SVC Videoconferencing Call Adaptation and Bandwidth Usage in SDN Networks

Christelle Al Hasrouty, Cristian Olariu, Vincent Autefage, Damien Magoni, John Murphy

SVC Videoconferencing Call Adaptation and Bandwidth Usage in SDN Networks

Christelle Al Hasrouty*, Cristian Olariu†, Vincent Autefage*, Damien Magoni*, John Murphy†
* University of Bordeaux – LaBRI, France
† University College Dublin – School of Computer Science, Ireland
alhasrouty@labri.fr, cristian.olariu@ucd.ie, autefage@labri.fr, magoni@labri.fr, j.murphy@ucd.ie

***Abstract*—Videoconferencing is a convenient way for meeting with people while avoiding transportation costs. With cameras now being a commodity feature on every laptop and smartphone, videoconferencing can be delivered by many software applications.However, live video streams have strong latency requirements and consume much bandwidth. Furthermore, videoconferencing is not efficiently implemented in current applications as it often relies on a central server and does not leverage network layer services. With the advent of Software Defined Networking, it is now possible to use advanced techniques such as multicasting and stream layering inside the network for optimizing live video call transmission. In this paper, we investigate the impact of these techniques on the bandwidth and latency of Scalable Video Coding videoconference calls. We also evaluate the call capacity of software defined networks using such techniques.**

I. Introduction

Videoconferencing enables live video communication between three or more participants in different locations, each possibly equipped with different devices. The development of free applications and low cost devices has popularized the use of videoconferencing among both individuals and businesses. However, the widespread use of mobile devices implies wireless network constraints, such as bandwidth availability and latency which makes it harder to maintain a high Quality of Service (QoS) for all users.

Two general methods exist for connecting all participants together. In the first one, typically used in standalone conference systems, a central control device is used to connect multiple heterogeneous participants in a video-conference call. This centralized control device, usually called Multipoint Control Unit (MCU), acts as a bridge for these participants and adapts their channels. The MCU is located in a premise providing large and stable network bandwidth. Inside the MCU, a multipoint media processor is used to re-encode the audio/video stream in order to match the other participants' video requirements. This operation requires significant processing power. In the second method, each participant directly connects to every other participant with a point-to-point connection. Video adaptation is done in the devices of the participants.

To overcome the re-encoding problem, videoconferencing systems have started to use Scalable Video Coding [1] (SVC). SVC is a layer-based video compression technique which allows a video stream bitrate to be reduced by removing some layers of it without preventing the stream to be decoded. Thus, SVC avoids any re-encoding and streams can easily be degraded to limit bandwidth consumption. While SVC allows a better control over the stream bitrate, the layers' selection is only done at the endpoints or in the MCU and remains unchanged inside the core of the network. In order to adapt the streams inside the network, we have defined in this paper an algorithm leveraging SDN (Software Defined Networking) for building multicast distribution trees and dropping video layers at optimal locations. This algorithm is able to define where inside the network, some video layers should be dropped in order to adapt the streams to the bandwidth capacities of the receiving devices.

Section II presents a brief state of the art on videoconferencing and SDN technologies. Section III details the model and assumptions of our proposal and describes an example scenario. It also provides a thorough description of our algorithm for setting up video-conference calls. Finally, Section IV details the methodology and parameters used in our simulations and provides results related to the bandwidth usage and latency values of calls related to parameters such as network topology type, network size and number of participants. It also provides results concerning the call capacity of the network.

II. Related Work

In recent years, many efforts have been made on videoconferencing systems in order to improve the QoS for users while saving network resources. In order to cope with encoding/decoding complexity and network dynamics on video transmitted over a network, the ITU has defined the Annex G of the H.264 standard, called Scalable Video Coding [1] in 2007. SVC allows the bitrate adjustment of video streams without re-encoding and is currently implemented in many hardware devices.

Using IP multicast is a logical approach for optimizing IP-based multiparty calls but network operators have always been reluctant to deploy it. Only broadcast services, such as IPTV, have been partially supported by IP multicast on some ISP networks. Constructing multicast trees for multi-layered video has been studied as early as 1999 in [2], however, video layers could only be managed at the endpoints.

In 2000, Wang and Hou have classified multicast routing problems according to their optimization functions and performance constraints [3]. In their classification, our proposal targets the "link and tree constrained tree optimization" prob-

lem aka the "constrained Steiner tree" problem which has been proved to be NP-complete.

Zhao *et al.* have used a multicast approach over a SDN network [4]. In their proposal, SDN is used to centralize a full view of the network and to apply an efficient selection and distribution of video streams. Their solution achieves higher video stream bitrates with a slight increase on delays. Unfortunately, this study does not intend to maximize streams quality since all users receive video flows at the same bitrate. Moreover, their experimentations do not provide any results on global bandwidth consumption and only focuses on small-scale networks (under 150 nodes).

