A New Approach to QoS-aware and Robust Traffic Engineering in heavily loaded SDN-based ISP Networks

Slavica Tomovic[*], *Igor Radusinovic*[**]

Keywords: *SDN, traffic engineering, QoS.*

Abstract: In this paper, we propose a new SDN (Software Defined Networking) controller design that jointly addresses the problems of optimal traffic engineering (TE), quality of service (QoS) provisioning and failure recovery in ISP (Internet Service Provider) networks. The control logic of the controller is divided into the offline and online component. The online component handles dynamic arrivals of users' requests in accordance with Service Level Agreements (SLAs). The offline component is responsible for periodic optimization of traffic distribution in the network. The goal of such architecture is to increase acceptance ratio for QoS requests and minimize degradation of background (best-effort) traffic in a scalable manner. Our simulation study has shown that the proposed solution outperforms the competitive one, which is commonly used in MPLS-based ISP networks, even when TE optimization is repeated after relatively long time intervals. This alleviates concerns regarding the scalability of the SDN control plane in large-scale ISP networks.

1. INTRODUCTION

With the increasing popularity of multimedia services and evolution of the Internet of Things (IoT), QoS (Quality of Service) provisioning becomes one of the most challenging aspects of next-generation networks. In order to provide the necessary QoS, ISP providers usually adopt a strategy of link overprovisioning [1-2]. However, due to high costs associated with WAN (Wide Area Network) links, such strategy cannot be economically justified in the long run. Thus, there is an ever-rising expectation from traffic engineering

[*] S. Tomović, Faculty of Electrical Engineering, University of Montenegro, 81000 Podgorica, Montenegro (e-mail: slavicat@ucg.ac.me).

[**] I. Radusinović, Faculty of Electrical Engineering, University of Montenegro, 81000 Podgorica, Montenegro (e-mail: igorr@ucg.ac.me).

(TE) techniques to improve the network utilization. TE techniques help to achieve a balanced consumption of the network resources, which is essential for QoS fairness among users. In addition, this helps in preserving end-to-end connectivity in case of sudden failures.

Software Defined Networking (SDN) provides a promising approach to manage networks with a high efficiency. However, SDN TE algorithms designed for the data center (DC) networks, inter-DC networks and small-scale wide area networks (WANs) [1, 6] cannot be directly applied to ISP networks, where incoming traffic cannot be controlled. For example, DC and private WAN networks have a privilege to throttle bulky data transfers when necessary [1, 3]. In this way, sudden leaps of traffic matrix could be efficiently smoothed out. On the other side, ISP networks must fulfill Service Level Agreements (SLAs) promises to their customers and thus cannot shape traffic demands. In SLA agreements, ISP commits itself to provide a specified QoS level to user traffic [7]. If the signed QoS level is not met, the ISP is penalized for the loss of service. Recent works [8, 9, 10] take into account this limitation and propose TE solutions for ISP networks. However, in [9] authors did not consider traffic differentiation, while [8] and [10] concern only with the efficiency of bandwidth-delay constrained routing. In [11], TE and failure recovery are jointly analyzed, but bandwidth and delay requirements of traffic flows were neglected.

This paper presents a new design of SDN controller with QoS on-demand, TE, and failure recovery capabilities. The control logic is divided into the online and offline component. In this way, we trade some routing optimality for the sake of the control plane scalability. The online component quickly handles arrivals of QoS requests. Best-effort flows are routed in the data plane, without any intervention from the controller's side. The offline component aims to periodically optimize load balancing in the network, such that maximum link utilization (MLU) is minimized and SLA policies are met. The controller is able to fulfill three types of QoS requirements: bandwidth, delay, and robustness to link failures. Therefore, we increase the portfolio of SLAs supported in traditional IP networks [11]. Simulation study on Sprint ISP topology has shown that the proposed approach strongly outperforms MPLS-based TE solution even when load balancing is optimized over long intervals.

The rest of the paper is organized as follows. Section II presents the system model as well as assumptions made in our analysis. Section III explains the simulation framework and the obtained results. Finally, we conclude the paper in Section IV.

