Ensuring end-to-end QoS based on multi-paths routing using SDN technology

Diego Leonel Cadette Dutra [1], Miloud Bagaa [1], Tarik Taleb[1] and Konstantinos Samdanis[2]
[1] Dep. of Communications and Networking School of Electrical Engineering, Aalto University, Espoo, Finland
[2] Huawei European Research Center, Munich, Germany
Emails:{firstname.lastname}@aalto.fi; konstantinos.samdanis@huawei.com

Abstract—**Software Defined Networking (SDN) is an emerging technology that will play an important role in enabling 5*G*, since it offers enhanced network management features. SDN allows programmability of the control plane, abstracting the underlying network infrastructure for applications and network services, e.g. through the OpenFlow protocol. In this paper, we propose a solution that enables the end-to-end Quality of Service (QoS) based on the queue support in OpenFlow, allowing an operator with a SDN-enabled network to efficiently allocate the network resources according to the users' demands, reducing or even eliminating the need for over-provisioning. For each traffic flow, the proposed solution guarantees the required end-to-end QoS, while efficiently managing the utilization of open virtual switches (OVSs), which leads to reduced cost. The cost could be also reduced as a fewer number of OVSs are needed, which are enabled in different data centers. For ensuring these objectives, the proposed solution explores the strength of multi-path routing based on SDN with a precise bandwidth allocation. The obtained results show the efficiency of the proposed solution in terms of cost and execution time.**

I. Introduction

Mobile network operators have been commonly accommodating high QoS applications by over provisioning network resources to assure the desired service performance considering peak demands. Such a process becomes complex and more challenging with the evolution of a plethora of new applications and services with stringent performance demands [1], [2], [3], [4]. To efficiently address such a challenge in the 5G era, network programmability solutions such as Software Defined Networks (SDN) [5], [6] are introduced into the mobile network architecture to bring flexibility in the resource allocation process [7], [8]. In particular, this paper explores the QoS queue support in OpenFlow that allows an operator with a SDN-enabled transport network to efficiently allocate network resources reflecting evolving users' demands [9]. Such an approach enables a fine tuning in resource allocation improving the network utilization, while assuring QoS provisioning, in terms of bandwidth and jitter. Nevertheless, the efficiency of the aforementioned solution requires a frequent recomputation of the allocated resources, ideally considering every arriving and hand-over user.

We are interested in computing a network configuration that can be applied via a SDN controller, e.g., ONOS. Such a configuration will use the queue control available on version 1.3 of the OpenFlow protocol to allocate an end-host connection that may simultaneously use multiple paths. Such multipath capability is transparent to end-hosts, which are not aware of flows splitting. For example, a mobile user that streams a Video on Demand (VoD) via the proposed SDN network, requires the mobile operator to allocate the desired bandwidth once the streaming request is accepted and initiates the streaming upon receiving such a confirmation. This simple example highlights an important promise of our solution regarding the establishment of QoS, which is in need of admission control, a process that may use application level information as described in our example or be applied based on the user profile, based, in turn, on the contract with the mobile network operator.

This paper introduces an optimization solution for the configuration of a SDN-enabled mobile network, and develops and evaluates the proposed solution. It also experimentally demonstrates that in case of our proposed solution, the computational cost increases linearly with the number of end-nodes and exponentially with the number of OVSs.

The remainder of this paper is organized as follows. Section II presents the related work. Section III describes our network model and our proposed solution. Section IV presents the performance evaluation and our results analysis. Finally, Section V concludes the paper.

II. Related work

Network programmability and SDN control is entering a mature phase with several contributions focusing on efficient resource allocation, especially for mobile systems. Egilmez et al. [10] introduces a solution using OpenFlow QoS to guarantee the end-to-end bandwidth for video streaming. The proposed solution assumes that the SDN controller manages the complete network with the clients being stationary during the entire transmission time. Egilmez and Tekalp [11] extended such initial proposal introducing multiple controllers considering a distributed scenario. In this environment, every Autonomous System (AS) performs an optimal QoS routing and, in a subsequent step, uses a summarized network view to perform inter-AS QoS routing. The authors have shown that the proposed distributed solution approximates the global optimum while providing scalability, but as in their original paper, it still assumes stationary clients for the entire video transmission.