Laga *et al.* study the benefits of using a SDN/SVC based solution in order to optimize delivery paths of each SVC layer [5]. Their approach significantly reduces the amount of video freezes even in a congested network. Those results validate the benefit of SDN in the context of a video delivery system. Nevertheless, their work only focuses on a video delivery from a single source to a single destination. Furthermore, they only provide results on video freezes on a small-scale network (under 10 nodes).

A recent videoconferencing system combining a multicast distribution, a SDN network and SVC streams has been proposed [6]. Similarly to our proposal, the SDN controller is used to manage and adjust the distribution of video layers in order to both optimize the QoS of each participant and reduce the network usage compared to a classic MCU approach. Contrary to our study, their solution primarily focuses on reducing bandwidth consumption on core network links to the detriment of the flow quality received by the users. Besides, participants' capability relies on the screen size and not on the network access links properties. Finally, their evaluation has only been performed on a small-scale grid topology as opposed to our experiment which relies on realistic large-scale network topologies.

III. System Model and Algorithm

In this section, we describe the model of our system as well as the algorithm for setting up video-conference calls. To the best of our knowledge, no network operator deploys IP multicast protocols **for** video conferences. Thus, using a SDN network is the only way of duplicating and modifying the streams **inside** the network. Application-layer multicast can only be deployed at endpoints and is not currently leveraged by video conferencing applications.

A. Videoconferencing Model

When a video-conference is set between a number of users, these users are called participants. A participant is, at the same time, a sender of its video stream and a receiver of the video streams of all other participants. The participants are placed randomly in the network by being connected to any node of the network. Many participants can be connected to the same node. Participants are considered to be connected wirelessly to their node by a 4G cellular technology such as LTE. Each node of the network is supposed to provide a wireless access function (such as an eNode-B) and an SDN switching function. As the network is SDN-enabled, we assume that all nodes within the network contain OpenFlow switches and can communicate with an SDN controller which is responsible for the management of the calls. All switches are supposed to be able to adapt SVC streams by dropping unused layers on the paths indicated by the controller. In this work, we have considered that the number of participants in a call can vary between 3 to 12. For scenarios with larger values (above 6), we assume that participants have appropriate devices for conferencing (e.g., large tablets, laptops, mobile A/V systems). As discussed in the previous section, SVC streams are structured in layers, all built upon a base layer. In our case, we consider that the SVC layers consist of a base layer L1, and three enhanced layers L2, L3, and L4. In addition, a given layer can be used only if all the lower layers are also received. Each participant accepts the highest number of layers allowable by its downlink capacity. With only the base layer, the participant will receive the lowest video quality. If the participant's downlink capacity falls below what is required to receive the base layer, it can still remain in the conference if it can at least receive an audio-only stream of 32kbps. If any participant can not receive the audio stream, the conference call is rejected.

Four SVC profiles and one fallback audio-only profile are used in the following simulations. The indicated bitrates include network headers' overhead and audio streams (for SVC profiles):

- Audio-only, ~32kbps
- Layer 1: Scalable Constrained Baseline, Level 1, ~90kbps
- Layer 2: Scalable Baseline, Level 1.1, ~250kbps
- Layer 3: Scalable Constrained High, Level 1.2, ~0.5Mbps
- Layer 4: Scalable High, Level 1.3, ~1Mbps

The audio-only profile enables receivers with very low access link bandwidth to participate in the call.

Access links are wireless, which make them subject to variable communication channel conditions. This randomness in access bandwidth availability can lead to a large heterogeneity of bandwidth between the participants. Therefore, during a video call, each participant sends the maximum bitrate stream that can be consumed by at least one receiver, and this stream is then degraded (i.e., higher layers are dropped) to adapt to the participants with lower downlink bandwidth capacity. After discovering the network topology and channel conditions using SDN global view, it is significantly easier to identify the locations of the switches where it is necessary to degrade SVC streams.