2. SYSTEM MODEL

We consider the ISP network architecture from [10], as illustrated in Fig. 1. The data plane consists of Core OpenFlow [12] switches and Provider Edge (PE) OpenFlow switches. We assume that ISP offers four basic types of SLAs to customers:

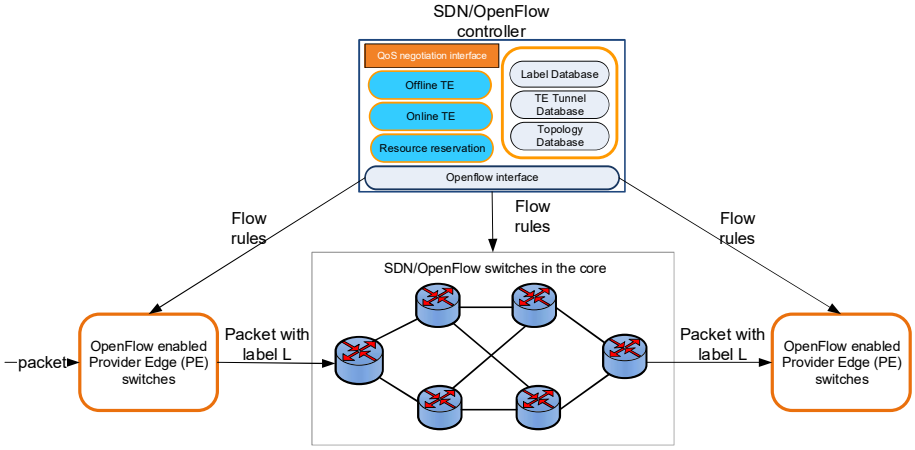

Fig. 1. The software-defined ISP network model.

1. **Best-effort –** offers connectivity service without guaranteeing performance for the connection.
2. **QoS1 –** specifies guarantees on the minimum bandwidth and maximum delay with loose bounds.
3. **QoS2 –** guarantees very low delay bound and the required bandwidth for time-critical data transfers.
4. **QoS3 –** guarantees bandwidth, delay, and robustness to a pre-defined number of link failures. It is intended for a traffic that requires a reliable transfer, with a low packet loss probability.

Decisions on SLAs are made by the control plane, which has two components responsible for traffic control:

- **Online routing –** computes the most suitable route for QoS1, QoS2 and QoS3 requests.
- **Offline TE –** runs TE algorithm periodically in order to determine the optimal traffic distribution scheme.

Up to K tunnels are installed a priori between each pair of PE devices, as usual for today's' ISP networks [1]. When a link or a switch failure occurs in the network core, it affects all tunnels that traverse it. In such situations, ingress PE switches just rescale traffic traversing a failed link or a core switch over the remaining tunnels, in proportion to load balancing weights that are proactively installed by the controller. Such rescaling is feasible with OpenFlow Group tables [12].

PE OpenFlow switches classify traffic in one of total four classes that correspond to the four SLA types. Each traffic class is marked with a specific label. The controller does not react on individual arrivals of best-effort traffic flows. All incoming best-effort traffic with

the common destination PE switch, splits across K tunnels, according to pre-configured load balancing weights that are periodically optimized by the controller's Offline TE component. This is acceptable because QoS is not guaranteed for best-effort class. On the other side, QoS1, QoS2, and QoS3 traffic initially follow constrained shortest paths computed by the Online routing component. However, the Offline TE component periodically redistributes this traffic over multiple tunnels (up to K) in order to optimize the network utilization, and therefore to create conditions for more QoS connection requests to be accepted in the network.

A. Online Routing module of SDN controller

The Online routing module performs admission control for QoS requests by using a QoS-aware routing algorithm, which seeks for the tunnel(s) that can fulfill guarantees specified in the corresponding SLA. If no such tunnel(s) exists in the network, a request is rejected. At this stage, the goal is not to optimize the routing decision, but rather to find a quick solution to route incoming traffic. This is because, for example, a combination of TE and bandwidth-delay constrained routing would result in problem unsolvable in real-time [8]. Thus, we designed the Online routing component based on the solutions used in traditional (MPLS-based) ISP networks. Namely, routes for QoS1 and QoS2 requests are installed with bandwidth-delay CSPF (Constrained Shortest Path First) algorithm. This is achieved in two simple steps. Firstly, all the tunnels with insufficient bandwidth are pruned from the network graph. In the second step, a tunnel with the least delay is found. If the tunnel delay is higher than required, the request is rejected. Otherwise, new routing rules are installed in the network. In case of QoS3 requests, n disjoint bandwidth-delay constrained paths are computed. In this way, the algorithm guarantees that no service disruption will happen for up to n link failures.

