A QoS-based Flow Assignment for Traffic Engineering in Software-Defined Networks

Lakshmi Priya Thiruvasakan, Quoc-Tuan Vien, Jonathan Loo, and Glenford Mapp

Abstract In order to meet a tremendous amount of data storage requirement in next-generation wireless networks, an increasing number of cloud data centers has been deployed around the world. The underlying core networks are expected to provide the ability to store data in a dynamic and scalable computing environment. The traditional Internet Protocol (IP) has shown to be restricted due to its static architecture, which accordingly motivates the development of Software-Defined Networks (SDNs). In the SDNs, Traffic Engineering (TE) is simpler and programmable with a controller without the requirement of reconfiguration for all network devices. However, the existing TE algorithm of the SDNs rejects a number of requested flows caused by their undetermined routing paths where only flow bandwidth is considered in path determination. This paper proposes a Quality-of-Service (QoS) based Flow Assignment algorithm which enables the computation of end-to-end path for traffic flows guaranteeing the QoS requirements including bandwidth, end-to-end delay and packet loss probability. Through the Open Source Hybrid IP/SDNs platform, the proposed algorithm is validated and shown to significantly reduce flow rejection rate of up to 50% compared to the conventional approach, and therefore can be used to implement an effective DiffServ mechanism for flow allocation in the SDNs.

1 Introduction

Software-Defined Network (SDN) has recently emerged to overcome the limitations of traditional Internet Protocol (IP) networks and Multi-Protocol Label-Switching (MPLS) systems, by enabling dynamic and scalable computing [1]. In SDN, control plane is decoupled from data plane and placed in a centralized SDN controller [2, 3]. The network intelligence and state are logically centralized in a single place, allowing network administrators to easily apply network wide policy at the controller. Cisco defined a new way of Traffic Engineering (TE) called Segment Routing (SR) [4] to be used with SDN. SR implemented in hybrid MPLS/SDN network can benefit from centralization logic of SDN, where routing paths can be computed at SDN controller using flow assignment algorithm.

As an SDN emulator, Open Source Hybrid IP/SDN (OSHI) [5], a part of DREAMER project, was designed and developed by Davoli *et al.* [6]. OSHI plat-

L. P. Thiruvasakan, Q.-T. Vien, G. Mapp
Middlesex University, London, United Kingdom
e-mail: lakshmipriya.thiru@gmail.com,{q.vien;g.mapp}@mdx.ac.uk

J. Loo
University of West London, London, United Kingdom. e-mail: jonathan.loo@uwl.ac.uk

form is a whole package which provides tools to build network topology, develop a flow assignment algorithm namely TE/SR algorithm, and test the algorithm using emulated network [6]. There is an inherent TE/SR or flow assignment algorithm within OSHI [7], which computes routing paths based only on bandwidth requirement of the flow. In the path computation of traffic flows in real-time systems, the end-to-end delay and packet loss probability are also important metrics; however, they have received less interests in the literature. To this extent, this paper aims at addressing these metrics in the flow assignment. The main contributions of this paper can be summarized as follows:

1. *Development of queuing models for network links and nodes*: To enable OSHI to use queuing models M/M/1 and M/M/1/K to simulate QoS metrics like delay and packet loss probability.
2. *A novel QoS based Flow Assignment (QFA) algorithm*: The novelty of the proposed QFA algorithm lies in enhancing the existing OSHI algorithm [7] by considering delay and packet loss probability in addition to the already supported bandwidth while computing routing path. The proposed algorithm is shown to reduce significantly the flow rejection rate compared to the existing algorithm.

2 Related Works

With rapid growth of real time Internet applications, packets are expected to reach destination on time, and also with a less packet loss. This can be achieved when routing algorithm considers critical flow parameters like bandwidth, delay and packet loss probability. Very few research efforts done in the past [8] [9] proposed an SDN routing algorithm, which considers bandwidth and end-to-end delay metrics.