Sharma et al. [12] proposed a QoS framework based on SDN using the Floodlight controller. Their proposal uses a single controller per AS that communicates with a bandwidth broker through a northbound API, which is responsible for maintaining the respective policies and for performing the SLA negotiation with end customers or neighboring brokers. Celenlioglu and Mantar [13] proposed a routing and resource management model considering pre-established paths with resource reservation for SDN-based intra-domain networks. Such a scheme improves routing scalability and decreases the admission time assuring QoS guarantees to stationary nodes.

Jinyao et al. [14] proposed HiQoS, a SDN-based multipath solution to guarantee QoS on computer networks. HiQoS uses the OpenFlow queuing mechanisms to implement bandwidth guarantees for different traffic. In their solution, the multipath is achieved using a modified Dijkstra algorithm with QoS constraints. Tariq and Bassiouni [15] proposed a QoS-Aware algorithm for optical SDN considering Multipath-TCP (MPTCP) in data center environments, named QAMO-SDN. Such a proposal precomputes multiple paths based on Dijkstra's algorithm and selects P paths between two end-nodes. Their simulations showed that QAMO-SDN closely approximates MPTCP ($P = 4$) without compromising the throughput of the low priority burst traffic.

Huang et al. proposed [16] and experimentally evaluated [17] an SDN multipath solution for GridFTP based on Dijkstra's algorithm, which is implemented in Trema [18]. Hussain et al. [19] evaluated a hashed based multipath solution using Floodlight, with their proposal concentrating on scheduling the available flows using a hash function over a set of precomputing paths. Also, Basit et al. [20] proposed a cross-layer coordination among ISPs, peering at multiple IXPs, that enables the use of the available multipath between the end-hosts. MPTCP can leverage on these different paths to enhance the throughput. Unlike other proposals, our soltuion adopts the SDN paradigm to enable an end-to-end Quality of Service (QoS) based on queue support in OpenFlow and multi-path routing, allowing an efficient resource alllocation considering service demands.

III. Network model and problem formulation

Let $G(V,E,\mathcal{W})$ denote a weighted graph, where V represents a set of nodes and E the set of edges in the network. $V = \mathcal{C} \cup \mathcal{O} \cup \mathcal{S}$, where $\mathcal{C}$, $\mathcal{O}$, and S denotes a set of clients, a set of open virtual switches, and the set of servers in the network, respectively. Each edge is associated with a weight $\mathcal{W}$, where $\mathcal{W}_{u,v}$ of an edge $(u,v) \in E$, denotes the bandwidth capacity between nodes u and v.

Table I
Notations used.

Notation	Description
$\mathcal{C}$	A set of clients in the network.
$\mathcal{O}$	A set of open virtual switches in the network.
$\mathcal{S}$	A set of servers in the network.
$G(V,E,\mathcal{W})$	A graph that shows the network topology, where $V = \mathcal{C} \cup \mathcal{O} \cup \mathcal{S}$, E denotes the set of edges. Meanwhile, $\mathcal{W}$ denotes the bandwidth communication between different nodes in V. $\omega_{i,j} \in \mathcal{W}$ denotes the bandwidth capacity between $i,j \in V$.
$X_{i,j}$	A decision boolean variable that shows if a node i selects j as parent.
Y_o	A decision boolean variable that shows if a switch $o \in \mathcal{O}$ is selected to forward the traffic or not.
$T_{i,j}$	A real number variable represents the amount of traffic that would be forwarded from i to j.
$\eta(u)$	A function that returns the neighbors of node u in graph G.
$\mathcal{F}^{s}_{i,j}$	An integer variable that mimics packet flow generated from different clients towards server s.