B. Offline TE module of SDN controller

The Offline TE module periodically runs the optimization algorithm, which tends to minimize MLU subject to QoS constraints. The proposed algorithm models ISP network as a directed graph G=(V, L), where V is the set of nodes and L is set of edges. Each link $l \in L$ is described by capacity c_l and propagation delay d_l. We assume that propagation delay predominantly determines a total path delay in ISP networks [14].

A traffic matrix TM is a $|N|x|N|$ matrix, where N is the set of PE OpenFlow switches. Matrix element t_{ij} refers to an aggregated traffic demand between origin PE switch i and destination PE switch j. For the purpose of the network optimization, matrix TM is decomposed into four matrices: TM_{be}, TM_{qos1}, TM_{qos2} and TM_{qos3}, one for each traffic class. A class-level traffic demand $t_{ij}^{(C)}$, $C \in$ {be, qos1, qos2, qos3} is distributed across the set of tunnels P_{ij}, which is computed a priori. For each traffic demand $t_{ij}^{(C)}$, the offline algorithm computes optimal splitting ratios $x_{t_{ij}^{(C)},p}$ for tunnels $p \in P_{ij}$, i.e. portions of the traffic demand that would be routed over each tunnel from the set P_{ij}. In order to ensure

that QoS3 traffic demands will not experience congestion as long as a total number of link failures is less than, additional resources are reserved. The optimization algorithm is formulated as follows:

$$\text{minimize } z$$

Subject to:

$$\sum_{p\in P_{ij}} x_{t_{ij}^{(C)},p} == 1, \quad \forall C \neq be, \forall (i,j) \in N \tag{1}$$

$$\sum_{p\in P_{ij}} x_{t_{ij}^{(qos3)},p} \geq 1, \ \forall (i,j) \in N \tag{2}$$

$$\sum_{\forall C} \sum_{(i,j)\in N} \sum_{p\in P_{ij}} R_{p,l} \cdot x_{t_{ij}^{(C)},p} \cdot t_{ij}^{(C)} \leq z \cdot c_l, \qquad \forall l \in L \tag{3}$$

$$\sum_{\forall C\neq be} \sum_{(i,j)\in N} \sum_{p\in P_{ij}} R_{p,l} \cdot x_{t_{ij}^{(C)},p} \cdot t_{ij}^{(C)} \leq c_l, \qquad \forall l \in L \tag{4}$$

$$y_{t_{ij}^{(C)},p} \cdot (\sum_{l\in p} d_l - D_{\max}^{(C)}) \leq 0, \ \forall (i,j) \in N, \forall C \neq be, \forall p \in P_{ij} \tag{5}$$

$$y_{t_{ij}^{(C)},p} \geq x_{t_{ij}^{(C)},p}, \forall (i,j) \in N, \forall C \neq be, \forall p \in P_{i,j} \tag{6}$$

$$\sum_{p\in P_{ij}^{\mu}} x_{t_{ij}^{(qos3)},p} \geq 1, \ \forall (i,j) \in N, \mu \in U_n \tag{7}$$

$$x_{t_{ij}^{(C)},p} \geq 0, \ \forall (i,j) \in N, \forall C \ , \ \forall p \in P_{ij} \tag{8}$$

$$y_{t_{ij}^{(C)},p} \in \{0,1\}, \ \forall (i,j) \in N, \forall C \neq be, \ \forall p \in P_{ij} \tag{9}$$

$$z \geq 0 \tag{10}$$

The optimization problem from above minimizes MLU z, subject to the set of constraints (1) - (10). The decision variable $x_{t_{ij}^{(C)},p}$ defines a fraction of traffic demand $t_{ij}^{(C)}$ to be routed over a path *p*. Constraints (1) and (2) ensure that all traffic demands are routed. The link