Figure 1 shows a call scenario example with 6 participants. The D_i values represent the available downlink bandwidth for each access link of each participant. The uplink bandwidth of each participant is assumed to be higher than the highest quality video stream (i.e., above 1Mbps). The d_i values indicate the SVC stream bitrate received per participant. Thus, at any participant i, $5 \times d_i < D_i$. Each video stream harbors the color of its sender. Each arrow points to the direction of the video stream and is labeled by the number of layers

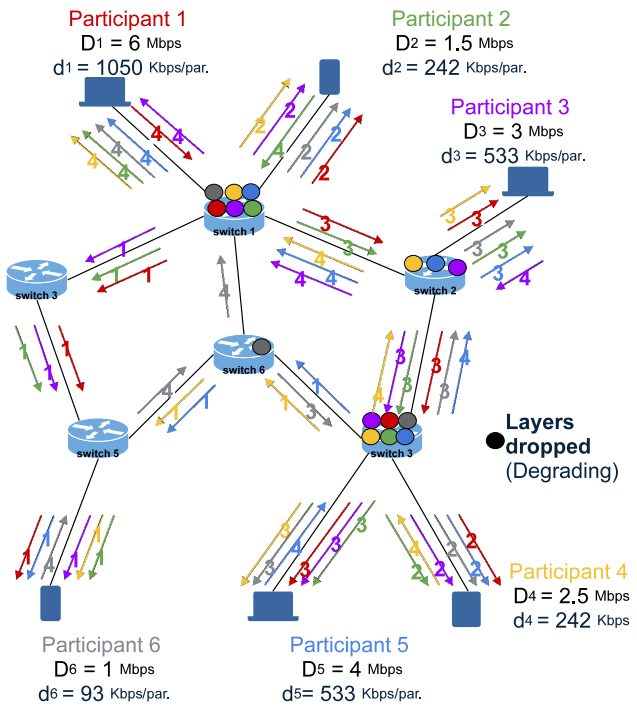

Fig. 1. Video-conference example

sent. For instance, an arrow labeled 3 represents a video stream containing layers 1 to 3. A stream can be degraded by removing layers at any switch but it can never be upgraded in quality inside a switch. It must be noted that our solution does not preclude the use of encrypted datagram connections (such as DTLS). By sending each layer on a different UDP destination port, the switches can drop the layers just by looking at the IP addresses and UDP port numbers, without the need to decrypt the DTLS payload.

B. Algorithm for Setting up a Videoconference Call

The algorithm, executed in the SDN controller for setting up a videoconferencing call, builds a multicast tree from each participant (sender) to all the others (receivers) and relies on several assumptions:

- For n participants, n source-rooted multicast trees will be built.
- For optimal synchronization, all SVC layers belonging to the video stream emitted by one sender, follow the same paths. There is not one tree per layer but one tree for all layers of a given video stream. Of course, some branches of that tree may carry only a subset of the layers.
- Access downlink bandwidth is chosen randomly for each participant from a range of plausible 4G data rates and is assumed to not vary over the duration of the call. However, they do vary for each new call/session. We plan for future work to execute our algorithm periodically to adapt the call to the variations of the access links' characteristics during a call.

TABLE I
ALGORITHM'S VARIABLES

Variable	Definition
B_k	Bitrate level $k \in [\![1, m]\!]$
S_i	Sender $i \in [\![1, p]\!]$
U_i	Uplink bandwidth of S_i
u_i	Uplink bitrate of S_i ($u_i = B_l$)
T_i	Tree rooted in S_i
$F_{i,k}$	Subtree(s) of T_i where links have bitrate k
R_j	Receiver $j \in [\![1, p]\!]$
D_j	Downlink bandwidth of R_j
d_j	Downlink bitrate of R_j
N	SDN Switch
P_j	Path from R_j to S_i or to T_i
SC	SDN Controller

- Core links are supposed to all have the same bandwidth **dedicated for this type of A/V traffic** (set between 100Mbps to 1Gbps depending on the scenario). This means that links may have different bandwidth capacities but they all reserve the same amount of bandwidth for the videoconferencing calls. We assume the SDN network to be WAN-sized (i.e., from 500 to 4000 nodes).
- The SDN controller knows the complete topology of the network, the available bandwidth on each core link, the position of the participants as well as their available uplink and downlink access bandwidth.
- Any SDN switch can degrade (i.e., drop higher quality layers) any SVC stream.

The optimization goal of our proposed scheme is to minimize the total bandwidth used by a call, while maximizing the bitrates received by each participant. Furthermore, our scheme is link constrained (i.e., the sum of the streams on **any** link can not exceed its bandwidth capacity), and tree constrained (i.e., the sum of the delays over **any** path can not exceed a given threshold). Thus, for the Wang and Hou classification [3], our proposed scheme targets the *link and tree constrained tree optimization* problem aka the *constrained Steiner tree* problem which is NP-complete. The algorithm 1 proposed for constructing the videoconference call is based on a single source/sender approach. It is a non optimal **heuristic** that builds the tree backwards from each receiver to a given sender and then reiterates the procedure for every sender. It is composed of four main steps:

1) Collecting the downlink bandwidth for each participant (line 1).
2) Computing the receiving and sending bitrates for each participant (lines 2-3).
3) Building a multicast tree from each participant to all the others if capacity permits (lines 4-25).
4) Checking the latencies over all paths between participants.