We assume that each client $c \in \mathcal{C}$ can be in the proximity of a set of OVSs, that act as base stations or eNodeBs. A client can be connected to a set of OVSs at any given time. Each switch in the network is associated with a set of clients and neighbor switches. Let $\eta(u)$ represent the set of neighbors of a node $u \in V$. Assuming that each client requires a single specified service at a time from a corresponding server with certain bandwidth, we are interested in handling different client requests that require different QoS. Without loss of generality, if a client is interested in $\mathcal{N}$ services, simply we replicate that client by $\mathcal{N}$, with each one concentrating on a specified service. We denote by $\mathcal{S}_s$ for $s \in \mathcal{S}$, the set of clients that consume a service offered by a particular server, formally $\bigcup_{s \in \mathcal{S}} \mathcal{S}_s = \mathcal{C}$. Each client $c \in \mathcal{C}$ requires a specified bandwidth $\mathcal{B}_c$ for different services. The notations used in this paper are summarized in Table I.

A. *Proposed solution: Full paths re-computation*

Hereafter, we describe our solution to guarantee the end-to-end QoS in our model. In this solution, we recompute our OVS configuration every time a new client arrives in our network or when a mobile user is handed-over to a new eNodeB with the aim of reducing the number of OVS activated in the core network. For all $u \in \mathcal{C} \cup \mathcal{O}$ and $v \in \mathcal{O} \cup \mathcal{S}$, we define the following variables:

$\mathcal{X}_{u,v}$, which is a Boolean variable that shows if u selects v as its successor.

$$\mathcal{X}_{u,v} = \begin{cases} 1 & \text{If } u \text{ selects } v \text{ as parent} \\ 0 & \text{Otherwise} \end{cases} \tag{1}$$

$\mathcal{T}_{u,v}$, which is a real number that shows the amount of traffic that can be forwarded from u to v.

For each switch $o \in \mathcal{O}$, we define the following variable:

$$\mathcal{Y}_o = \begin{cases} 1 & \text{If } o \text{ is selected to forward the different traffic} \\ 0 & \text{Otherwise} \end{cases} \tag{2}$$

For each server $s \in \mathcal{S}$, we define a matrix $\mathcal{F}^s$ of integer variables that mimic the traffic generated and forwarded to that server. Each element $\mathcal{F}^s_{i,j}$ represents the number of flows that shall be forwarded from i to j, whereas $i \in \mathcal{C} \cup \mathcal{O}$ and $j \in \mathcal{O} \cup \mathcal{S}$.

In the Objective Function 3, we aim to minimize the number of switches used when all paths between clients and requested servers are super-positioned. Meanwhile, the constraints used ensure the following conditions: Constraint 4 ensures that each client in the network forwards its traffic through only one OVS switch (parent); Constraint 5 represents the traffic aggregated from a Client to its Servers that go through its parent; Constraint 6 ensures that all traffic that goes in an OVS from its neighbors $\eta(i)$, clients or other OVSs must be output by that said OVS to $\eta(i)$, be it servers or other OVSs; Constraint 7 ensures that no extra network traffic is created in the OVSs for the clients; Constraint 8 and Constraint 9 ensure that an OVS is only selected to forward the traffic if it is a parent in the path between a client and a server; Constraint 10 ensures that the number of flows arriving in a server is equal to the number of clients that requested that server; Constraint 11 limits to one the number of flows a client can generate to each server in our topology; Constraint 12 ensures that all traffic $\mathcal{F}$ from a client directed to a server S goes to that server; Constraint 13 forces the generated flow to be routed only within the constructed tree, from each node i to its parent j, avoiding loops.

$$\boldsymbol{min} \sum_{\forall i \in \mathcal{O}} \mathcal{Y}_i \tag{3}$$

s. t.