capacity constraints are given with (3) and (4). $R_{p,l}$ is a binary value which indicates whether a path p includes a link l. If yes, $R_{p,l}$ is 1, otherwise, it is 0. Since QoS classes require bandwidth guarantees, a total amount of reserved bandwidth must not exceed a link capacity. On the other side, best-effort traffic flows enter the network without admission control. Thus, it is possible for the network to be overloaded, i.e. z could be greater than 1. In (5), we have introduced decision variables $y_{t_{ij}^{(C)},p}$, which serve for providing delay guarantees for priority traffic flows. This is necessary because total path delay for QoS traffic classes should not exceed $D_{\max}^{(C)}$, which stands for the maximum tolerable delay for a class. If path $p \in P_{ij}$ cannot meet delay requirement for a traffic class, (5) ensures that no traffic of that class is routed over *p*.

A computational challenge is imposed by (7), which ensures that QoS3 traffic will maintain the required throughput when up to *n* link fails. A case of link faults can be represented by a vector $\mu = \left[\mu_l \middle| l \in L \right]$, where μ_l is 1 when link l has failed or 0 otherwise. Therefore, a traffic demand is robust to *n* link failures if its bandwidth requirement can be met under the set of failure cases $U_n = \left\{ \mu \middle| \sum_l \mu_l \le n \right\}$. Given a fault case μ, for each traffic demand, we can determine the set of residual tunnels P_{ij}^{μ} that do not traverse any failed link. The constraints from (7) require that these tunnels are able to handle $t_{ij}^{(qos3)}$. Clearly, a larger number of residual tunnels entails greater network throughput. In order to lose as few as possible tunnels when a link failure occurs, for each ingress-egress (IE) PE switch pair, we bound the number of tunnels *q* that can traverse the same link, as proposed in [11]. Since tunnels between an IE pair are *q*-disjoint, for any link fault case $\mu \in U_n$, the number of residual tunnels $\left|P_{ij}^{\mu}\right|$ is no less than $\left|P_{ij}\right| - n \cdot q$. Thus, we can replace $|N| \cdot |N-1| \cdot \sum_{k=1}^{n} \binom{|L|}{k}$ constraints from (7) with:

$$\sum_{k=1}^{|P_{ij}|-n\cdot q} x_{t_{ij}^{(qos3)},p}^{\min k} \ge 1, \forall (i,j) \in N \tag{11}$$

Where $x_{t_{ij}^{(qos3)},p}^{\min k}$ denotes the *k*-th smallest element in $X = \left\{ x_{t_{ij}^{(qos3)},p} \middle| p \in P_{ij} \right\}$. If *n* is small, and the number of residual tunnels is large compared to the total number of tunnels, then it is more suitable to use:

$$\sum_{p \in P_{ij}} x_{t_{ij}^{(qos3)},p} - \sum_{k=1}^{n\cdot q} x_{t_{ij}^{(qos3)},p}^{\max k} \ge 1, \forall (i,j) \in N \tag{12}$$

The method for encoding M largest (or smallest) variables as linear constraints has been proposed in [11]. We applied this method to express (12) in the optimization problem.

4. SIMULATION RESULTS

To evaluate performance of the proposed TE model, we wrote simulator in Python that uses CPLEX [15] to solve optimization problems. In simulations, we used POP-level Sprint ISP topology from Fig. 2. Link capacities were set to 2.4Gb/s. according to the information available in [16], while delay on each link was derived from a great circle distance between the connected POPs and speed of light in the fiber [9].

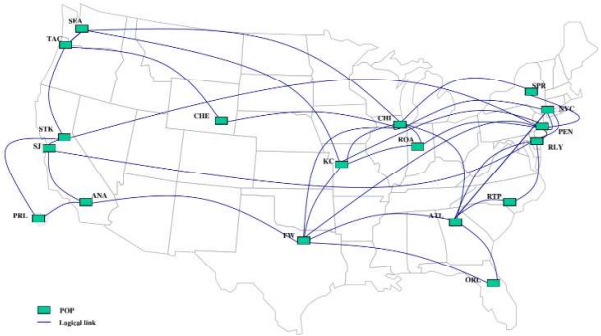

Fig. 2. Sprint POP-level network topology.