Before launching the algorithm `BuildVideoCall`, the network state is backed up by the controller. In step 1, the controller retrieves the downlink bandwidth value of each participant by polling its access switch (the one through

which the participant is connected). In step 2, the receiving bitrate of each participant is calculated by first dividing its downlink bandwidth by the number of participants excluding itself, and then by picking the highest bitrate level lower or equal to this computed value (line 2). This means that any given receiver will ask the same bitrate to all the senders (optimizing reception with different bitrates would increase the algorithm's complexity and is left for future work). The sending bitrate is then defined as the maximum received bitrate level found among all participants, noted B_l (line 3). As the uplink bandwidth is considered to be always higher than the highest bitrate level B_1, the sending bitrate B_l is thus the same for each participant. It may be lower than the highest quality bitrate B_1 if no receiver can receive B_1.

In step 3, each participant, considered as a sender, builds a multicast tree to all others (line 4). The bitrate levels are sorted from the highest B_1 to the lowest B_m (line 6). The receivers R_j are sorted by decreasing downlink bitrates and, at any given bitrate, by increasing distances (in hops) to the sender (line 7). For the highest bitrate B_l and the first receiver nearest to the sender, a shortest path is built to the sender (line 12). Then, for the other receivers at the same bitrate B_l, two modes are defined for connecting to the sender:

- **Minimizing Spanning Tree (MST)**: a shortest path is built up to the closest switch belonging to the tree, thus making a spanning tree which minimizes bandwidth usage in the network (line 16). It is a heuristic which does not necessarily provide a mininum spanning tree, hence the term *minimizing*.
- **Shortest Path Tree (SPT)**: a shortest path is built up to the sender, potentially stopping at the first switch belonging to the tree, thus making a shortest path tree which minimizes latency between participants (line 20).

If no path is found by function `BuildShortestPath`, due to saturated links, the call is rejected. The algorithm is stopped and the controller restores the network state previously backed up before starting the algorithm. If a path is found, the stream is then duplicated at the junction switch found by one of the above methods (line 24). After all the receivers having the highest bitrate B_l are connected to the sender, they form a tree $T_i = F_{i,l}$. The next highest receiving bitrate B_k is then chosen and receivers having this bitrate, again sorted by their closeness to the source, connect to the T_i tree by one of the two modes described earlier. In step 4, each path is checked against the maximum acceptable latency. If the latency is higher, the call is rejected (latencies between participants can be computed when building the trees). The variables used in this algorithm are defined in Table I.

Algorithm 1: Setup of the Videoconference Call

```
   BuildVideoCall() return Bool
1  SC collects all D_i
2  SC computes all d_i = max_{1≤k≤m} (B_k) | B_k ≤ D_i/(p−1)
3  SC computes all u_i = max_{1≤i≤p} (d_i) = B_l
4  foreach S_i, i ∈ [[1,p]] do
5      T_i ← {0}
6      foreach B_k, k ∈ [[l,m]] | ∀ 0 < k < m, B_k > B_{k+1} do
7          foreach R_j, j ∈ [[1,p]] | R_j ≠ S_i ∧ d_j = B_k
           ∧ dist(R_j, S_i) ≤ dist(R_j', S_i) ∀ j' > j do
8              P_j ← {∅}
9              N ← {∅}
10             if B_k = B_l then
11                 if j = 1 then
12                     P_j ← BuildShortestPath(R_j, S_i)
13                     if P_j = {∅} then
14                         return false
15                 else
16                     if mode = MST then
17                         P_j ← BuildShortestPath(R_j,
                           N ∈ T_i)
18                         if P_j = {∅} then
19                             return false
20                     else if mode = SPT then
21                         P_j ← BuildShortestPath(R_j, S_i)
                             stopping at N ∈ T_i
22                         if P_j = {∅} then
23                             return false
24                     Set N to duplicate B_l on P_j
25             else
26                 if mode = MST then
27                     P_j ← BuildShortestPath(R_j,
                       N ∈ F_{i,k'})
28                     if P_j = {∅} then
29                         return false
30                 else if mode = SPT then
31                     P_j ← BuildShortestPath(R_j, S_i) stopping
                         at N ∈ F_{i,k'}
32                     if P_j = {∅} then
33                         return false
34                 if k < k' then
35                     Set N to drop layers for degrading B_{k'} → B_k
                       on P_j
36                 Set N to duplicate B_k on P_j
37             T_i = T_i ∪ P_j
38 foreach S_i, i ∈ [[1,p]] do
39     foreach R_j, j ∈ [[1,p]] | R_j ≠ S_i do
40         if latency(S_i, R_j) > Max_latency then
41             return false
42 return true
```

