Scalable Guaranteed-Bandwidth Multicast Service in Software Defined ISP networks

Hardik Soni, Thierry Turletti, Walid Dabbous, Hitoshi Asaeda

Scalable Guaranteed-Bandwidth Multicast Service in Software Defined ISP networks

Hardik Soni, Walid Dabbous, Thierry Turletti
Université Côte d'Azur, Inria, France
Email: *firstname.lastname*@inria.fr

Hitoshi Asaeda
NICT, Japan
Email: asaeda@nict.go.jp

***Abstract*—New applications where anyone can broadcast video are becoming very popular on smartphones. With the advent of high definition video, ISP providers may take the opportunity to propose new high quality broadcast services to their clients. Because of its centralized control plane, Software Defined Networking (SDN) seems an ideal way to deploy such a service in a flexible and bandwidth-efficient way. But deploying large scale multicast services on SDN requires smart group membership management and a bandwidth reservation mechanism to support QoS guarantees that should neither waste bandwidth nor impact too severely best effort traffic. In this paper, we propose a Network Function Virtualization based solution for Software Defined ISP networks to implement scalable multicast group management. Then, we propose the Lazy Load balancing Multicast (L2BM) routing algorithm for sharing the network capacity in a friendly way between guaranteed-bandwidth multicast traffic and best-effort traffic. Our implementation of the framework made on Floodlight controllers and Open vSwitches is used to study the performance of L2BM.**

I. Introduction

The massive increase of live video traffic on the Internet and the advent of Ultra High Definition (UHD) videos put great strain on ISP networks. These networks follow a hierarchical structure to provide Internet access to millions of customers spread over large geographical areas and connected through heterogeneous access technologies and devices. Recently, new over-the-top (OTT) applications where anyone can broadcast its own channel like with Periscope or Facebook Live Stream are becoming very popular on smartphones. To satisfy their clients and attract new ones, ISP providers may decide to offer them new services for supporting upcoming high-quality multicast applications at home. One solution is to use built-in multicast within their infrastructure to implement flexible, bandwidth-efficient and scalable multicast delivery services. This may enable efficient deployment of many-to-many broadcast services such as Periscope. But so far, multicast has failed to achieve Internet-wide support [1], and even for the limited deployment in managed networks for services like IPTV, it requires complex integration of specialized multicast enabled routers and protocols, traffic engineering and QoS mechanisms.

Software Defined Networking (SDN) appears to be an appealing approach to implement and deploy innovative multicast routing algorithms in ISP networks [2–8] thanks to its logically centralized control plane. More specifically, in Software Defined ISP networks, live video streaming applications could benefit from QoS guaranteed dynamic multicast tree construction algorithms that exploit the global view of the network. Also, ISPs could exploit fine-grained control over QoS guaranteed multicast and best-effort traffic to implement traffic engineering policies that are friendly to low priority best-effort traffic. Several advanced multicast routing algorithms integrating load balancing techniques have been proposed in the literature to better utilize the network bandwidth, avoid traffic concentration and limit congestion in the network [5, 6, 9–12]. However most of these approaches require costly real time monitoring of link utilization in order to allow network resources sharing between the QoS guaranteed and best effort traffic classes according to ISP traffic management policies.

Moreover, SDN-based centralized architectures suffer from well-known scalability issues. Different approaches either based on distributed [13–15] and hierarchical [16–18] control planes or on stateful data planes [19, 20] have been proposed to address SDN scalability issues in general. Distributed controllers usually need costly state synchronization mechanisms. Therefore, only the approaches that propose delegation [16, 17] could be followed but they require to implement the whole functionalities of controllers at each router. Indeed, in the presence of large scale multicast applications, extra processing is required at routers to handle locally all Internet Group Management Protocol (IGMP) membership messages that would otherwise be flooded to the controller.

In this work, we address these two problems: (1) how to avoid implosion of IGMP group membership messages at the SDN controller and (2) how to deploy guaranteed-bandwidth multicast services in Software Defined ISP networks with low cost and while being friendly with best effort traffic.

To address the first problem, we propose to exploit the hierarchical structure of ISP networks and to use Network Function Virtualization (NFV). In a nutshell, we delegate when needed the multicast group membership management through specific network functions running at the edge of the network.

To answer the second problem, we propose a novel threshold-based load balancing algorithm in which a certain amount of link capacity in the ISP's infrastructure is reserved in priority for guaranteed-bandwidth traffic. This means that in absence of guaranteed-bandwidth traffic, best-effort can use this capacity. Hence, we dynamically increase the capacity share by gradually increasing the threshold. This approach is friendly to best-effort and helps in indirectly load-balancing the guaranteed-bandwidth traffic without the need of real-time link traffic monitoring mechanisms, as the controller is responsible of accepting or rejecting multicast subscription requests and is aware of bandwidth requirements.

