility of defining a network layer strategy to mitigate the effect of sudden congestion events. Differently to other higher layer solutions, such the Bottleneck Bandwidth and RTT (BBR) [5] or the Low Latency Low Loss Scalable Throughput (L4S) [6] that rely on transport layer mechanism, handling the congestion at the network layer has the advantage to not require actions to end users (which might not cooperate). Alleviating the congestion level of network links is effective for reducing the queuing delay and the packet loss probability, thus represents a concrete answer to the requirements of time sensitive applications. Classically, congestion events are handled at different layers of the TCP/IP stack using different approaches, such as at the transport one, where reactive congestion control mechanisms are introduced in the TCP protocol to throttle traffic flows if the network is overloaded, or at the network layer, where proactive Traffic Engineering routing strategies are defined to load balance the link load and reduce the risk of congestion. All the aforementioned mechanisms work in a time scale that is critical for delay sensitive applications, there is a great deal of space for the definition of new solutions.

The strategy here proposed is named *QLR* (Queue Length based Routing) and aims at performing re-routing operations based on the occupancy of the buffers associated to the network links so that to avoid that queues grow over a given threshold, thus limiting the maximum queuing delay. Through the Extensible In-Band Processing (EIP) [7], all nodes have a wide and updated view of the network status, either in terms of queue occupation and link load. Based on such information, each ingress node can locally compute alternative routes for incoming traffic flows, i.e. external traffic injected in the network by the ingress node, so that to alleviate the congestion on overloaded links. The re-routing selection is performed locally to reduce the *QLR* execution time, thus minimizing its time scale applicability. The local re-routing requires the availability of a source routing approach, thus *QLR* is defined in a Segment Routing over IPv6 (SRv6) [8] network. The main contributions of our preliminary study are:

- to identify the technology required to support the *QLR* solution;
- the definition of a simple local control algorithm to select the re-routing operations;
- the creation of a prototype of *QLR*;
- a preliminary evaluation conducted in an emulated envir-

978-3-903176-51-5

onment.

The rest of the paper is organized as follows. In Sec. II the state of the art is reviewed to provide an overview of network performance optimization frameworks based on in band telemetry information. Furthermore, it introduces the background knowledge of all the *QLR* enabling technology. Sec. III introduces the main working principle of *QLR*, the system model and the algorithmic aspects. A preliminary performance evaluation is proposed in Sec. IV, while Sec. V concludes the paper.

II. Related Works and Background

The advent of programmable networks have increased the interest of the research community towards nearly real time Traffic Engineering operations, i.e. at seconds/milliseconds time scale. The challenge is to dynamically adjust the routing to optimize the network performance reacting to, or anticipating, traffic spikes that can reduce the QoS performance (e.g., in terms of end to end delay, packet loss, etc.). As an example, in [3] authors leverages a Deep Reinforcement Learning (DRL) based algorithm applied on network statistics gathered by means of IN band Telemetry (INT), which is executed by an SDN controller. The idea is to optimize the routing, in a time scale of the order of seconds, according to the information about the current network status that is gathered by means of INT probes. Clearly, frequent rerouting operations can degrade the QoS perceived by end users, due to effects such out of order packets or temporary loops. To take this side effect into account, in [9] the *network disturbance* metric is defined and considered while selecting the rerouting operation, by means of a centralized algorithm based on DRL. Also in this case, the time scale of the solution is in the order of seconds/minutes.

Centralized solutions have the advantage to allow consistent routing decisions (there is a single decider with a global view of the current status) to be taken [10]. SR Flex Algo [11] is also an interesting solution to enforce low latency paths on incoming flows. In this case, instead of having a central controller, each node computes its own segment lists with the aim of optimize a performance metric (e.g., e2e delay), by enforcing an algorithm over a network representation that is distributed by means of an IGP protocol. Nonetheless, frameworks that rely on these type of architecture cannot operate to very lower time scales (milliseconds), thus also decentralized solutions have been proposed. Two relevant examples are: i) [12], where a Control Agent driven by a DRL algorithm is placed on a network node and, according to a local view, takes rerouting decision with the goal of keep the utilization of the local links below a threshold value; and ii) [13] where a new behavior for the SRv6 network programming model is defined, which is able to steer the incoming traffic flows over the path that currently ensures the largest throughput. In this case the path is calculated locally by a simple heuristic that exploits global information gathered through INT probes.