C. Computational Complexity

In order to obtain the shortest paths needed by our algorithm, we need to compute the Dijkstra algorithm for all nodes in the network. The complexity of Dijkstra on a sparse graph of size n (i.e., the number of edges is in $O(n)$) is $O(n^2 \times \log n)$ with a Fibonacci heap implementation. If the graph is dense (i.e., the number of edges is in $O(n^2)$), the complexity becomes $O(n^3)$. These path computations can be done upfront and results can be stored in a multidimensional array.Obtaining a shortest path from this structure takes at most $O(n)$ steps. If we define p as the number of participants in the call and k the number of video layers, then the complexity of our algorithm is $O(p^2 \times k \times n)$. Given that both p and k are very small and independent of n, the complexity of our algorithm becomes $O(n)$. The final complexity is thus equal to

the one required by Dijkstra for computing all shortest paths.

IV. Evaluation

This section presents the evaluation of our videoconferencing call setup algorithm compared to a typical unicast approach. As said above, we assume the calls to be short so that network conditions are considered stable for the duration of the calls. We will deal with long calls, i.e., dynamic reconfiguration of the connections, in our future work. The results presented here have been obtained by simulations in order to study large network topologies. We have developed a static simulator in Python implementing our algorithm described earlier. The code is open-source[1] and can be easily ported to a Python SDN controller such as POX or Ryu. We have not used real SVC streams but have mimicked their bitrate requirements as indicated in section III-A.

A. Parameters

For evaluating our solution, we need to know the topologies of some network operators. As this information is difficult to obtain, we have generated synthetic maps of various sizes by using the Erdös-Rényi model (ER) [7], as well as our Magoni-Pansiot model (MP) [8] where the degree distribution follows a power law (as observed in the Internet and in large communication networks). This will enable us to assess the impact of the topology on our solution.

Calls are created by the two modes of our algorithm (SPT and MST) presented in Section III and by a unicast mode (typical of most applications), where each participant has unicast connections to each others (not going through a central server). In our simulations, the access downlink capacity between a participant and its access switch is randomly selected from 4 Mbps to 14Mbps and the uplink bandwidth is set to 1.5Mbps, which are conservative real-life values observed in current 4G LTE networks [9]. The core links are supposed to all have a dedicated and limited bandwidth for live A/V communications such as videoconferencing (the value is indicated in the captions). Access links delays are set to 30 ms (a typical average value as shown in [10]) and core link delays are set to 10 ms (as observed in [11]). The maximum latency authorized over any path is set to 250ms. Most of our results are obtained by instantiating each network topology type 100 times for every network size (i.e., thus generating 100 networks for each size), every time with a different random number generator seed. On every network and for each number of participants, we execute 100 randomly placed calls (i.e., for each call, the position of each participant is randomly picked). Thus, each point on the following plots is the average of 10000 values. For some results, this number has been lowered in order to reduce computation time. In this case, the number of runs is indicated in the captions.

B. Results

We present here our results that were obtained by our simulations carried out with the parameters described in the previous subsection. Each point in the plots below is the average of 10000 measured data points (100 calls per network and 100 networks per topology type). As the relative errors, measured at 95% confidence level, were always below 5% in figures 2 to 5, the error bars were not depicted. They are shown for all other figures.

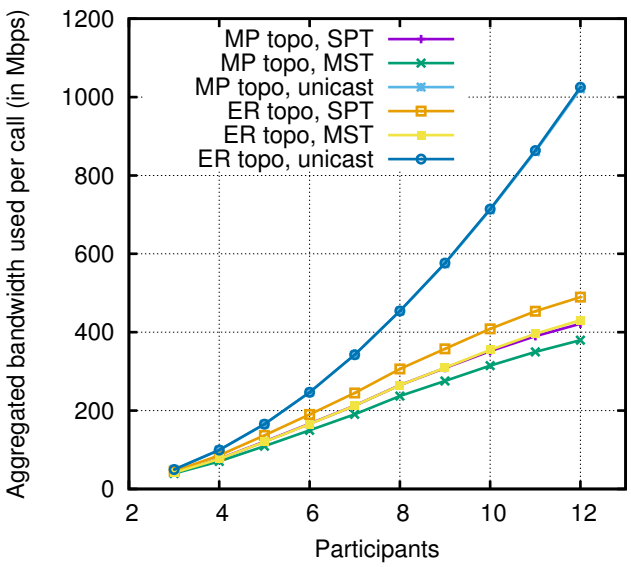

Fig. 2. Bandwidth consumption per call *vs* modes used and participants (2k-node network size, 1Gbps core links)