One approach to meet delay requirement of a flow is to map traffic flows to a statically configured output queue of a switch based on their priorities. This allows managing both bandwidth and delay properties of the flow at each SDN switch. Kim et al., in 2010 [8] proposed an algorithm for automatic QoS control over OpenFlow protocol (OF) [10]. The algorithm guarantees flow's bandwidth by generating an optimal routing path, where the links in the path have enough capacity to support new flow. Delay requirement is then satisfied by configuring multiple priority based output queues to which the flows are mapped. Flow paths generated by SDN algorithm are placed in flow tables in the OF switch using OF protocol [10]. Any packet arriving at the switch is matched against the flow table entries to determine its routing path. A drawback in setting up bandwidth-delay guaranteed tunnel in a large-scale network is that it requires a highly computational complex algorithm [9]. To reduce complexity of TE algorithms, Tomovic and Radusinovic proposed a simple QoS provisioning algorithm [9], where control plane classified incoming QoS requests into finite number of categories based on delay sensitivity level. A QoS request gets rejected when algorithm fails to calculate shortest routing path. Their experiment proved that simple algorithm can also perform better and lead to smaller rejection of QoS requests than complex solutions. This paper aims to develop such a simple QoS based routing algorithm.

For timely delivery of packets in real-time systems, it is essential for packets to follow a routing path, where total delay encountered by packet is less than or

equal to delay requirement of flow. Implementing delay based flow optimization in traditional networks involves expensive custom-built hardware and software [11]. On other hand, with SDN the controller is aware of global view of network and statistics, and thus delay based routing algorithm can be implemented in a cheaper and efficient way. Recently, Kumar et al., in 2017 [11] designed an algorithm which considers only high criticality flows in safety-critical systems, whose properties like delay and bandwidth are known to algorithm designers well ahead of time, before actual data transmission starts. The algorithm calculates end-to-end delay value by summing up various delay elements like nodal processing delay, link transmission delay, link propagation delay and queuing delay.

Most useful research is based on OSHI [6] (see Section 3). Our paper is different from aforementioned works in two key ways in terms of optimization goals and the considered use case. Furthermore, most of the papers only considered bandwidth and/or end-to-end delay, while our research designs a QFA algorithm to identify routing path that guarantees specified bandwidth, delay and packet loss probability metrics.

Although there exist a variety of simulation platform for SDN such as MATLAB and NS3 [12], OSHI is selected as a base for validating proposed algorithm. This is due to the fact that it can provide a simple and efficient virtual environment to develop and test flow assignment algorithm using simulated SDN network.

3 OSHI System Model

OSHI encompasses various software components like Network Topology Parser, Random Demand Generator, Flow Assignment or TE algorithm, SR algorithm etc. [6]. Network can be graphically designed for small-scale topology and deployed using Mininet [13] extensions. Alternatively, large-scale topology can also be downloaded from Topology Zoo [14]. These graphs are converted to JSON files and fed to a random demand generator.

Random demand generator [15] uses network topology as input, and generates a set of flow catalogues. TE algorithm [7] accepts network topology and flow catalogue as inputs and computes optimal routing path for each flow in flow catalogue file. This section describes various phases of existing TE algorithm [7]. TE algorithm is a classical flow assignment problem which computes optimal path between source and destination, given flow's expected bandwidth (Megabits per Second or Mbps). Path computed by TE algorithm is further shortened by SR algorithm. SR pusher then deploys SR paths into the network SDN nodes via Ryu controller [6].

As mentioned in OSHI Quarterly report [16], TE algorithm implements below 3 phases in sequence.

1. Initialization phase: Network topology and flow catalogue JSON files are read and converted to in-memory data structures. A network graph is created using topology files.
2. Constrained Shortest Path First (CSPF) phase: Shortest routing path is calculated for each flow using Dijkstra's algorithm. Weight of any link 'i' is calculated as,

$$W_i = \frac{BIGK}{C_i - \lambda_i} \tag{1}$$

where W_i is link weight, BIGK is biggest capacity present among all links, C_i is capacity of link and λ_i is load of link. Flows for which shortest path cannot be computed gets rejected in this phase. Only accepted flows continue to next phase. At end of this phase, average crossing time of network [16] is calculated as shown below.

$$T = \frac{1}{\gamma} \sum_{i=links} \frac{\lambda_i}{(C_i - \lambda_i)} \tag{2}$$

where γ is total load introduced in network.