Our contributions in this paper are the following: (1)

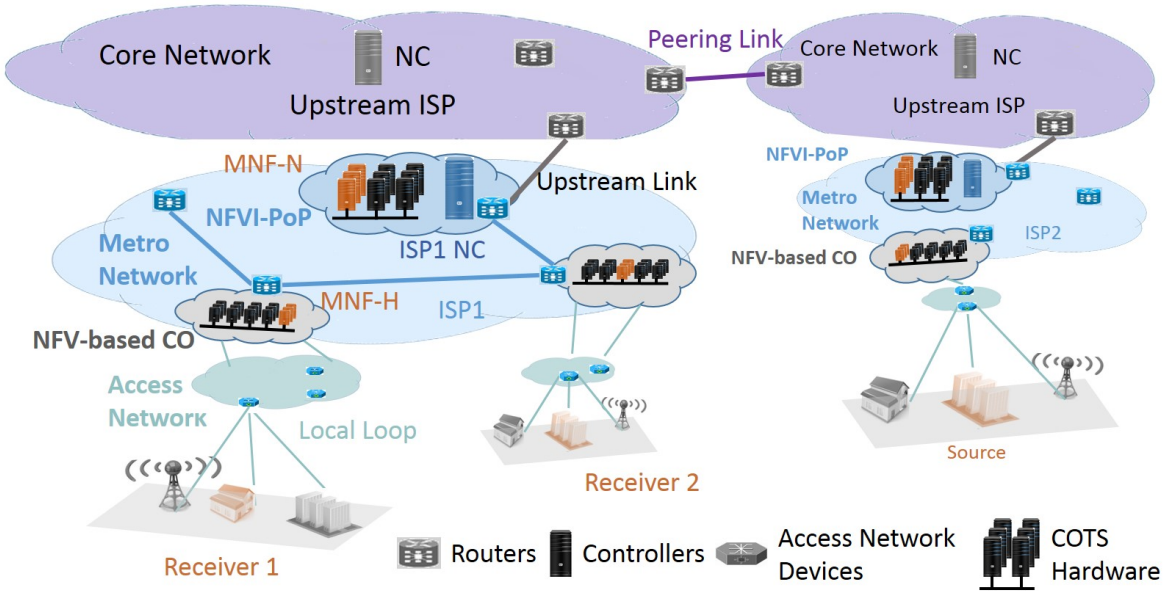


Fig. 1: Example of Software Defined ISP Network with NFVI-PoPs

an original solution to handle multicast group management in a scalable way on Software Defined ISPs with multicast network functions running locally on NFV Infrastructure Point of Presences (NFVI-PoPs) and NFV-based Central Offices (NFV-based COs); (2) a smart multicast routing algorithm called L2BM (Lazy Load-Balancing Multicast) for large scale live video streaming applications, which runs on the SDN controller and follows threshold-based traffic engineering policy for capacity sharing; (3) an implementation of the whole framework on Open vSwitches (OVS) [21] and Floodlight controllers and its evaluation, with comparisons with state-of-the-art solutions.

The rest of the paper is organized as follows: Section II presents our architectural approach for deploying scalable, flexible and hierarchical control, Section III presents L2BM, Section IV describes the implementation of our framework, Section V presents the evaluation of L2BM, Section VI discusses the related work and Section VII concludes the work.

II. Scalable Multicast Group Management for Software Defined ISP Networks

In this section, we tackle the problem of deploying multicast functionalities in a scalable and flexible way on Software Defined ISP networks (SDISPs). Traditional multicast routing and management protocols such as PIM-SM [22] and IGMP [23] effectively establish and maintain multicast communication paths between sources and receivers. More precisely, multicast routers deal with IGMP join/leave messages sent by the receivers to manage group membership state, and accordingly send PIM Join/Prune messages to the upstream routers to coordinate the multicast routing paths. Deploying multicast functionalities in SDN without taking precautions can lead to congestion issues at the controller as SDN routers need to forward all IGMP messages to the controller because they cannot take decisions autonomously and do not store group membership state information.

Let us consider hierarchical ISP networks as shown in Figure 1. With the advent of NFV, network aggregation points at central offices (COs) are being transformed into mini-datacenters. These NFV-based COs gather commercial-off-the-shelf (COTS) hardware that can run any network functions such as NATs, firewalls or caches [24]. The metro ISP network interconnects central offices, where customers' access lines are aggregated. Similarly, the core network interconnects gateway Central offices serving as Points of Presence and including NFV infrastructure that we call NFVI-PoPs. With SDN, a controller is responsible for programming packet forwarding in its own domain. In this paper, we refer to it as the Network Controller (NC) of a given domain.

In our approach, NCs delegate multicast group management functionalities to virtual network functions (VNFs) running at NFV Infrastructure at the edge of the metro networks. We call these functions MNFs for Multicast Network Functions and distinguish between MNFs-H that process IGMP Host membership traffic and run in NFV-based COs and MNFs-N that run in NFVI-PoPs and process PIM-like join/prune signals sent by MNFs-H to notify membership of the corresponding NFV-based CO. Unlike PIM join/prune messages, these PIM-like signals do not contribute in creating multicast tree using reverse-path forwarding.