$$\forall i \in \mathcal{C} : \sum_{\forall j \in \eta(i)} \mathcal{X}_{i,j} = 1 \tag{4}$$

$$\forall i \in \mathcal{C}, \forall j \in \eta(i) : \mathcal{T}_{i,j} = \sum_{\forall s \in \mathcal{S} \wedge i \in \mathcal{C}_s} \lambda^s_i \times \mathcal{X}_{i,j} \tag{5}$$

$$\forall i \in \mathcal{O} : \sum_{\forall j \in \eta(i) \cap (\mathcal{C} \cup \mathcal{O})} \mathcal{T}_{j,i} = \sum_{\forall j \in \eta(i) \cap (\mathcal{O} \cup \mathcal{S})} \mathcal{T}_{i,j} \tag{6}$$

$$\forall i \in \mathcal{C} \cup \mathcal{O}, \forall j \in \eta(i) \cap (\mathcal{O} \cup \mathcal{S}) : \mathcal{T}_{i,j} \leq \mathcal{W}_{i,j} \times \mathcal{X}_{i,j} \tag{7}$$

$$\forall i \in \mathcal{O}, \forall j \in \eta(i) \cap (\mathcal{O} \cup \mathcal{S}) : \mathcal{X}_{i,j} \leq \mathcal{Y}_i \tag{8}$$

$$\forall i \in \mathcal{O}, \forall j \in \eta(i) \cap (\mathcal{O} \cup \mathcal{C}) : \mathcal{X}_{j,i} \leq \mathcal{Y}_i \tag{9}$$

$$\forall i \in \mathcal{S} : \sum_{\forall j \in \eta(i)} \mathcal{F}^i_{j,i} = |\mathcal{C}_i| \tag{10}$$

$$\forall s \in \mathcal{S}, \forall i \in \mathcal{C}_s : \sum_{\forall j \in \eta(i)} \mathcal{F}^s_{i,j} = 1 \tag{11}$$

$$\forall i \in \mathcal{O}, \forall s \in \mathcal{S} : \sum_{\forall j \in \eta(i) \cap (\mathcal{C} \cup \mathcal{O})} \mathcal{F}^s_{j,i} = \sum_{\forall j \in \eta(i) \cap (\mathcal{O} \cup \mathcal{S})} \mathcal{F}^s_{i,j} \tag{12}$$

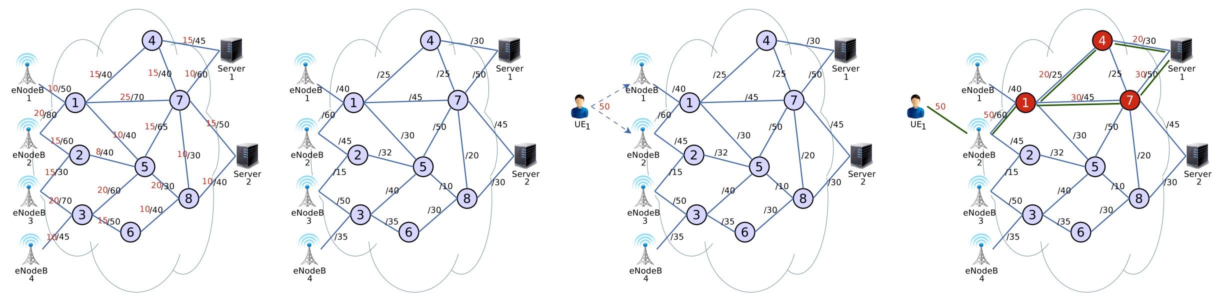
$$\forall s \in \mathcal{S}, \forall i \in \mathcal{C} \cup \mathcal{O}, \forall j \in \mathcal{O} \cup \mathcal{S} : 0 \leq \mathcal{F}^s_{i,j} \leq |\mathcal{C}_s| \times \mathcal{X}_{i,j} \tag{13}$$

Fig. 1 serves as a detailed example that illustrates the operation of our proposed solution. The figure represents a simple mobile network, which consists of four eNodeBs, a set of OVSs numbered from 1 to 8 and two servers. Also note that for clarity, we suppressed the SDN controller from Fig. 1. A mobile user that wants to access a server using our network needs to attach to one of the available eNodeBs.