3. Heuristic reassignment phase: This phase aims at reducing T (average crossing time) value, calculated in end of previous phase. During this phase, edge weights W_i are calculated using (3), which is derived by differentiating T with respect to λ_i. The derived value is increase of delay on link 'i' brought on by a very small amount of traffic on the link.

$$W_i = \frac{1}{\gamma} \frac{C_i}{(C_i - \lambda_i)^2} \tag{3}$$

Thus, shortest path calculated in this phase reduces overall delay in network compared to previous phase.

4 Problem Definition and Proposed Solutions

Below are the enhancements added to OSHI platform [6] and its TE algorithm [7].

1. Improvise OSHI platform to support M/M/1 and M/M/1/K queuing models.
2. Reduce flow rejection ratio by sorting flows. Existing algorithm [7] has a very high flow rejection ratio i.e., 1470 flows out of 2198 requested flows gets rejected.
3. Modify TE algorithm design such as to calculate routing paths that guarantee QoS metrics bandwidth, packet loss and delay.

The new QFA algorithm is an offline and static algorithm which can be scheduled to run periodically, say during night or off-peak times. The algorithm can be used to handle predefined traffic flows in an organization or data centres.

4.1 OSHI platform changes to support queuing models

4.1.1 Guaranteed Packet Loss Probability

Modelling each router as M/M/1/K system [17], packet loss probability or blocking probability is given by formula [17],

$$P_K = (1 - \rho) \frac{\rho^K}{1 - \rho^{(K+1)}} \tag{4}$$

where $\rho = \lambda/\mu$ and is system utilization , λ is arrival rate (Packets per Second), μ is service rate (Packets per Second), K is maximum system capacity which is total number of packets present in ingress buffers plus one packet which is being serviced.

For the experiments, packet loss probability is considered as 0.0001% [18], which signifies that maximum packet loss allowed in any node is 1 in 10^6 packets. With K and μ known for a node (which will be explained in 4.1.3), λ is calculated such that P_K <0.0001%. This λ value denotes maximum packet arrival rate allowed in a node to keep packet loss probability less than 0.0001%. When calculating the shortest path between end points, the algorithm makes sure that the packet arrival rate at any node is kept within this calculated λ value.

4.1.2 Guaranteed End-to-End Delay

End-to-end delay ($T_{EndtoEnd}$) of a packet is summation of below components [19]

$$T_{EndtoEnd} = \sum_{links} (T_{trans} + T_{queue} + T_{propagation}) + \sum_{nodes} T_{nodalProc} \tag{5}$$

where T_{trans} is link transmission delay, T_{queue} is queuing delay, $T_{propagation}$ is link propagation delay and $T_{nodalProc}$ is nodal processing delay.

When each link in routing path is modelled as M/M/1 system [19] total time spent by packet waiting in outgoing queue and time for transmission over link 'i' is given by [19],

$$T_{trans} + T_{queue} = \frac{1}{\mu C_i - \lambda_i} \tag{6}$$

where λ_i is load of i^{th} link (Packets Per Second), C_i is capacity of i^{th} link (Bits Per Second) and $\frac{1}{\mu}$ is average length of data packet (Bits per Packet).

Propagation speed in physical media ('v') is usually a fraction of speed of light i.e., 0.59c to 0.77c where c is speed of light in vacuum [20]. Propagation delay of link 'i' can be expressed as [19],

$$T_{propagation} = \frac{l_i}{v} + \frac{1}{\mu C_i} \tag{7}$$

where l_i is length of i^{th} link (Metres) and is assumed 100 metres [11], v is propagation speed of link, C_i is capacity of i^{th} link (Bits Per Second) and $\frac{1}{\mu}$ is average length of data packet (Bits per Packet).