We argue that delegating group membership management processing at NFV-based COs can greatly reduce the concentration of control traffic emerging from multicast applications. By doing so, our solution aims to achieve scalability similarly to traditional multicast protocols. It gives flexibility using NFV to enable multicast support on demand and does not put the burden of requiring multicast state management functionality on all the routers and especially core routers. NCs communicate with the NFV orchestrators that run on each NFV-based CO of the domain to instantiate MNFs when necessary. Note that NFV orchestrators are responsible for

scaling in/out their VNFs according to the group membership traffic load, providing flexibility. We emphasize that implementing the MNFs functionalities requires several features that are not compatible with hardware SDN routers, which are usually dumb devices. In particular, it is necessary to run a state machine for implementing IGMP and for generating periodically membership queries to the multicast receivers. As said earlier, we argue that the presence of mini data centers in central offices (NFV-based COs) as shown in Figure 1 will enable running the MNFs functionalities as VNFs. Even if such data centers are not deployed in central offices in the near future, MNFs could either be implemented as middleboxes running next to edge routers or integrated within software routers as switching at the edge is becoming virtual, handled on x86 cores as anticipated by SDNv2[1].

Let us now examine our proposed architecture with an example. At the start, in absence of MNFs running at the access NFV-based COs, the first group join request among the receiver hosts is forwarded as a packet-in to the metro NC. If the corresponding source or the multicast tree is already present in the metro network, then the metro NC establishes[2] the bandwidth guaranteed path for the requested multicast flow between the edge router that receives the join request at the access NFV-based CO and the multicast tree. At the same time, the metro NC interacts with the NFV orchestrator of the access NFV-based CO to instantiate an MNF-H and with its local NFV orchestrator at the NFVI-PoP to instantiate an MNF-N. After that, the group specific IGMP traffic received by the edge router is redirected to the MNF-H and handled locally. In addition, the PIM-like join/prune signaling traffic is redirected to the MNF-N that manages group membership of all the NFVI-based COs and communicates with the metro NC to update multicast tree for each group in the metro network. In case the access NFV-based CO is already receiving the requested multicast flow, the MNF-H is responsible of configuring the edge router to forward the multicast flow to the port where the IGMP join membership request has been received. Once the processing of IGMP messages are delegated to MNF-H, both the metro NC and MNF-H can configure the SDN edge routers. This design makes all the flow tables in the edge router vulnerable to unauthorized modification from the corresponding MNF. Hence, careful programming of MNFs is required to avoid race conditions on flow tables and maintain consistency in routers tables.

Metro NCs inform upper level NCs in the hierarchy of the presence of all the multicast sources in their domain and also exchange this information with peering ISPs' NCs. On detecting a multicast source, a NC communicates with the orchestrator on its local NFVI-PoP to instantiate a MNF-N if the latter is not yet running, in order to store information on the new multicast source and process future join/prune signals. If neither the source nor the multicast tree corresponding to the join signal belongs to the domain, the MNF-N sends the join message request to the upstream network through the upstream route set by the NC. If the source and the receivers are not in the same ISP, the join request will propagate through the peering link to reach the MNF-N corresponding to the source's ISP and a bandwidth guaranteed path will be established on both ISPs.

[1]See article "Time for an SDN Sequel? Scott Shenker Preaches SDN Version 2," www.sdxcentral.com/articles/news/scott-shenker-preaches-revised-sdn-sdnv2/2014/10/.

[2]The algorithm used to dynamically construct multicast trees with bandwidth guarantee is described in Section III.

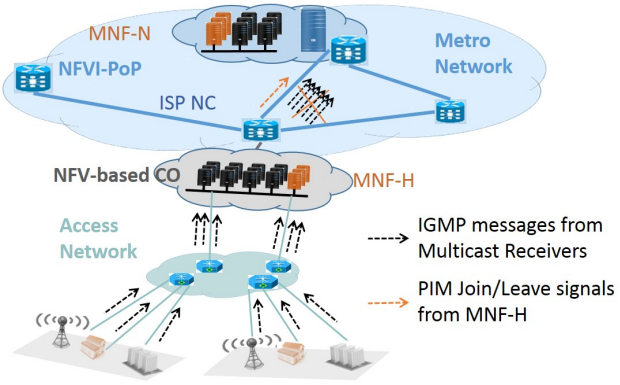


Fig. 2: Multicast Group Membership Management Analysis

MNF-Hs are responsible for aggregating the group membership reports received from their NFV-based CO networks, and according to the state machine, they can send join and prune signals to the MNF-N for the different multicast groups. Hence, similar to multicast routers in traditional multicast protocols, MNFs can maintain the group membership state of their downstream receiver hosts. Figure 2 illustrates our approach. Without the deployment of the MNF-H, the edge routers do not maintain multicast state and do not take decision to replicate the multicast stream to the required downstream interfaces in the centralized SDN based approach. Hence, all the multicast group membership messages are forwarded to the NC. In our proposed approach, once the MNF-H at the access NFVI-based CO receives a multicast stream, all the successive IGMP join messages received for the group at the edge router from the downstream access network are locally handled by the MNF-H. Hence, irrespective of the number of multicast group membership messages received for a group from end hosts in an access network, only the first IGMP join and the last IGMP leave messages result in sending a PIM join/leave signals from MNF-N to the metro NC in order to add/remove the NFVI-based CO from the multicast tree of the group. Therefore, with this mechanism NC is involved only for routing in core network and does not have to maintain IGMP state machines at any of the end hosts.