Since for most ISP networks access to real traffic traces is not available, we used a well-known gravity model [17] to generate average value of traffic matrix *TM*. From the gravity model matrix, we derived "admissible" traffic matrix, which certainly can be accommodated in the network if all flows are best-effort. This is done by firstly partitioning the gravity matrix into 100 identical elementary matrixes. Then, iteratively, we used bandwidth-constrained SPF algorithm to route each elementary matrix. This procedure stops when some link in the network gets congested. Sum of all elementary matrices that are successfully routed by CSPF makes "admissible" traffic matrix.

Finally, based on "admissible" traffic matrix, we simulated dynamic evolution of the network traffic. We used exponential distribution with the mean t =120s to model flow durations, and Poisson distribution to model flow arrivals between ingress-egress pairs. Mean rate of flow arrivals is computed such that resulting traffic model in average converges to "admissible" traffic matrix. To achieve that, we identify the average number of active traffic flows between ingress-egress pair as an average number of elements in $M/\mathrm{M}/\infty$ queue [18]. The average number of active traffic flows between ingress node i and egress node j is obtained as:

$$N_{ij} = \frac{t_{ij}}{B} \tag{11}$$

where B denotes bandwidth demand of single traffic flow in average. Since it is well known that average occupancy of $M/\mathrm{M}/\infty$ queue is equal to traffic intensity [18], we derived average rate of Poisson arrivals between an ingress-egress pair as:

$$\lambda_{ij} = \frac{N_{ij}}{t} \tag{12}$$

For simplicity, bandwidth requirement of each Poisson arrival was chosen randomly from the set [10Mb/s, 15Mb/s, 20Mb/s]. QoS2 delay bound was set to 62ms, which corresponds to delay on the shortest path between the mutually most distant ingress-egress pair in the network. Maximum tolerable delay for QoS1 and QoS3 traffic classes was set to 100ms and 150ms respectively. For simplicity, we assumed that QoS3 SLA guarantees robustness to single link failure (n=1). Each traffic class accounted for 25% of overall traffic.

The proposed TE approach has been compared with the approach that is commonly applied in traditional MPLS networks. In the rest of the Section, by MPLS-based TE we assume approach without optimizations, which implies uniform best-effort load balancing over set of tunnels installed a priori, and CSPF routing of QoS-demanding traffic flows.

Fig. 3 shows the results in terms of rejected QoS requests for the proposed TE approach (denoted as SDN-TE) and MPLS-based approach. The results correspond to scenario where only disjoint tunnels between edge switches are used for traffic routing (i.e. q=1). From Fig. 3 it is obvious that the proposed TE approach is able to use network resources more efficiently than MPLS-based approach, which we refer to as the competing solution. These results are not surprising, considering that in SDN scenario load balancing parameters for each traffic class are optimized periodically. MPLS-based TE approach uses CSPF algorithm to selects tunnels for QoS demands, which results in forcing tunnels with the least delay. Although these tunnels are optimal from the perspective of delay sensitive applications, it happens that links critical for many ingress-egress pairs become quickly congested. These are links with the very low delay, which are usually part of the least delay paths. The consequence is a high number of rejected QoS requests. Inefficiency of the competing approach also stems from the fact that QoS3 requests are served by allocating double value of their bandwidth demand.

Having in mind the unpredictable nature of traffic demands in ISP networks, we explored the impact of the frequency of the offline optimizations on the network performance. Thus, we run simulations for different recurrence periods T of offline phases: 2, 5, 10 and 20 minutes. The best results were obtained in scenario where optimization algorithm was run every 2 minutes. However, it is probably unrealistic to expect such frequent reconfigurations and gathering of network statistics in ISP networks, where scalability of the control plane is a primary concern. Increase in the interval between consecutive offline phases resulted in an increase of rejected QoS requests. This performance drop could be attributed to the fact that traffic matrix is prone to significant variations during the time interval between consecutive offline phases. Thus, load-balancing parameters that are considered optimal at the beginning of the interval, deviate significantly from the real optimum after a while. However, we can see that SDN-TE is scalable, and achieves performance improvement even for large T.