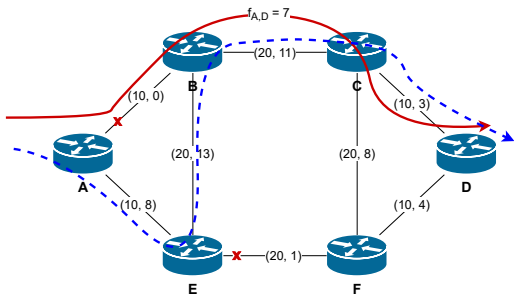


Figure 1. Illustration of the working principle of *QLR*.

A. QLR *Enabling Technologies*

SRv6 is an overlay architecture that implements the source routing paradigm, realized on top of an IPv6 underlay, that is in charge of providing connectivity among nodes. In an SR domain each node is identified by a Segment IDentifier (SID), that is represented by an IPv6 address. When an IP packet enters an SRv6 domain, the ingress node encapsulates it inside an outer IPv6 packet that carries the SR Header (SRH), and enforces a segment list on it, which specifies the set of middle-points that the packet has to traverse.

EiP is an extensible architecture that extends the IPv6 one, which allows to perform complex functions directly in the network, instead of in the end systems. Common use cases are: advanced monitoring, semantic routing, deterministic networking, etc. EiP information is included in a standard message having a header and a set of Information Elements (IEs). For each of these use cases, different IEs are defined in [14], and new ones can be defined if needed. EiP messages can be included in a packet as an option for IPv6 Hop by Hop extension header, or as a Type Length Value (TLV) in the SRv6 header.

III. Queue Length based Routing

The *QLR* algorithm aims at keeping the queue occupation below a critical threshold, by means of re-routing operations that are decided by *Control Agents* (CA) placed on each ingress node. The working principle of *QLR* is reported in Fig. 1, showing a network topology with the relevant statistics. In particular, each link is labeled with two numbers representing the maximum and available capacity, while the interfaces in red indicate that the current number of packets in the buffer exceeds the threshold. Furthermore, Fig. 1 shows that the current volume of the traffic flow $f_{A,D}$ injected in the network by node A and leaving it through node D is equal to 7 Mbps. In this scenario, the aim of the CA placed at the node A is to alleviate the congestion on the link between nodes A and B. This result is achieved by re-routing the traffic flow $f_{A,D}$ over the alternative path represented by the dashed blue line in Fig. 1.

The selected alternative path of the example must be: i) a congestion free path, i.e. its links have an available bandwidth lower than the size of the rerouted flow and the buffers occupation is kept below the threshold, and ii) a loop-free path. To be

compliant with the first property, considering that the algorithm is locally executed on a router, a mandatory requirement for the practical implementation of *QLR* is that each CA knows the network status, in terms of buffer occupation and available bandwidth. Moreover, the CA also needs to know the size of the traffic flow to reroute. As regard to the loop-free property, it is a classical requirement of any routing strategy and it is not a straightforward feature in a scenario where local rerouting actions are performed in different nodes in an uncoordinated manner.

All these issues can be managed in the context of an SRv6 network, that represents a key enabling technology for *QLR* implementation. In fact, in SRv6 each node is equipped with traffic counters [15] that can be used to measure incoming traffic flows, thus allowing each node to know the volume of the flows it is injecting in the SRv6 domain. Furthermore, SRv6 is one of the main building block for the EiP [14]. EiP allows the insertion of telemetry information directly inside packets that carry user data, thus drastically increasing the frequency of the updates without the need to perform a dedicated monitoring protocol. Clearly, there is no guarantee that the information gathered by different nodes are time aligned, i.e., each router can have a different view of the network status, thus making the process of selecting routing strategy challenging with respect to the possible creation of loops. Nonetheless, in case of *QLR* this problem does not arise due to the source routing paradigm implemented by SRv6, i.e., the entire path followed by a traffic flow is enforced by the ingress node of the SRv6 domain. In this way, the path followed by a flow cannot be modified by different network nodes.