In Section IV, we describe an implementation of MNFs on top of Open vSwitches that we use to evaluate the whole framework in Section V.

III. Lazy Load-balancing Multicast routing Algorithm (L2BM)

In this section, we describe L2BM, a threshold-based load balancing routing algorithm proposed to deploy a guaranteed-bandwidth multicast service in ISP networks, that is friendly with best-effort traffic and without the need of real time link measurement mechanisms.

The main idea is to reserve a certain fraction of link capacity, referred as the *threshold*, for guaranteed-bandwidth multicast services and to restrict the corresponding traffic to

this threshold through traffic shaping. Then, to make sure that the best-effort traffic can use the reserved link capacity in the absence of guaranteed-bandwidth traffic, we use in forwarding devices Hierarchical Token Bucket [25], a classful queuing discipline which allows sharing the link capacity with different priorities. More precisely, we associate a threshold parameter to each multicast group join request received at NFV-based COs. While connecting the NFV-based CO to the multicast tree of the requested group, the L2BM algorithm avoids the links with utilization equal or greater than the current threshold value. L2BM attempts to reserve the required bandwidth on the minimum length reverse path from the receiver to any node in the multicast tree. If no reverse path to the tree can be found, L2BM increases the threshold value to consider previously avoided links and retries to attach the receiver to the corresponding multicast tree. In presence of multiple shortest paths length with equal threshold value, L2BM selects the one with the least maximum utilization for guaranteed traffic among its links. This information is available at no cost at the NC as it keeps track of previous requests.

Algorithm 1 shows the pseudo-code of L2BM for adding a new node in the multicast tree, using notations defined in Table I.

TABLE I: Notations used in Algorithm 1

Symbols	Definition
$\mathcal{E}$	set of edges
$\mathcal{V}$	set of nodes
$\mathcal{V}_\mathcal{T}$	set of nodes in multicast tree $\mathcal{T}$ of group M
e_{vu}	edge from v to u
B_{vu}	link bandwidth consumption of e_{vu}
C_{vu}	link capacity of e_{vu}
U_{vu}	link utilization of e_{vu}; $U_{vu} = B_{vu}/C_{vu}$
θ	threshold parameter
θ_{init}	Initial value of θ
θ_{max}	Maximum value of θ
r	new receiver for group M
b	bandwidth requirement of group M
$\mathcal{P}, \mathcal{Q}$	FIFO queues
$Path[v]$	set of edges constructing path from v to r
$len(v)$	path length from node v to r
$U_{max}(Path[v])$	Maximum link utilization among edges in $Path[v]$

Algorithm 1: Lazy Load-balancing Multicast

```
1  Function AddReceiverInTree(): Path
2      Q.Enqueue(r), len(r) = 0, visited ← ∅
3      return ThresholdBFS(Q, θ_init, visited)
4  Function ThresholdBFS(Q, θ, visited): Path
5      P : To store pruned nodes, θ_next = 1
6      while Q ≠ ∅ do
7          u ← Q.Dequeue()
8          if u ∈ V_T then
9              return Path[u]
10         end if
11         visited ← visited ∪ u
12         foreach e_vu and v not in visited do
13             U^new ← U_vu + b/C_vu
14             if U^new ≥ θ_max then
15                 continue
16             end if
17             if U^new ≤ θ then
18                 Path^new ← Path[u] ∪ e_vu
19                 len^new ← len(u) + 1
20                 if v ∈ Q then
21                     if len(v) = len^new and
                         U_max(Path[v]) > U_max(Path^new)
                         then
22                         Path[v] ← Path^new
23                     end if
24                 else
25                     Q.Enqueue(v)
26                     len(v) ← len^new
27                     Path[v] ← Path^new
28                 end if
29             else
30                 P.Enqueue(u)
31                 θ_next = min(θ_next, U^new)
32             end if
33         end foreach
34     end while
35     if P ≠ ∅ then
36         visited ← visited \ {v : ∀v ∈ P}
37         return ThresholdBFS(P, ⌈θ_next × 10⌉/10,
             visited)
38     end if
39     return NULL
```