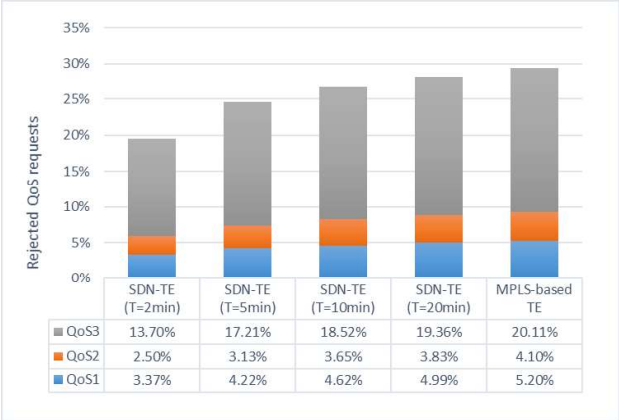

Fig. 3. Rejected QoS requests (q=1).

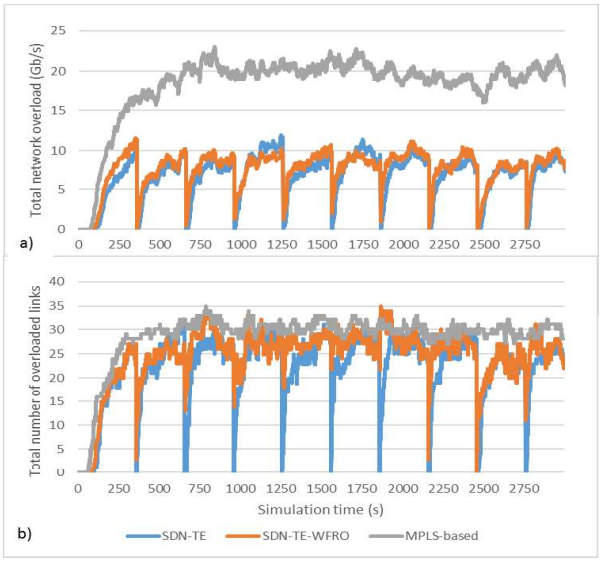

Fig. 4. Total network overload in Gb/s (a) and number of overloaded links (b) during the simulation.

Fig. 4a shows total network overload in each second of simulation time for following three cases: i) the proposed approach – (SDN-TE); ii) the proposed approach without periodic optimization of QoS3 traffic distribution (SDN-Without Failure Recovery

Optimization); iii) MPLS-bases TE. The network overload is computed by summing difference between the input traffic demand and capacity of each overloaded link. The number of overloaded links during the simulation running time is shown in Fig. 4b. These results might serve to analyse performance of best-effort traffic. It should be noted that in all simulations of SDN-TE approach we performed offline optimization in 60th second. After that, offline phases were repeated according to the periods T specified in the Figures. At the beginning of SDN-TE simulations, when all elements of traffic matrix are zero, the uniform load balancing over set of available tunnels is used. Thus, by performing early optimization, we wanted to reduce the impact of the traffic oblivious load balancing in period before the first regular offline phase. From Fig. 4a it could be observed that in SDN-TE case, total network overload and the number of overloaded links experience drops to zero in the moments of TE optimization (i.e. in 61+T, 61+2T, etc.). On the other side, if we just omit optimization of QoS3 traffic, offline TE cannot bring total overload down to zero in most of the cases. Moreover, number of overloaded links remains very high most of the time. This suggests decrease in throughput loss of best-effort flows, achieved by implementing failure recovery mechanism from [11].

We also investigated the impact of tunnel diversity on TE performance. The number of tunnels is controlled by the parameter q, but bounded to 20 due to hardware limitations of OpenFlow switches [19]. Fig. 5 shows the percentage of rejected QoS requests for q=1, q=2 and q=3. In general, splitting traffic across more tunnels leads to lower MLU when SDN-TE is used, so the number of rejected requests decreases correspondingly. However, the complexity of offline optimization increases with q. Namely, when q is increased from 1 to 3, the number of tunnels increases from 868 to 2357. Fig. 6 illustrates how the increase of q affects the total network overload and the number of overloaded links. The total network overloaded is the lowest for q=1 and the highest for q=3, but there is a huge difference in the number of accommodated QoS request for these two scenarios. It is interesting to note that for q=3 the number of overloaded links was increased by offline TE algorithm. This is because the optimization goal is the minimization of MLU, which often results in splitting traffic of highly overloaded links over multiple heavily loaded (but not overloaded) links.