When modelling network node as M/M/1/K system, total time spent by packet in the system is given by [21],

$$T_{nodalProc} = \frac{(\frac{\rho}{1-\rho}) - (\frac{(K+1)\rho^{(K+1)}}{1-\rho^{(K+1)}})}{(1-P_K)\lambda} \tag{8}$$

where ρ is system utilization, λ is packet arrival rate (Packets per Second), K is maximum system capacity and P_K is probability of having K packets in the system as mentioned in (4).

4.1.3 OSHI Input Parameters to QFA Algorithm

QoS metric values for flows are taken from Colt CPE Solutions Service Level Agreement [18]. The input data to algorithm are json files like nodes.json and links.json which are created based on network topology, and flow_catalogue.json which is generated by random demand generator. Section 5 mentions the steps to

generate these JSON files. Table 1 shows various input parameters to QFA algorithm along with their values.

Kumar et al., studied end-to-end delay constraints using SDN switches [11] and they measured average packet processing time in a SDN switch to be in range of 3.2 μs to 4.1 μs [11], assuming average packet size as 1600 bytes [11]. Same values are considered for our research testing. Above mentioned μs values leads to packet service rate values ranging from 243902 to 312500 packets/second. Buffer size at the node can be found using $B = RTT \times \frac{C}{\sqrt{n}}$, where RTT is average round trip delay of the flow passing through the link, C is capacity of link and n is number of flows multiplexed on same link. In OSHI experiments, 'n' was found to be 29 with Colt Telecom topology. Assuming RTT as 250 milliseconds [23], value of B is 2,785,430 bits. With packet size as 1600 bytes [11], B was found to be around 218 packets.

Table 1: Simulation Parameters and their values

Parameter	**File where the parameter is defined**	**Range of values**
Bandwidth	flow_catalogue.json	0.0014 to 12.2 Mbps [22]
End-to-End Delay	flow_catalogue.json	50 to 75 milliseconds [18]
Packet_Loss	Hardcoded in source code	0.0001% [18]
Link_Capacity	links.json	60 Mbps [22]
Average_Packet_Size	Hardcoded in source code.	1600 bytes [11]
Packet_Service_Rate	nodes.json	243902 to 312500 packets/second
Node_Buffer_Size	nodes.json	220 to 230 packets

4.2 Reducing Flow Rejection Ratio

In CSPF phase of existing Flow Assignment algorithm [7], a huge amount of flows gets rejected when their shortest path cannot be computed. This is mentioned in section 3. Nearly two-thirds of requested flows i.e., 1470 out of 2198 flows gets rejected. When a flow is processed in CSPF phase, a temporary network graph is created in such a way that the network links whose residual capacity is not sufficient enough to support new flow's bandwidth (or) the links which are overloaded are pruned from temporary network graph, before invoking Dijkstra API on it. When Dijkstra API could not compute path between source and destination nodes, such flows are rejected. This is because links present in potential routing path might have been overloaded and removed from temporary graph. One way to solve link overload issue is to redistribute traffic across multiple links. Traffic redistribution can be easily achieved by considering first, the flows with smaller bandwidth values [24]. Hence we sort flows in ascending order as per their bandwidth value during initialization phase.

4.3 QoSBasedFlowAssignment Flowchart

The proposed QFA algorithm has five phases like Initialization, CSPF, Heuristic Reassignment, FlowLatencyCalculation and FlowReallocation. The first three phases are already present in existing design [7], while the last two phases are newly added in this work to handle delay logic. As shown in Figure 1, the design of these two phases are as follows:

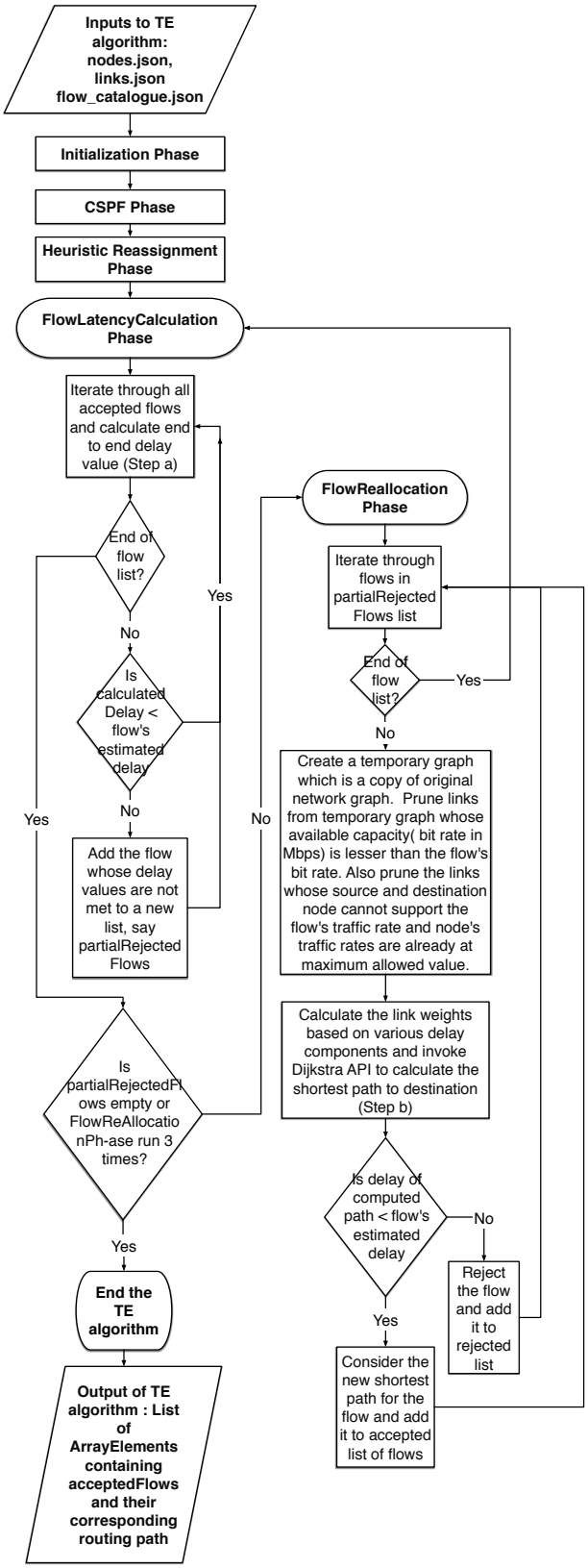

Fig. 1: FlowLatencyCalculation and FlowReallocation Phase

- *FlowLatencyCalculation Phase*: In this phase, end-to-end delay value is calculated for of all accepted flows as per formulae mentioned in section 4.1.2 (Defined as 'Step a' in Figure 1). Flows whose calculated delay is greater than expected delay are added to a list called 'partialRejectedFlows'.
- *FlowReallocation Phase*: Flows added to 'partialRejectedFlows' list are processed again to recompute their routing path with minimal delay value. Delay value of links itself is used as link weights to calculate shortest path. Delay components like T_{trans}, T_{queue}, $T_{propagation}$ and $T_{nodalProc}$ are calculated using formulae mentioned in Section 4.1.2. Summation of these delay values is used as link weights in network graph (Defined as 'Step b' in Figure 1). It is guaranteed that if shortest path computed at this stage does not meet delay requirement of flow, no optimal path exists for this flow and so it can be rejected. This phase is repeated for each flow in 'partialRejectedFlows' list.
  Once all flows of 'partialRejectedFlows' list are processed, execution returns to FlowLatencyCalculation phase. This is because, due to new routing paths and new loads added on the network links in FlowReallocation phase, its probable that end-to-end delay of few of already accepted flows might have crossed their limit. FlowLatencyCalculation and FlowReallocation phases runs in a loop until the list 'partialRejectedFlows' is empty which means all acceptedFlows meet their Delay requirements (or) for hard-set rule of 3 times, which was found to be sufficient enough to optimize all flows in experiment.