Line 2 initializes the FIFO queue $\mathcal{Q}$ and path related variables. Then, the graph search starts with the initial value of the threshold (θ_{init}) to find a reverse-path from the new receiver, r, to any node u in the multicast tree of the requested group. Next, the algorithm recursively performs Breadth First Search (BFS) considering only edges utilized below current threshold θ and pruning the rest. In line 5, the $ThresholdBFS$ function initializes the prune queue $\mathcal{P}$ and the local parameter θ_{next}. Queue $\mathcal{P}$ stores the nodes for which any of their edge is pruned due to higher utilization than the threshold θ. In case none of the tree nodes is found, a recursive call of the function is done with queue $\mathcal{P}$ and θ set to θ_{next} to continue the search. The loop in line 6 starts the graph search from u, the head node of the queue. If node u is part of the tree for the requested multicast group, the algorithm terminates (lines 8-10). Line 11 adds node u in the $visited$ set. The algorithm expands the search by considering each incoming edge e_{vu} to node u (line 12). It computes U^{new}, the new link utilization of the edge e_{vu} by adding the bandwidth demand b (line 13). If U^{new} is higher than the maximum fraction authorized for guaranteed traffic it discards the edge (lines 14-16). Otherwise, it further checks U^{new} against threshold θ (line 17). If U^{new} is below θ, $Path^{new}$ and len^{new} are computed to reach v via u (lines 18,19). If another path of the same length to v already exists, the algorithm updates $Path[v]$ with the one having the lowest maximum edge utilization of the two (lines 20-23). Otherwise, node v is added in the queue $\mathcal{Q}$ and $Path$ and len are updated for v (lines 24-27). If U^{new} is above θ (lines 29-32), node u is added in prune queue $\mathcal{P}$, node u is removed from $visited$ set and θ_{next} is set to the minimum edge utilization value among all the pruned edges. If no tree node is found in the current search, the algorithm makes a recursive call to the function with prune queue $\mathcal{P}$, round up to tenth of θ_{next} and $visited$ set as parameters (lines 35-38).

When all the members from an access network leave a particular group, the MNF from the corresponding NFV-based

-CO has to notify the NC to remove its membership from the multicast tree. Node deletion from the tree is done by recursively removing the non-branch nodes in the reverse path of the stream till a branch node is encountered. This approach does not perturb the existing multicast tree, which prevents packet loss and reordering problems that could have emerged when restructuring the tree.

For each multicast join request, L2BM starts with an initial threshold value of θ_{init}. L2BM performs BFS using only edges with utilization below or equal the threshold. Nodes with higher link utilization are saved in prune queue $\mathcal{P}$ to continue, if needed, the search with increased threshold value through recursive calls. Let us consider the worst case scenario in which each call of $ThresholdBFS$ visits only one node and the rest of the nodes are enqueued in $\mathcal{P}$. This leads to at most 10 consecutive calls of $ThresholdBFS$ as the algorithm increases θ and rounds it up to tenth of the minimum of all link utilization operating above current θ, and each edge is visited exactly once. Hence, the order of run time cost of L2BM is the same as the one of BFS, $\mathcal{O}(|\mathcal{V}| + |\mathcal{E}|)$.

IV. Implementation

We have implemented[3] and evaluated our solution considering an ISP network with a two-level hierarchy, i.e., metro and access networks. To evaluate L2BM, we emulate guaranteed-bandwidth multicast routing experiments on OpenFlow 1.3 switches [26]. These software switches support traffic policing mechanisms such as rate-limiting for versions 1.3 and later, but such mechanisms throttle the bandwidth consumption by dropping packets. So, we opted for the traffic shaping functionality provided by the Linux traffic control system, which allows supporting bandwidth guarantee without loss in order to provide high QoE video [27] and we use the *ovs-vsctl* utility to configure queues in linux traffic control system for each guaranteed-bandwidth flow. We implemented L2BM in the Floodlight SDN controller and wrote a wrapper of *ovs-vsctl* for the controller that provides an API for handling queues on remote software switches. Regarding MNFs, we implemented them as software running within the same operating system environment as Open vSwitches to handle locally IGMP messages and PIM Join/Prune messages. Each access network is emulated as a host connected to a software router representing a metro edge router. Each MNF-H is connected to a software router through one of its data plane network interface to receive IGMP messages redirected by the NC. The latter uses the libfluid [28] OpenFlow driver to update the multicast group table of the software router. Then, MNF-N is implemented as a controller application running at a metro NC.

V. Evaluation

In this section, we study the performance of L2BM for routing guaranteed-bandwidth multicast flows in a single domain ISP. In this evaluation, we consider the scenario where the guaranteed-bandwidth traffic can use the whole link capacity of the network (i.e., $\theta_{max} = 1$). We assume that best effort traffic can use this reserved bandwidth in absence of high-priority traffic as routers implement the Hierarchical Token Bucket queuing discipline.