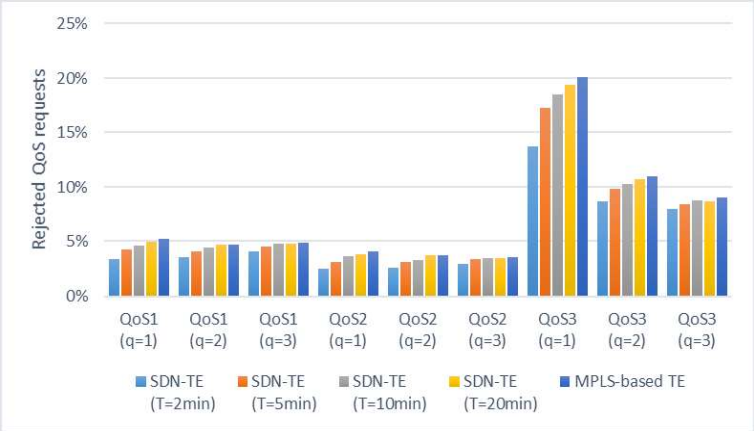

Fig. 5. Rejected QoS requests in function of parameter q.

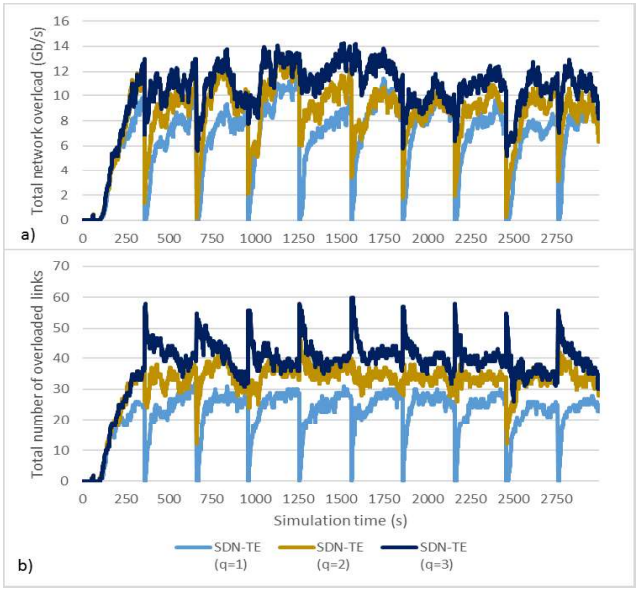

Fig. 6. Total network overload in Gb/s (a) and number of overloaded links (b) for different values of q.

5. CONCLUSION

This paper is extended version of [20], where we have proposed a new SDN controller design, that is capable of jointly handling TE, QoS provisioning and failure recovery problems ISP networks. The controller's logic for traffic control is divided into the Online routing component and the Offline TE component. The Online routing component is designed to quickly find a solution to route traffic demands with strict QoS requirements. Best-effort traffic demands are routed over multiple tunnels, without interaction with the controller, based on load-balancing weights that are installed a priori. Offline TE component periodically reconfigures the network in order to minimize MLU subject to the QoS constraints. The simulation results are promising considering that an improvement over the traditional MPLS-based approach is evident even when the offline TE is performed infrequently. In that sense, the proposed controller design alleviates concerns for scalability of the centralized TE and has potential to reduce the operating costs for ISPs. In order to prove the claimed benefits of the proposed solution in the more realistic testing environment, a further analysis will be performed in Mininet emulator [21]. Also, the QoS framework would be adapted to the service orchestration framework [22, 23] for Internet of Things environments.

REFERENCES

[1] C.-Y. Hong et al., “Achieving high utilization with software-driven WAN,” presented at ACM SIGCOMM 2013, pp. 15–26, 2013.