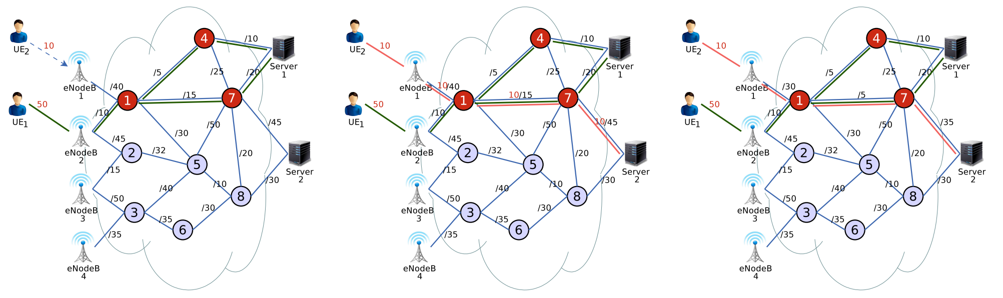
Fig. 1(a) illustrates the network in its initial configuration, showing the bandwidth resources partially in use as highlighted by the red numbers. Based on this topology, we compute our reference graph G that removes all used resources as depicted in Fig. 1(b). Fig. 1(c) depicts the network attach moment for UE 1, which is interested in a service from $Server$ 1. That service requires $50Mbps$ bandwidth, which enforces the UE 1 to attach to $eNodeB$ 2, which can comply with this demand. We assume, without compromising our solution, that simultaneously UE 2 arrives, which needs to initiate (Fig. 1(e)) a new request to $Server$ 2. Executing our optimization solution considering both clients and G as input parameters, we get as output the configuration represented by Fig. 1(f). The links allocated for UE 1 and UE 2 are shown in black and in red, respectively, while the activated OVS are depicted in red.

Fig. 1(g) presents our network after both mobile users have been accepted. For UE 1, the required $50Mbps$ has been allocated at $eNodeB$ 2, via multiple paths that facilitate communication between the corresponding UE and $Server$ 1 using both OVS 4 and 7. For UE 2, a single path is allocated towards $Server$ 2 to provide the required resources. This simple example illustrates the constraints that were described in our optimization solution. As stated in Equation 3, we minimize the number of activated OVS in our final configuration as Fig. 1(g) shows. Constraint 4 can be seen in practice in Figs. 1(d) and 1(f). All input traffic in a OVS is equal to the said OVS output traffic, as required by constraint 6.

Algorithm 1 summarizes these constraints. For every new user that arrives or is handed-over, the path of each client has to be recomputed. This algorithm is trigged by the arrival of a new client or by the mobility of an existing one. The input parameters for our solution are the graph G considering the new client(s) that request a path and an updated version of the existing paths from



(a) Network view showing bandwidth in use in red and total bandwidth in black. (b) Computed graph of the topology removing the bandwidth currently in use. (c) *UE* 1 arrives and requests 50*Mpbs* from eNodeBs 1 and 2. (d) Network operator allocates 50*Mpbs* using multiple paths in the transport network between *UE* 1 and *Server* 1.



(e) Arrival of *UE* 2. (f) Network resources allocated for *UE* 1 and *UE* 2. (g) Network resources reference graph is recalculated.

Figure 1. Example of our proposed solution for E2E QoS guarantee.