Figure 2 shows the impact of the number of participants on the total bandwidth used by a video call. The total bandwidth is defined as the sum of the call bitrates on every crossed link. While not meaningful in itself, this value is helpful for comparing the three approaches. As expected, for both topologies, the MST mode is the least bandwidth consuming, closely followed by the SPT mode. They both exhibit a sub-linear relationship between the number of participants and the bandwidth consumption. The unicast mode, on the other hand, exhibits a polynomial relationship and consumes much more bandwidth. This is due to the redundancy of the streams on the shared links. With 8 participants, each of the two multicast modes roughly consumes up to 70% of the bandwidth used by the unicast mode and at 12 participants, this ratio drops to 40%.

Figure 3 shows the influence of the network topology and size on the total bandwidth used by a call. The bandwidth used slowly increases with the network size because average distances, and thus paths taken by the video streams, do also marginally increase with the network size. From 500-node to 4k-node sizes, the largest increase is around 25% for unicast and 30% for SPT. The network size has a significant influence on the bandwidth used by a call, although smaller than the number of participants.

Figure 4 shows the impact of the number of participants on the maximum latency of a call. The maximum latency of a call is defined as the maximum latency measured over

[1] http://www.labri.fr/perso/magoni/cemeqacs

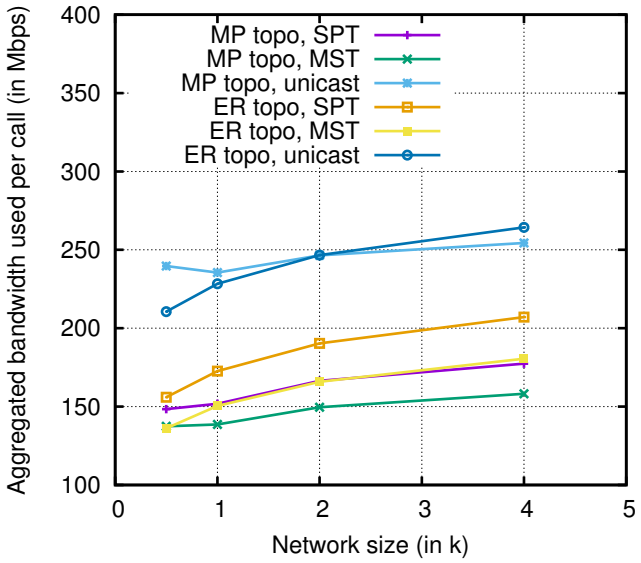

Fig. 3. Bandwidth consumption per call *vs* modes used and network type/size (6 participants, 1Gbps core links)

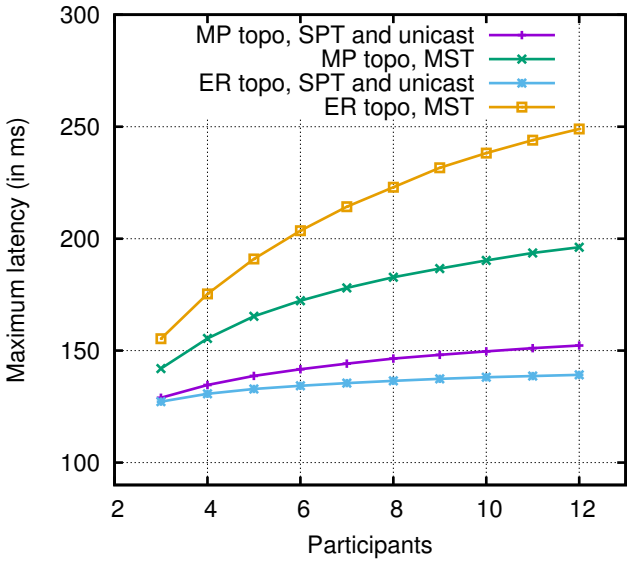

Fig. 4. Maximum path latency *vs* modes used and participants (2k-node network size, 1Gbps core links)

all paths between all participant pairs. The latency is exactly the same for the SPT and unicast modes, as expected, and increases marginally with the number of participants (e.g., +15% between 3 and 12 participants in MP topologies). The MST mode, however, has an important impact on the latency which increases with the number of participants (+60% between 3 and 12 participants in ER topologies). This is expected as the participants will optimize their paths to the tree but not to the source thus leading to longer paths. Latency values remain acceptable however, for live interactive communications, except for MST mode on ER topologies which results in latencies above the commonly accepted 250ms limit.