[2] X. Jin, Y. Li, D. Wei, S. Li, J. Gao, L. Xu, G. Li, W. Xu, and J. Rexford, “Optimizing bulk transfers with software-defined optical WAN,” in Proceedings of the ACM SIGCOMM '16, New York, pp. 87-100, 2016.

[3] S. Jain et al., “B4: Experience with a globally-deployed software-defined WAN”, in Proc. of ACM SIGCOMM, vol. 43, no. 4, pp. 3–14, 2013.

[4] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, “Hedera: dynamic flow scheduling for data center networks”, in Proc. of NSDI’10, San Jose, 2010, vol. 10, pp. 19–29.

[5] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, “DevoFlow: scaling flow management for high-performance networks”, in Proc. ACM SIGCOMM, vol. 41, no. 4, pp. 254–265, 2011.

[6] J. M. Wang, Y. Wang, X. Dai and B. Bensaou, "SDN-based multi-class QoS-guaranteed inter-data center traffic management," in Proc. IEEE 3rd International Conference on Cloud Networking, pp. 401-406, 2014.

[7] P. Pongpaibool and H. S. Kim, "Guaranteed service level agreements across multiple ISP networks," in Proc. 4th International Workshop on Design of Reliable Communication Networks (DRCN 2003), pp. 325-332, 2003.

[8] S. Tomovic and I. Radusinovic, "Fast and efficient bandwidth-delay constrained routing algorithm for SDN networks," in Proc. IEEE NetSoft Conference and Workshops, Seoul, pp. 303-311, 2016.

[9] J. L. Martins and N Campos, "Short-sighted routing, or when less is more," IEEE Comm. Magazine, vol. 54, no. 10, pp. 82-88, Oct. 2016.

[10] S. Tomovic, I. Radusinovic, “Traffic engineering approach to virtual-link provisioning in software-defined ISP networks,” in Proceedings of TEFOR ’17 conference, Belgrade, Nov. 2017.

[11] H. H. Liu et al., “Traffic Engineering with Forward Fault Correction,” Proc. of ACM SIGCOMM 2014, pp. 527–38, 2014.

[12] D. C. Verma, "Service level agreements on IP networks," in Proceedings of the IEEE, vol. 92, no. 9, pp. 1382-1388, Sept. 2004.

[13] Open Networking Foundation - OpenFlow v1.4 specification. [Online]. Available: https://www.opennetworking.org/, 2017.

[14] C. Fraleigh, F. Tobagi, and C. Diot. "Provisioning IP backbone networks to support latency-sensitive traffic," in Proc. INFOCOM 2003, vol. 1, pp. 375-385, 2003.

[15] CPLEX Optimization Studio. [Online]. Available: http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud, 2017.

[16] R. Fukumoto, S. Arakawa, T. Taking, and M. Murata, “Analyzing and modeling router-level Internet topology,” in Proceedings of ICOIN’07, pp. 171–182, Jan. 2007.

[17] A. Gunnar, M. Johansson, and T. Telkamp, “Traffic matrix estimation on a large IP backbone: a comparison on real data”, In Proc. of the 4th SIGCOMM conference, pp.149-160, Italy, 2004.

[18] P. Harrison, and N. M. Pate, “Performance Modelling of Communication Networks and Computer Architectures”. Addison–Wesley, p. 173, 1992.

[19] Broadcom Trident II series. [Online]. Available: http://www.broadcom.com/docs/features/StrataXGS_Trident_II_presentation.pdf, 2012.

[20] S. Tomovic and I. Radusinovic, "A new traffic engineering approach for QoS provisioning and failure recovery in SDN-based ISP networks," 2018 23rd International Scientific-Professional Conference on Information Technology (IT), Zabljak, 2018, pp. 1-4.
[21] Mininet: http://mininet.org/
[22] S. Tomovic et al., "An architecture for QoS-aware service deployment in software-defined IoT networks," 2017 20th International Symposium on Wireless Personal Multimedia Communications (WPMC), Bali, 2017, pp. 561-567.
[23] B. Skrbic, D. Radovanovic, S. Tomovic, L. Lazovic, Z. Zecevic and I. Radusinovic, "A decentralized platform for heterogeneous IoT networks management," 2018 23rd International Scientific-Professional Conference on Information Technology (IT), Zabljak, 2018, pp. 1-4.