Algorithm 1 Algorithm to Compute Network Configuration.

```
Require:
    $\mathcal{G}$: Graph representation of the network.
    $\mathcal{C}$: List of new/updated clients.
    $\mathcal{S}$: List of server in the network.
  while true do
    if $c$ in $C$ == ($new$ or $moved$) then
      return $computeSDNconfig(\mathcal{G}, \mathcal{C}, \mathcal{S})$
    end if
  end while
```

each client to the corresponding server on the network. Every time the algorithm returns a new configuration, the main control loop is re-executed waiting for any update in our network topology.

IV. Simulation Setup and Result Analysis

This section introduces our simulation setup and presents the obtained results. Our experiments were conducted on a multi-core server, described in Table II.

Table II
Hardware Configuration.

Type	Configuration
CPU	Dual Intel Xeon E5-2680 v3 @ 2.5GHz
Memory	256GB
Linux	Ubuntu 16.04
Kernel	4.4.0-72

We evaluated the behavior of our solution by varying the number of clients, servers, and OVSs. For each one of these scenarios, we run 100 repetitions, changing the client position and computed the number of OVS's used as well as the computational times. Hereafter, we present the mean and 95% Confidence Interval of the number of OVS's used and the computational cost in seconds. Moreover, in each of these scenarios, the positions of clients, OVSs and servers were uniformly distributed. As described in Section III, a client/server can be connected to only one OVS.

Fig. 2 presents the behavior of our solution when we vary the number of clients from 5 to 100, while keeping both the number of OVSs and servers constant, i.e. running our simulation with 25 OVS and 10 Servers. In Fig.2, the left Y-axis shows the number of OVSs used, while the right Y-axis shows the time in seconds spent to find the solution and the X-axis represents the number of clients in the network. Fig.2 shows that the number of active OVSs increases from 4 to 6 when we vary the number of clients 20 fold, while beyond 20 clients, the mean number of OVSs starts to stabilize. Looking only at

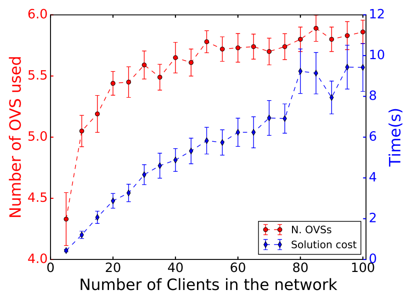


Figure 2. Varying the number of clients from 5 to 100.

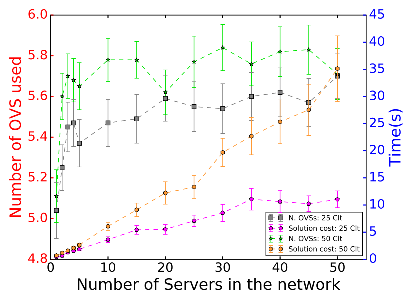


Figure 4. Varying servers and clients.

the steady state part of our simulation, the mean number of OVSs activated is 5.695 with a standard deviation of 0.134. Meanwhile, the results from the computational time show that as the number of clients increases, the computational cost increases linearly from 0.436 s to 9.421 s. The linear regression parameters for the mean computational time of our experiment are 0.885 and 0.0897, as α and β, respectively.

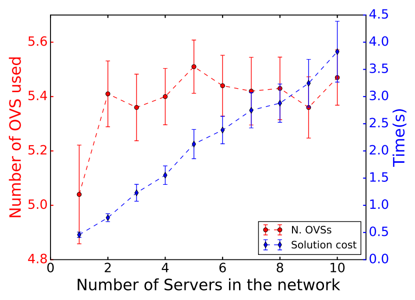


Figure 3. Varying servers from 1 to 10 with 25 clients.

Fig. 3 shows our evaluation with respect to the number of servers, keeping the same representations as before for the Y-axes, while the X-axis now shows the number of servers in our network. We vary the number of Servers from 1 to 10, while the number of clients and OVSs are set to 25. It is observed that our results start to stabilize after 2 servers, with a mean number of activated OVSs of 5.416 and a standard deviation of 0.0889. In this figure, the mean computational time to find the solution increases linearly with the number of servers in the network: the linear regression of these samples returned 0.132 and 0.362, as α and β, respectively.