A. System Model

Let us consider a network (e.g, access, datacenter, ISP) operating according to the SRv6 architecture. The network topology is represented by the graph $\mathcal{G}(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ and $\mathcal{L}$ are the set of the N nodes and L links, respectively. Each link $l_{i,j}$ is associated with a tuple of attributes $\sigma_{i,j}(t) =< q_{i,j}(t), c_{i,j}(t) >$ representing the queue length and the available capacity at time t. The set of links leaving from node i is referred to as $\mathcal{L}_i^-$. The network status at time t is described by the set $\Sigma(t) = \{\sigma_{i,j}(t), \forall l_{i,j} \in \mathcal{L}\}$. In order to reduce the overhead due to the insertion of EiP headers in the data packets, the network status is collected every δ_T seconds (i.e., the time axis is slotted). The volume of the traffic flow entering the network from the ingress node i and leaving it through the egress node e at time t is referred to as $f_{i,e}(t)$.

The CA located at node i is indicated with the symbol α_i. Its goal is to constantly monitors the network status $\Sigma(t)$ through the EiP. Furthermore, by means of the SR traffic counters, α_i constantly measures the volume of traffic flows it injects in the network. Node i maintains a set $\Pi_{i,e}$ of alternative paths for each possible egress node e. At each time slot there is a single working path (indicated as $\pi_{i,e}^w$), enforced at the ingress node, thus the routing of a flow can be quickly changed by the ingress node; the working path $\pi_{i,e}^w$ is defined as a vector of length L, whose l-th component ($\pi_{i,e}^w(l)$) represents the fraction of traffic sent over the link l. In a similar way, the k-th alternative path is referred to as $\pi_{i,e}^k$.

Algorithm 1 Pseudo code of the *QLR* algorithm performed by CA α_i.

Require: $\Sigma_i(t)$, $f_{i,e}(t), \forall e \in \mathcal{N}$, $\Pi_{i,e}, \forall e \in \mathcal{N}$
1: sort links in $\mathcal{L}_i^-$ in ascending order w.r.t $c_{i,j}(t)$ and store in list *sorted_links*
2: create vector $c^{\text{tmp}} = \{c_{i,j}(t), \forall l_{i,j} \in \mathcal{L}\}$
3: **for all** $l_{i,j}$ in *sorted_links* **do**
4: **if** $q_{i,j}(t) > \tau$ **then**
5: sort flows $f_{i,e}(t)$ such that $\pi_{i,e}^w(l_{i,j}) > 0$ in descending order w.r.t $f_{i,e}(t)$ and store in list *sorted_flows*
6: **for all** $f_{i,e}(t)$ in *sorted_flows* **do**
7: $\pi_{i,e}^w =$*select_path*$(\Pi_{i,e}, c^{\text{tmp}}, \Sigma_i(t))$
8: update c^{tmp}
9: **if** $c_{i,j}^{\text{tmp}} > 0$ **then**
10: go to line 4

B. QLR Algorithm

In this subsection we detail the control logic and the actions performed by the CA α_i. We recall that the same process is executed at each ingress node of the network in an uncoordinated manner. The choice of having a per-router CA and of the lack of synchronization among them is due to the low time scale at which *QLR* have to work. As previously stressed, thanks to the adoption of the source routing paradigm each CA can reroute a traffic flow avoiding loops. Nonetheless, the different flows share the same network resources, then it is possible that two CAs take conflicting decisions, i.e. moving the congestion from a link to a different one. For instance, with reference to Fig. 1 the nodes A and E can simultaneously decide to reroute the traffic flow directed to the destination D using an alternative path passing for the link $l_{E,B}$, that would become congested. We also have to consider that the status $\Sigma_i(t)$ can be significantly different from $\Sigma_j(t)$.