5 Simulation Results

This section compares QFA algorithm against existing OSHI algorithm [7] in various aspects like Global Network Crossing Time, Number of rejected flows and link loads which were measured from the algorithm's output. Both the algorithm were evaluated in OSHI platform using Colt Telecom topology (large-scale) [14]. Main steps to test the QFA algorithm are [15]

1. Download Colt Telecom topology from Topology Zoo [14].
2. Topology Parser transforms network topology to JSON files (nodes.json, links.json).
3. Random Demand Generator generates set of flow catalogues (flow_catalogue.json) using topology files.
4. Execute QFA algorithm from eclipse using nodes.json, links.json and flow_catalogue.json as input. Output is array elements containing accepted flows and their routing path.

5.1 Flow Rejection Ratio

Figure 2 shows number of rejected flows measured with existing algorithm [7] and proposed algorithm for same number of requested flows. As seen, existing algorithm [7] rejects 1470 flows among 2198 flows that were requested, whereas QFA algorithm rejects 880 flows for same number of requested flows. Sorting logic added has reduced flow rejection ratio to around half its size. Section 4.2 explains in detail the reason behind this reduction. Assuming algorithm is scheduled for nightly periodic runs, with a single run the proposed design enables 1318 (2198 minus 880) traffic

flows at a time, when existing algorithm [7] enables only 728 (2198 minus 1470) flows in network.

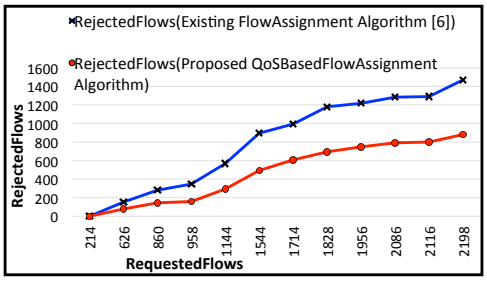

Fig. 2: Number of Rejected Flows in Existing Vs Proposed Solution (Colt Telecom)

5.2 Global Network Crossing Time

The proposed design sorts the flows in ascending order as per their bandwidth during Initialization phase. Sorting logic along with delay based optimization, allows algorithm to calculate routing paths with minimal delay value. This leads to reduction in overall network crossing time. Graph in Figure 3 shows global network crossing time value measured by old and new algorithm for various number of requested flows. Global network crossing time which is referred as 'T' is weighted average of delays seen on the links and is measured using (2). It can be clearly seen

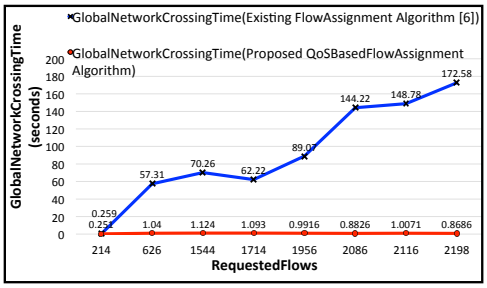

Fig. 3: Global Network Crossing Time in Existing Vs Proposed Solution (Colt Telecom)

in Figure 3 that there is a drastic drop in T value measured with proposed solution than existing algorithm [7]. For instance, value of T measured with existing algorithm is 172.58 seconds and with QFA algorithm is 0.8686 seconds, when number of requested flows is 2198. This reduction is due to reason that the traffic gets evenly distributed across links with new design compared to existing one [7], which is explained in detail in following section.

Impact of link load on its delay value:

As mentioned in (6), when each network link is considered as M/M/1 system with packet service rate following an exponential distribution, transmission delay encountered in link can be generalized and given by,

$$T_{trans} = \frac{1}{LinkCapacity - LinkLoad} \tag{9}$$

From (9), it is obvious that with link capacity being a constant value, transmission delay value increases with increase in link load value and even approaches infinity when link load is same as link capacity. This rapid increase of delay value as load approaches capacity of link [25] is illustrated in Figure 4. In Figure 4 increase in