The obtained results suggest that the computational cost increases linearly with the number of end-nodes. To better understand this behavior in our proposed solution, we devised two additional scenarios, whereby we vary the number of servers and clients. Fig. 4 illustrates that for all configurations of Server/Client, the mean number of activated OVSs is close to 5.5, which implies that the number of activated OVSs has a high positive correlation with the density of OVSs in the network as this parameter was constant across these scenarios. Moreover, as suggested by our previous results, the computational cost still increases linearly with the number of end-hosts, while it has a stronger correlation with the number of clients than the number of servers, as the comparison between the curves of 50 clients in this figure and Fig. 3 suggests. This behavior is the result of the number of flows being directly proportional only to the number of clients in our network.

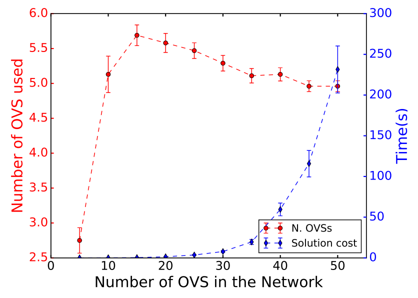


Figure 5. Varying OVSs from 5 to 50 in 40x40 grid.

Finally, in Fig. 5, we run our simulations with 10 servers and 25 clients while varying the number of activated OVSs between 5 and 50, over a grid of 40x40. The number of activated OVSs is around 5 with a mean value of 5.258 and a standard deviation of 0.251. As for the negative slope that we see beyond a network with 20 OVSs, it is a consequence of the increase in the OVSs density in our experiment, which raises the probability of smaller paths connecting clients and servers. Furthermore, the increase in the OVSs density also causes an increase in the number of links between them, which also increases the number of possible paths and that is the main reason for the exponential growth in computational cost.

This assumption is further corroborated by the results of Fig. 6 whereby we show the same experiment but now

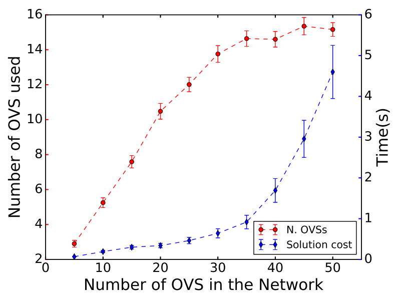


Figure 6. Varying OVSs from 5 to 50 in a 80x80 grid.

over a 80x80 grid. The increase in the number of active OVSs as the OVSs density increases is expected, since we are only considering fully connected networks as input, which reduces the number of valid input topologies. As the confidence interval size suggests the curve starts to stabilize around 16 OVSs, which is 3 times higher than in the experiment of Fig. 5, and close to a 4 times increase of our simulated area. Still in Fig. 6, we can see the same exponential behavior in the computational cost, however with a lower threshold. This indicates that the increase in the number of links between the OVSs due to the higher density of OVSs in our network is the main component in the computational time to find the optimization for our proposed solution, as described in Section III-A.

V. Conclusions

This paper introduces a SDN-based solution to assure QoS guarantees without over-committing the network resources to the users that request high bandwidth and low jitter. This is achieved through a multi-path approach combined with a precise bandwidth allocation through SDN QoS support. Our simulations showed that our solution can find a network configuration that offers theses QoS guarantees, while minimizing the number of active OVSs in the network. Furthermore, our results show that the computational cost to find a solution increases linearly with the number of end-nodes, and with a positive correlation with the number of clients. Another important finding was that in its present form, our solution computational cost has an exponential positive correlation with the OVSs density in our network.

Acknowledgement

This work was partially supported by the European Union's Horizon 2020 research and innovation programme under the 5G!Pagoda project with grant agreement No. 723172.