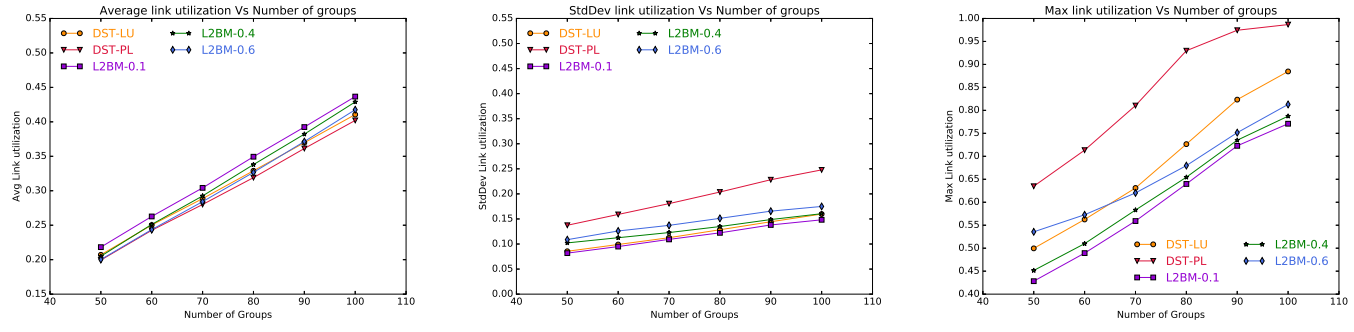
[3]Source code available at URL https://team.inria.fr/diana/software/l2bm/

Comparison Algorithms: We compare L2BM with two multicast routing algorithms that implement greedy heuristics of the Dynamic Steiner Tree (DST) algorithm with two different metrics: path-length (DST-PL) and link utilization (DST-LU) [29]. Both L2BM and DST-PL follow a *nearest node* approach with the path length metric proposed in [29], but in addition, L2BM tries to limit the maximum link utilization below some threshold. With DST-LU, new receivers join the existing multicast tree using the path with the minimum total link utilization. Three flavors of L2BM with different initial threshold values are used: θ_{init} = 0.1, 0.4 and 0.6.

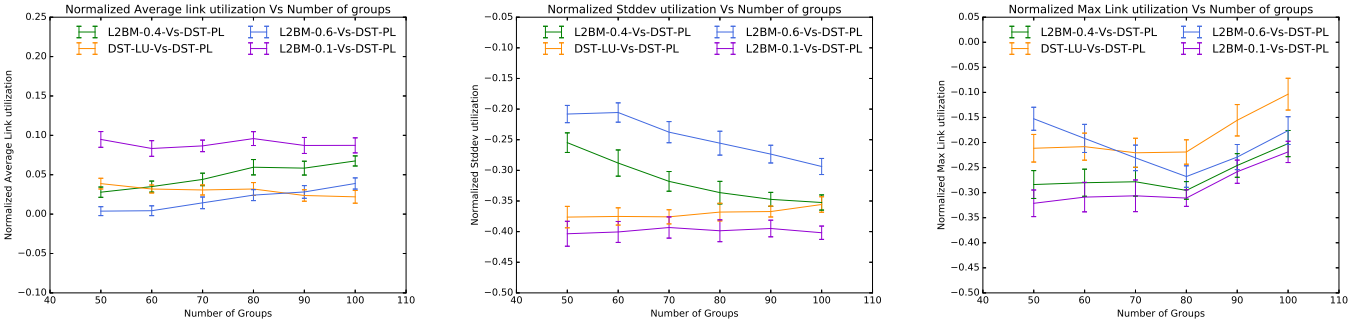
Testbed and Network Topology: Testing our bandwidth allocation implementation with Open vSwitches and Floodlight requires a testbed capable of emulating QoS-oriented SDN experiments. Existing emulation tools like mininet [30] do not consider physical resource constraints, hence the experiments can suffer from errors emerging from testbeds. We have chosen the DiG [31] tool to automate the procedure of building target network topologies while respecting the physical resources constraints available on the testbed. Regarding the network topology, we chose *INTERNET2-AL2S* [32] to represent an ISP network with 39 nodes and 51 bidirectional edges. Then we virtualized this topology using DiG on a grid network. DiG implements routers using Open vSwitches running with processing power of two computing cores for seamless packet switching at link rate. As the grid network uses 1Gbps links and *INTERNET2-AL2S* operates with 100Gbps links, we had to scale down the link capacity to 100Mbps in our experiments.

Guaranteed-Bandwidth Multicast Workload: To evaluate the algorithms with different traffic loads, we generate multicast groups with equal bandwidth demand of 2Mbps and the number of multicast groups is varied from 50 to 100. We associate 1 sender and 10 receivers per multicast group and the location of senders and receivers is randomly chosen across the set of nodes in the network. For every workload run, inter-arrival time of senders is exponentially distributed with mean of $3s$ and once the sender for a group is available in the system, receivers of the group are launched with an inter-instantiation time exponentially distributed with mean of $5s$. We make 20 workload runs for generating the same traffic load for the 5 multicast routing algorithms and allow fair comparison.