Figure 5 shows the influence of the network size on the maximum latency of a call. As with Figure 4, we observe that the network size has a bigger influence on the latency in the MST mode. This is expected as this mode is more likely to select longer alternate paths whose number increases when the network size increases. SPT and unicast modes are not much influenced by the network size, especially over MP topologies. This is also expected as MP topologies are power-law type graphs whose average distance and diameter does not increase when the network size increases.

Figure 6 shows the impact of the network size on the network call capacity. The network call capacity is determined as follows: calls are randomly generated (i.e., participants' position and access bandwith capacity) and added on the network one after the other. At some point, the `BuildVideoCall` function defined in Section III will return false, indicating that the call could not be setup because one or more links involved in the call were saturated (i.e., filled at the maximum of their bandwidth capacity). In this case, the call is rejected and the network call capacity is considered reached. TAs our algorithm

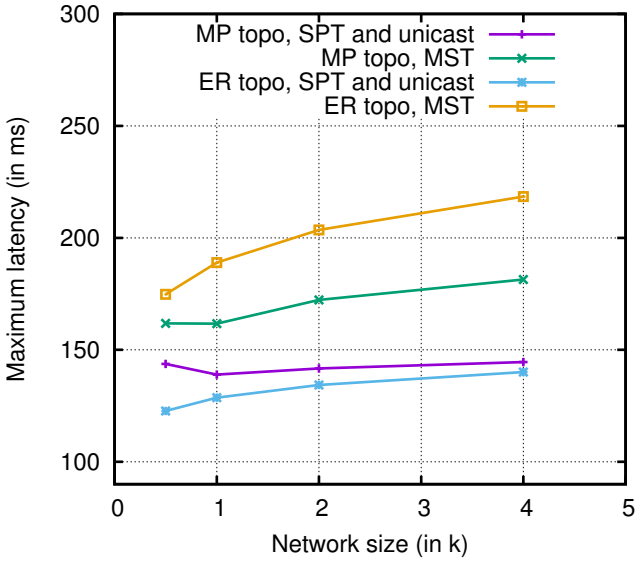

Fig. 5. Maximum path latency *vs* modes used and network type/size (6 participants, 1Gbps core link bandwidth)

is a non optimal heuristic, the number of supported calls thus depends on the sequence of construction of all the calls which are themselves randomly generated. For each experiment, the network capacity will thus slightly vary. As the process of calculating the call capacity is very time consuming, we have performed only 100 experiments on MP networks only (as MP topologies are more realistic than ER ones) with core link bandwidth set to 500Mbps (instead of 1Gbps). On this figure, as well as on Figure 7, points show the average capacity of the 100 experiments, while the error bars show the standard error of the mean. For any mode, the call capacity does not increase linearly with the network size but flattens when the

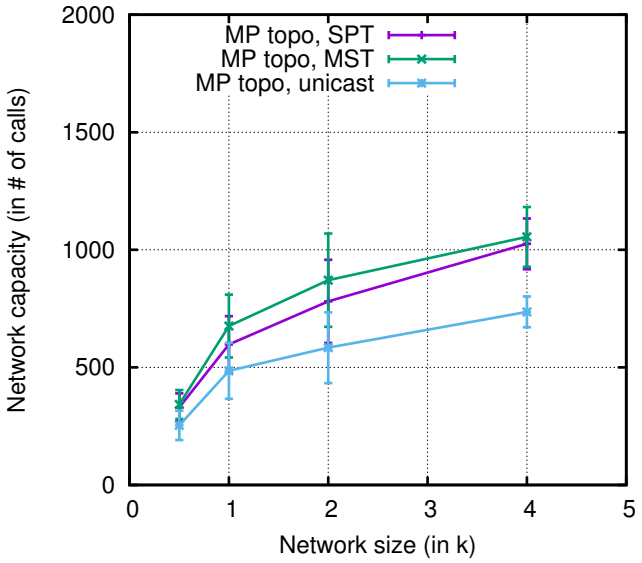

Fig. 6. Call capacity *vs* network size (6 participants, 500Mbps core links)

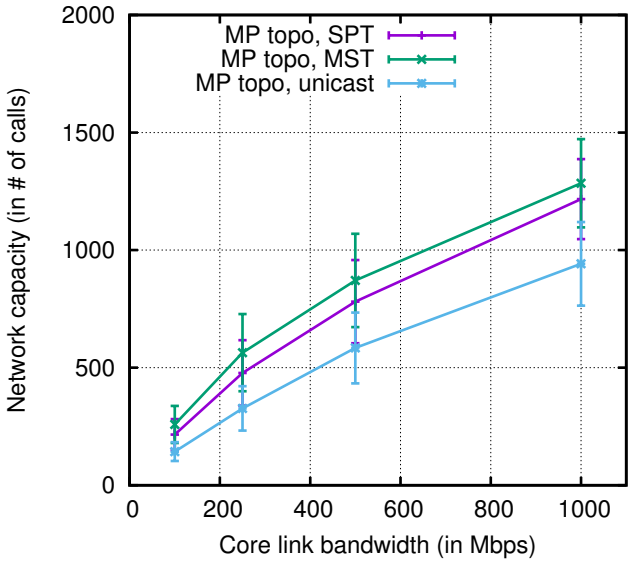

Fig. 7. Call capacity *vs* core link bandwidth (2k-node, 6 participants)