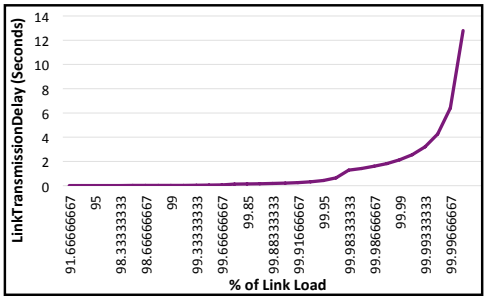

Fig. 4: Effect of Link load % on its Transmission Delay value (Assuming Link capacity 60 Mbps)

transmission delay value with respect to link load is very less until the link is loaded to 99.6% of its capacity. When link is overloaded and crosses 99.95% which can be referred as alarming limit, there is a rapid rise in delay value even for a small amount of increase in link load. The existing algorithm [7] overloads few links in this alarming limit i.e., beyond 99.95%, thereby resulting in a huge global network crossing time value as shown in Figure 3. On other hand, QFA algorithm prevents links getting loaded more than alarming limit. This is illustrated in Figure 5. When

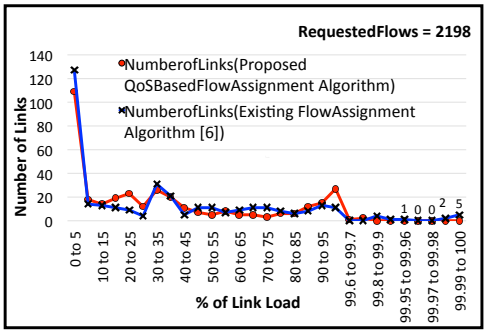

Fig. 5: Number of links and their load % (Colt Telecom, requested flows = 2198)

number of requested flows is 2198, proposed design does not load any links beyond 99.95% (alarming limit), whereas existing algorithm loads 8 links beyond this point (Figure 5). Although 8 is very minimal number compared to total number of 354 links, these 8 links have a very higher delay values, which is more predominant than other link's delay values (as in Figure 4). These critical 8 links are reason for global networking crossing time being very high with existing FlowAssignment algorithm [7], as shown in Figure 3.

Above mentioned fact was proved to be right by extending our testing to monitor link load % values for different number of requested flows. Results shown in Figure 6 confirms the existing design [7] overloads links beyond alarming limit (99.95%), which is avoided with proposed solution. QFA algorithm prevents link overload issue by redistributing traffic across links. Flows with smaller bandwidths are found to be best suited for fair distribution of traffic across links [24] and thus sorting flows in ascending order of their bandwidth value (proposed design) helped with this redistribution. Furthermore, delay based routing optimization also contributed to reduction in global network crossing time.

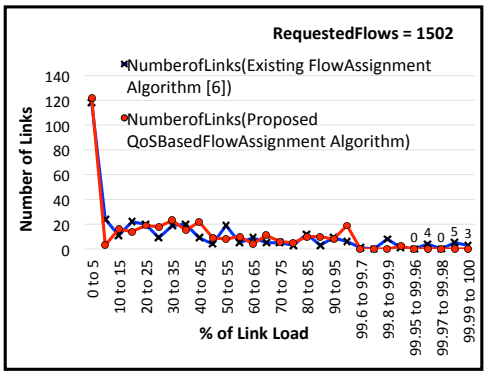

Fig. 6: Number of links and their load % (Colt Telecom, requested flows = 1502)

6 Conclusion

In this paper, a new QoS based flow optimization algorithm for TE in SDN was designed and implemented using OSHI platform. The experiment provides promising results and proves new algorithm's efficiency compared to the existing algorithm. Our study used M/M/1 and M/M/1/K queuing systems to model SDN network, whereas real traffic flows can be more bursty than traffic represented by these queuing models. Various other queuing distributions can be validated in future to match more closer to the real time traffic and algorithm can be further tested in a small-scale network using real SDN switches.

References

1. D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.
2. M. A. Hassan, Q.-T. Vien, and M. Aiash, "Software defined networking for wireless sensor networks: A survey," *Advances in Wireless Communications and Networks*, vol. 3, no. 2, pp. 10–22, May 2017.
3. R. C. Ramirez, Q.-T. Vien, R. Trestian, L. Mostarda, and P. Shah, "Multi-path routing for mission critical applications in software-defined networks," in *Proc. EAI INISCOM 2018*, Da Nang, Vietnam, Aug. 2018.