Evaluation Metrics: We use 3 different values of link utilization to evaluate the performance of the different multicast routing algorithms with the guaranteed-bandwidth multicast workload: 1/ Average (Avg) refers to the overall network bandwidth consumed, 2/ Standard Deviation (StdDev) estimates imbalance of traffic spread across the links and 3/ Maximum (Max) corresponds to the most congested link in the network. Hence, we use the 3 measures of link utilization to analyze the network bandwidth consumption and qualitatively estimate the impact of guaranteed-bandwidth on best-effort traffic for the different algorithms. A high Avg value means best-effort will have few overall network bandwidth available. StdDev shows uneven spread of available bandwidth across the network links, higher values of StdDev mean higher congestion possibility for best-effort. Max estimates domination of guaranteed-bandwidth traffic on critical links. We use the sample of network statistics collected after the instantiation of all the receivers of all the groups in the workload run. We compute the average of the metrics over 20 runs of the



(a) Average link utilization (b) Standard Deviation of link utilization (c) Maximum link utilization



(d) Normalized Average link utilization (e) Normalized Standard Deviation of link utilization (f) Normalized Max link utilization

Fig. 3: Performance comparison of L2BM with different θ_{init}, DST-PL and DST-LU

workload of a given number of multicast groups. We also compute normalized Avg, StdDev and Max link utilization obtained with reference to DST-PL, as shown in Equation 1 for Max, to compare the algorithms against DST-PL. The link utilization measures are normalized for each run of the workload and averages of them are computed with 90% of confidence intervals for performance comparison.

$$\overline{\text{DST-LU}_{Max}} = \frac{\text{DST-LU}_{Max} - \text{DST-PL}_{Max}}{\text{DST-PL}_{Max}} \quad (1)$$

Results and Analysis: As shown in Figure 3a, DST-PL gives marginally lower Avg link utilization, compared to L2BM and DST-LU algorithms, because it finds the shortest path from the new receiver of a group to the existing multicast tree without considering utilization of the links along the path. Hence, DST-PL results in higher congestion and imbalance of multicast traffic spread in the network compared to other approaches as evaluated using StdDev and Max metrics in Figures 3b, 3c respectively. L2BM is more aggressive than DST-LU in distributing the traffic. As a matter of fact, DST-LU decreases the sum of links' utilization along the path, which increases with path length. On the other hand, L2BM minimizes the maximum link utilization using the threshold mechanism. Hence, it uses longer paths till any node in the multicast tree is reachable with all the links operating below the threshold utilization. The normalized Avg graph (Figure 3d) shows that L2BM has 5% higher Avg link utilization compared to DST-LU, except for the higher value of $\theta_{init} = 0.6$. DST-LU and L2BM consumes 5% and 5 to 10% more bandwidth than DST-PL, respectively. However, both L2BM and DST-LU result in lower StdDev values compared to DST-PL as shown in Figure 3e, specifically for lower values of θ_{init}. For $\theta_{init} = 0.10$, L2BM is able to efficiently distribute traffic in the network even at low load. It decreases the Max link utilization by 20 to 30% and 10 to 15% when compared against DST-PL and DST-LU, respectively, depending on the load and on the value of θ_{init}, see Figure 3f. By controlling the link utilization with a threshold, L2BM can further decrease the Max link utilization than DST-LU. From the different values of θ_{init} L2BM is evaluated with, it can be observed that reserving 40% ($\theta_{init} = 0.4$) of link capacity and then gradually increasing it represents the most friendly way to spread the guaranteed-bandwidth traffic. Indeed, L2BM-0.6 spreads the traffic at high load, while L2BM-0.1 spreads the traffic at low load but consumes higher Avg link utilization with lower StdDev and Max link utilizations as visible in Figures 3. To sum up, although L2BM consumes marginally higher link bandwidth than DST-LU, it succeeds to fairly share the overall network capacity between the guaranteed-bandwidth multicast traffic and the best-effort flows.

VI. Related Work

Several approaches have been proposed to provide multicast solutions that leverage the logically centralized SDN control plane [2–5, 7, 8]. However, they do not address the specific scalability issue emerging from centralized processing of group membership messages at the controllers. In [6] a solution is proposed to prevent IGMP flooding in the network domain of the controller, but still, all the IGMP membership messages have to reach the controller and may overwhelm it

in presence of large number of receivers.

Recently, adding programmable control logic in the switches has been proposed to offload logically centralized controllers [19]. Such an approach is appealing to implement simple network functions but may result in a too high switch complexity to implement the full state machine of MNFs, which include timers. It is the reason why we have opted for an NFV-based MNF implementation. In [7], a similar network architecture than us is proposed, where the SDN controller is responsible for setting up the routing path and for NFV nodes to run video transcoders. However, they do not tackle the group membership scalability issue and their multicast routing algorithm targets a different objective than ours.

VII. Conclusion

In this paper we propose a novel threshold-based load balancing algorithm to deploy at low cost a guaranteed-bandwidth multicast service that nicely cohabits with best effort traffic. We plan to evaluate the L2BM algorithm with congested traffic scenarios to model flash-crowd and peak hours traffic in metro or backbone networks of ISP. By distributing the group membership management processing with NFs deployed at the edge of the infrastructure, we show that the NFV approach can be efficiently used to overcome the scalability issue of centralized SDN architectures. In future, we will explore how to achieve greater programmability for in-network state management and computation at the edge of ISP networks to realistically implement SDN architecture. The code and scripts used to evaluate our solution are made available to the community to ease reproduction of the experimental results[4].