It is then important to limit possible conflicting events, where the term event refers to a routing operation performed by the CA α_i as a consequence of queue length increase over a threshold value τ. For this reason we limit the range of actions that the CA α_i can perform. In particular, it can only re-route traffic flows that are injected in the network from the node i (in this way we also limit the number of times that a flow is re-routed). Furthermore, its goal is limited to the mitigation of congestion events happening on local interfaces.

The pseudo-code of *QLR* performed by CA α_i is reported in Alg. 1. Here we stress that this algorithm takes as input the current network status which is monitored through the EiP as a background process. More specifically, the network status is not collected by sending probes once a possible congestion is detected, thus keeping the reaction time low enough to handle bursty traffic. As first, the links leaving the node i are sorted in ascending order with respect to the available capacity (line 1). The rationale behind this choice is to give the priority to rerouting operation offloading congested links. In line 2 it is created a temporary data structure containing the current available capacity. It will be used along the CA execution to

take into account the effects of multiple rerouting operations. Then, output links are considered one at a time (line 3) and, if the current queue occupation overcomes the threshold value τ, then it tries to reroute some of the flows passing through it so that to reduce the link load and alleviate the congestion. In line 5, all the traffic flows injected in the network from the node i and having a working path that passes through the link that is currently under test are sorted. Then, a feasible alternative path is searched for each flow (lines 6-7): if in the set $\Pi_{i,e}$ exists a path having enough available capacity and that does not pass through any link whose buffer occupation is above the threshold, then it is selected as the new working path for the considered flow. In case there are multiple feasible alternative paths, the one that has the maximum bottleneck capacity is selected. After that (line 8) the available capacity is updated accordingly, and the need for another rerouting operation is checked (line 9): if the available capacity on the link under test is now greater than 0, then the *for loop* in line 6 is exited and the next link is inspected.

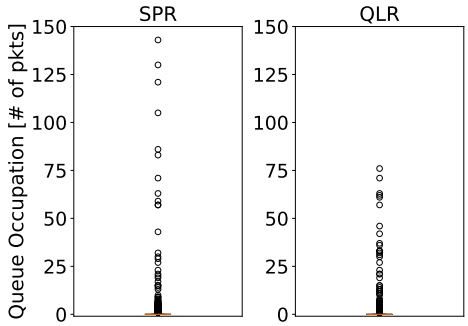
IV. Performance Evaluation

We have tested the performance of *QLR* in the network scenario shown in Fig. 1 emulated by Mininet, where programmable bmv2 switches [16] are used to implement the source routing and EiP functionalities. The network has been loaded by generating a set of UDP flows between all pairs of nodes using Iperf. Each flow has a volume that is randomly selected in the set $[3, 5, 8]$ Mbps with a probability of 0.7, 0.15 and 0.15 respectively, and a duration of 0.3 seconds. Once the flows expire, new ones are started. The simulation lasts after 20 seconds. Each node is controlled by a CA implemented as a custom python script, that is executed every 40 milliseconds. The threshold considered for the queue occupation is set to 30 packets.

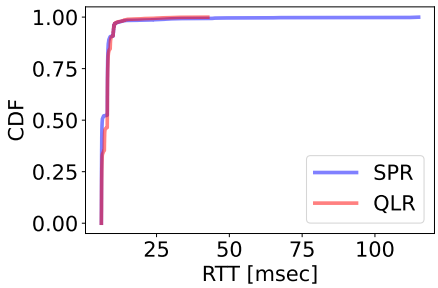
We compare the performance of *QLR* with the case where the routing is kept constant (Shortest Path Routing, SPR) throughout the simulation. The results of the performed test are reported if Fig. 2. In particular, Fig. 2(a) shows the boxplot of the queue occupation in case of SPR and *QLR* respectively: each reported point represents the queue occupation of a specific interface in a given time instant. From the figure it is evident that *QLR* is able to considerably reduce the maximum queue occupation, that passes from 143 to 77 packets, i.e., a reduction of approximately the 43%. This reduction in the maximum queue occupation leads to a decrease of the delay. This fact is confirmed by the result reported in Fig. 2(b), where the CDF of the RTT between all the pairs of nodes, is reported. In particular, it is interesting to notice that the maximum RTT passes from 115 to 42.6 milliseconds (see the maximum value reached by each curve with respect to the x axis), i.e., a reduction of the 63%.