size increases. This is due to the fact that some links are much more used than others, given their strategic position in the network (especially in MP power-law type topologies), and they become bottlenecks. To leverage the network size, bigger bandwidth shall thus be used in those links if possible. For 4k-node networks, and assuming that each core link of the network has 500Mbps of dedicated bandwidth for A/V call traffic, we can see that our MST mode enables more than 1050 simultaneous calls while the standard unicast approach supports around 730 calls, translating in a 43% improvement. The gain is smaller for lower network sizes such as 500-node networks where a 35% is improvement is observed. The MST mode enables more simultaneous calls than the SPT mode but the gap is much smaller than compared to unicast.

V. Conclusion

Improving videoconferencing systems can lead to large bandwidth savings for network operators and can increase their ability to serve a larger number of customers. In this paper, we have proposed a videoconferencing system leveraging the SDN architecture. Our solution provides adaptive SVC layers to each participant by adapting them inside the network at the most appropriate place. In addition, it uses multicast for eliminating stream redundancy. Our results show that a videoconference call based on our MST mode uses only a fraction of the bandwidth of a unicast-based videoconference system (70% at 8 participants and 40% at 12 on a 2k-node network) while increasing the average path latency by 25% at 12 participants, but still remaining under 150 ms. Regarding the call capacity of the network, our results show an increase of 35% to 49% more calls (depending on the network size, at a given core link bandwidth) for our solution compared to a typical unicast approach with stream adaptation on the edge switches. Results also exhibit call capacity gains from 80% to 36% when varying the bandwidth of the core links in networks of a given size. Thus, our solution enables SDN-ready telco operators to significantly increase the call capacity of their networks. Our system is designed to dynamically adapt to time-evolving user access bandwidth, but we have not validated this aspect yet. Our future work will consist in analyzing our system over the duration of a call when time-varying access bandwidths are experienced, and evaluating a prototype implementation over a virtual network for re-assessing our current results obtained by simulations.

References

[1] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h. 264/avc standard," *IEEE Trans. on circuits and systems for video technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[2] M. D. de Amorim, O. C. Duarte, and G. Pujolle, "Single-loop packet merging for receiver-oriented multicast multi-layered video," in *International Conference in Computer and Communication*, 1999.

[3] B. Wang and J. C. Hou, "Multicast routing and its qos extension: problems, algorithms, and protocols," *IEEE Network*, vol. 14, no. 1, pp. 22–36, 2000.

[4] M. Zhao, B. Jia, M. Wu, H. Yu, and Y. Xu, "Software defined network-enabled multicast for multi-party video conferencing systems," in *IEEE International Conference on Communications*, 2014, pp. 1729–1735.

[5] S. Laga, T. Van Cleemput, F. Van Raemdonck, F. Vanhoutte, N. Bouten, M. Claeys, and F. De Turck, "Optimizing scalable video delivery through openflow layer-based routing," in *IEEE Network Operations and Management Symposium*, 2014.

[6] E.-z. Yang, L.-k. Zhang, Z. Yao, and J. Yang, "A video conferencing system based on sdn-enabled svc multicast," *Frontiers of Information Technology & Electronic Engineering*, vol. 17, no. 7, pp. 672–681, 2016.

[7] P. Erdös and A. Rényi, "On random graphs i." *Publicationes Mathematicae*, vol. 6, p. 290–297, 1959.

[8] D. Magoni and J.-J. Pansiot, "Internet topology modeler based on map sampling," in *Proceedings of the 7th IEEE Symposium on Computers and Communications*, 2002, pp. 1021–1027.

[9] V. Buenestado, J. M. R. Avilés, M. Toril, and A. Mendo, "Analysis of throughput performance statistics for benchmarking lte networks," *IEEE Communications Letters*, vol. 18, pp. 1607–1610, 2014.

[10] M. Laner, P. Svoboda, and M. Rupp, "Latency analysis of 3g network components," in *18th European Wireless Conference*, 2012, pp. 1–8.

[11] M. Hoerdt and D. Magoni, "Cartographie distribuée du coeur de l'internet," *Ann. des Télécom.*, vol. 60, no. 5-6, pp. 558–587, 2005.