Acknowledgements

This work has been partly supported by Inria and the Japanese Society for the Promotion of Science (JSPS) in the context of the UHD-on-5G associated team and by the French ANR under the "ANR-13-INFR-013" DISCO project on SDN. Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

References

[1] C. Diot et al. Deployment issues for the IP multicast service and architecture. *IEEE Network*, Jan 2000.

[2] Y. Kok-Kiong et al. Towards software-friendly networks. In *ACM APSys*, pages 49–54, New York, NY, USA, 2010.

[3] C.A.C. Marcondes et al. Castflow: Clean-slate multicast approach using in-advance path processing in programmable networks. In *IEEE ISCC*, July 2012.

[4] L. Bondan et al. Multiflow: Multicast clean-slate with anticipated route calculation on openflow programmable networks. *JACR*, 2(2):68–74, 2013.

[5] S. Tang et al. Realizing video streaming multicast over SDN networks. In *CHINACOM*, pages 90–95, Aug 2014.

[6] A. Craig et al. Load balancing for multicast traffic in SDN using real-time link cost modification. In *IEEE ICC*, pages 5789–5795, June 2015.

[7] S. Q. Zhang et al. Network Function Virtualization enabled multicast routing on SDN. In *IEEE ICC*, 2015.

[8] J. Ruckert et al. Flexible, Efficient, and Scalable Software-Defined OTT Multicast for ISP Environments with DynSDM. *IEEE TNSM*, 2016.

[9] M. Kodialam et al. Online multicast routing with bandwidth guarantees: a new approach using multicast network flow. *IEEE/ACM ToN*, 11(4):676–686, 2003.

[10] C. Debasish et al. A dynamic multicast routing satisfying multiple qos constraints. *International Journal of Network Management*, 13(5):321–335, 2003.

[11] Y. Seok et al. Explicit multicast routing algorithms for constrained traffic engineering. In *ISCC*, 2002.

[12] J. Crichigno et al. Multiobjective multicast routing algorithm for traffic engineering. In *ICCCN*, Oct 2004.

[13] Y. Soheil et al. Beehive: Simple Distributed Programming in Software-Defined Networks. In *ACM SOSR*, 2016.

[14] K. Phemius et al. DISCO: Distributed multi-domain SDN controllers. In *NOMS*, pages 1–4, May 2014.

[15] K. Teemu et al. Onix: A Distributed Control Platform for Large-scale Production Networks. In *ACM OSDI*, 2010.

[16] H. Yeganeh et al. Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications. In *ACM HotSDN*, 2012.

[17] M.A.S. Santos et al. Decentralizing SDN's Control Plane. In *IEEE LCN*, September 2014.

[18] Y. Fu et al. A hybrid hierarchical control plane for flow-based large-scale software-defined networks. *IEEE TNSM*, 12(2):117–131, June 2015.

[19] G. Bianchi et al. OpenState: Programming Platform-independent Stateful Openflow Applications Inside the Switch. *SIGCOMM CCR*, 44(2):44–51, April 2014.

[20] H. Song. Protocol-oblivious Forwarding: Unleash the Power of SDN Through a Future-proof Forwarding Plane. In *ACM HotSDN*, 2013.

[21] B. Pfaff et al. The Design and Implementation of Open vSwitch. In *ACM NSDI*, May 2015.

[22] H. Holbrook et al. PIM-SM: Protocol Specification (revised). RFC 7761, March 2016.

[23] B. Cain et al. Internet Group Management Protocol, Version 3. RFC 3376, October 2015.

[24] Network Functions Virtualisation (NFV) . http://portal.etsi.org/NFV/NFV_White_Paper3.pdf.

[25] HTB - Hierarchy Token Bucket, . https://linux.die.net/man/8/tc-htb. [Online; accessed 27-October-2016].

[26] N. McKeown et al. OpenFlow: Enabling Innovation in Campus Networks. *ACM CCR*, 38(2):69–74, 2008.

[27] A. Begen et al. Toward lossless video transport. *IEEE Internet Computing*, 15:48–57, 2011.

[28] libfluid. The ONF OpenFlow Driver. http://opennetworkingfoundation.github.io/libfluid/.

[29] B. M. Waxman. Routing of multipoint connections. *IEEE JSAC*, 6(9):1617–1622, Dec 1988.

[30] B. Lantz et al. A Network in a Laptop: Rapid Prototyping for Software-defined Networks. In *ACM Hotnets*, 2010.

[31] H. Soni et al. DiG: Data-centers in the Grid. In *IEEE NFV-SDN*, pages 4–6, Nov 2015.

[32] Internet2 AL2S Topology. https://noc.net.internet2.edu/i2network/advanced-layer-2-service/maps-documentation/al2s-topology.html.

[4] See URL https://team.inria.fr/diana/software/l2bm/