V. Conclusions

In this paper we evaluated the feasibility of dynamically adjusting the network routing to limit the maximum queue occupation, thus reducing the maximum end to end delay.



(a) Boxplot of the Queue Occupation.



(b) CDF of the RTT.

Figure 2. The first figure shows the boxplot of the queue occupation in case of SPR and *QLR*, while the second figure shows the CDF of the RTT.

The proposed *QLR* solution defines a per-router Control Agent and is enabled by the availability of SRv6 for the re-routing operations and EiP for statistics collection and distribution among nodes. The preliminary performance evaluation has proven that *QLR* can work at millisecond time scale, being able to reduce the maximum queue occupation of the 43% and the maximum RTT of the 63%. Future effort must be done in the direction of the definition of a smarter control algorithm for selecting the re-routing and a deep evaluation, including the impact of re-routing operations on TCP flows, and more advanced experiments to compare the solution with other existing techniques.

References

[1] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (sdn): a survey," *Security and communication networks*, vol. 9, no. 18, pp. 5803–5833, 2016.

[2] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, and L. J. Wobker, "In-band network telemetry via programmable dataplanes," in *ACM SIGCOMM*, vol. 15, 2015.

[3] Q. Li, J. Zhang, T. Pan, T. Huang, and Y. Liu, "Data-driven routing optimization based on programmable data plane," in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2020, pp. 1–9.

[4] ITU-TR, "Transport network support of IMT-2020/5G," 2018.

[5] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time," *Queue*, vol. 14, no. 5, pp. 20–53, 2016.

[6] D. Brunello, I. Johansson, M. Ozger, and C. Cavdar, "Low latency low loss scalable throughput in 5g networks," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*. IEEE, 2021, pp. 1–7.

[7] S.Salsano, G.Sidoretti, C.Scarpitta, H.ElBakoury, D. Lopez, L.Bracciale, and P.Loreti, "Supporting future internet services with extensible in-band

processing (eip)," *1st ACM SIGCOMM Workshop on Future of Internet Routing & Addressing (FIRA)*, 2022.

[8] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment Routing Architecture," RFC 8402, Jul. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8402

[9] M. Ye, J. Zhang, Z. Guo, and H. J. Chao, "Date: Disturbance-aware traffic engineering with reinforcement learning in software-defined networks," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. IEEE, 2021, pp. 1–10.

[10] S. Sivabalan, C. Filsfils, J. Tantsura, W. Henderickx, and J. Hardwick, "Path Computation Element Communication Protocol (PCEP) Extensions for Segment Routing," RFC 8664, Dec. 2019. [Online]. Available: https://www.rfc-editor.org/info/rfc8664

[11] P. Psenak, S. Hegde, C. Filsfils, K. Talaulikar, and A. Gulko, "IGP Flexible Algorithm," Internet Engineering Task Force, Internet-Draft draft-ietf-lsr-flex-algo-22, Sep. 2022, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-lsr-flex-algo/22/

[12] D. Aureli, A. Cianfrani, M. Listanti, and M. Polverini, "Intelligent link load control in a segment routing network via deep reinforcement learning," in *2022 25th Conference on Innovation in Clouds, Internet and Networks (ICIN)*, 2022, pp. 32–39.

[13] M. Polverini, D. Aureli, A. Cianfrani, F. G. Lavacca, and M. Listanti, "Enhancing the srv6 network programming model through the definition of the maximize throughput behavior," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, 2022, pp. 1–5.

[14] S. Salsano, H. ElBakoury, and D. Lopez, "Extensible In-band Processing (EIP) Architecture and Framework," Internet Engineering Task Force, Internet-Draft draft-eip-arch-00, Jun. 2022, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-eip-arch/00/

[15] M. Polverini, A. Cianfrani, and M. Listanti, "A theoretical framework for network monitoring exploiting segment routing counters," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1924–1940, 2020.

[16] "P4 Tutorial, howpublished = https://github.com/p4lang/tutorials, note = Accessed: 2022-09-16."