References

[1] "The Mobile Economy 2017," GSMA Intelligence, Tech. Rep., 2017, accessed: 2017-04-13.

[2] T. Taleb, M. Corici, C. Parada, A. Jamakovic, S. Ruffino, G. Karagiannis, and T. Magedanz, "Ease: Epc as a service to ease mobile core network deployment over cloud," *IEEE Network*, vol. 29, no. 2, pp. 78–88, March 2015.

[3] T. Taleb, "Toward carrier cloud: Potential, challenges, and solutions," *IEEE Wireless Communications*, vol. 21, no. 3, pp. 80–91, June 2014.

[4] T. Taleb, B. Mada, M. I. Corici, A. Nakao, and H. Flinck, "Permit: Network slicing for personalized 5g mobile telecommunications," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 88–93, May 2017.

[5] S. ONF, "Architecture Issue 1.1," Tech. Rep., 2016, accessed: 2017-04-13.

[6] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[7] A. Ksentini, M. Bagaa, T. Taleb, and I. Balasingham, "On using bargaining game for optimal placement of sdn controllers," in *2016 IEEE International Conference on Communications*, May 2016, pp. 1–6.

[8] A. Ksentini, M. Bagaa, and T. Taleb, "On using sdn in 5g: The controller placement problem," in *2016 IEEE Global Communications Conference*, Dec 2016, pp. 1–6.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[10] H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp, "Openqos: An openflow controller design for multimedia delivery with end-to-end quality of service over software-defined networks," in *The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, Dec 2012, pp. 1–8.

[11] H. E. Egilmez and A. M. Tekalp, "Distributed qos architectures for multimedia streaming over software defined networks," *IEEE Transactions on Multimedia*, vol. 16, no. 6, pp. 1597–1609, Oct 2014.

[12] S. Sharma, D. Staessens, D. Colle, D. Palma, J. Gonçalves, R. Figueiredo, D. Morris, M. Pickavet, and P. Demeester, "Implementing quality of service for the software defined networking enabled future internet," in *2014 Third European Workshop on Software Defined Networks*, Sept 2014, pp. 49–54.

[13] M. R. Celenlioglu and H. A. Mantar, "An sdn based intra-domain routing and resource management model," in *2015 IEEE International Conference on Cloud Engineering*, March 2015, pp. 347–352.

[14] J. Yan, H. Zhang, Q. Shuai, B. Liu, and X. Guo, "Hiqos: An sdn-based multipath qos solution," *China Communications*, vol. 12, no. 5, pp. 123–133, May 2015.

[15] S. Tariq and M. Bassiouni, "Qamo-sdn: Qos aware multipath tcp for software defined optical networks," in *2015 12th Annual IEEE Consumer Communications and Networking Conference*, Jan 2015, pp. 485–491.

[16] C. Huang, C. Nakasan, K. Ichikawa, and H. Iida, "A multipath controller for accelerating gridftp transfer over sdn," in *2015 IEEE 11th International Conference on e-Science*, Aug 2015, pp. 439–447.

[17] ——, "An sdn-based multipath gridftp for high-speed data transfer," in *2016 IEEE 36th International Conference on Distributed Computing Systems*, June 2016, pp. 763–764.

[18] H. Shimonishi, Y. Takamiya, Y. Chiba, K. Sugyo, Y. Hatano, K. Sonoda, K. Suzuki, D. Kotani, and I. Akiyoshi, "Programmable network using openflow for network researches and experiments," 2012.

[19] S. A. Hussain, S. Akbar, and I. Raza, "A dynamic multipath scheduling protocol (dmsp) for full performance isolation of links in software defined networking (sdn)," in *2017 2nd Workshop on Recent Trends in Telecommunications Research*, Feb 2017, pp. 1–5.

[20] A. Basit, S. B. Qaisar, H. R. Syed, and M. Ali, "Sdn orchestration for next generation inter-networking: A multipath forwarding approach," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2017.