4. Cisco. Introduction to segment routing. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/seg_routing/configuration/xe-3s/segrt-xe-3s-book/intro-seg-routing.pdf
5. L. Davoli, L. Veltri, P. L. Ventre, G. Siracusano, and S. Salsano. OSHI homepage. [Online]. Available: http://netgroup.uniroma2.it/twiki/bin/view/Oshi
6. L. Davoli, L. Veltri, P. L. Ventre, G. Siracusano, and S. Salsano. (2015, Dec.) Traffic engineering with segment routing: Sdn-based architectural design and open source implementation. Extended version of poster paper accepted for EWSDN 2015. [Online]. Available: https://arxiv.org/abs/1506.05941v4
7. P. L. Ventre. (2015, Jun.) Existing Flow assignment algorithm in OSHI. [Online]. Available: https://github.com/netgroup/SDN-TE-SR-tools/blob/master/java-te-sr/src/it/unipr/netsec/sdn/algorithm/FlowAssignmentAlgorithm.java
8. W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S.-J. Lee, and P. Yalagandula, "Automated and scalable qos control for network convergence," in *Proc. Internet Network Management Workshop / Workshop on Research on Enterprise Networking (INM/WREN)*, Apr. 2010.
9. S. Tomovic and I. Radusinovic, "Fast and efficient bandwidth-delay constrained routing algorithm for sdn networks," in *NetSoft Conference and Workshops (NetSoft), 2016 IEEE*, Seoul, South Korea, Jun. 2016, pp. 303–311.
10. Open Networking Foundation. (2013, Oct.) Openflow switch specification. [Online]. Available: https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf
11. R. Kumar, M. Hasan, S. Padhy, K. Evchenko, L. Piramanayagam, S. Mohan, and R. B. Bobba, "Dependable end-to-end delay constraints for real-time systems using sdns," in *15th International Workshop on Real-Time Networks*, Dubrovnik, Croatia, Jun. 2017.
12. A. S. Hamood. (2016) Simulator.ppt. [Online]. Available: https://www.researchgate.net/publication/301887282_most_Simulator_used_in_Software_Defined_Networking_SDN_and_Cognitive_Radio_Network_CRN
13. Mininet. [Online]. Available: http://mininet.org/
14. The University of ADELAIDE. Topology zoo. [Online]. Available: http://www.topology-zoo.org/dataset.html
15. L. Davoli, L. Veltri, P. L. Ventre, G. Siracusano, and S. Salsano. (2015) "OSHI github". [Online]. Available: https://github.com/netgroup/SDN-TE-SR-tools
16. P. L. Ventre and S. Salsano. (2014) OSHI Quaterly Reports QR.2. [Online]. Available: http://netgroup.uniroma2.it/twiki/pub/Oshi/WebHome/qr2_2_ventre.pdf
17. L. Kleinrock, *Queueing Systems Volume 1: Theory*. Wiley-Interscience, 1975.
18. *Service Level Agreement*, COLT Telecommunications, 2001. [Online]. Available: https://www.rtr.at/uploads/media/24875_SLA_CPE.pdf
19. L. Kleinrock, *Queueing Systems, Volume 2: Computer Applications*. Wiley-Interscience, 1976.
20. P.Coiner. (2011, Jan.) Calculating the propogation delay of coaxial cable. GPS Source. [Online]. Available: https://cdn.shopify.com/s/files/1/0986/4308/files/Cable-Delay-FAQ.pdf
21. G. R. Dattatreya, *Performance Analysis of Queuing and Computer Networks*. CRC Press, 2008.
22. L. Davoli, L. Veltri, P. L. Ventre, G. Siracusano, and S. Salsano. (2016, Apr.) OSHI virtual image OSHI_VM7b. [Online]. Available: http://netgroup.uniroma2.it/twiki/bin/view/Oshi
23. G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *SIGCOMM '04 Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, Portland, Oregon, USA, Sep. 2004, pp. 281–292.
24. S. Köhler and A. Binzenhöfer, *Providing Quality of Service in Heterogeneous Environments*. Berlin, Germany: Elsevier, 2003, vol. 5, pp. 21–30.
25. S. Weinstein, *The multimedia internet*. Springer, 2005.