A Control Architecture for Massive Adaptive Video Streaming Delivery

Giuseppe Cofano
Politecnico di Bari, Italy
giuseppe.cofano@poliba.it

Luca De Cicco
Politecnico di Bari, Italy
luca.decicco@poliba.it

Saverio Mascolo
Politecnico di Bari, Italy
saverio.mascolo@poliba.it

ABSTRACT

The optimization of massive video delivery, with the purpose of improving user engagement by delivering the maximum available quality of experience (QoE) to users, is a hot topic. In this paper we propose a Network Control Plane (NCP) for video streaming aimed at jointly maximizing users QoE and network utilization by reserving bandwidth on a per-flow basis. The NCP is placed on top of the controlled network and cooperates with distributed buffer-based adaptation techniques implemented at the client. It is specifically designed to take into account scalability and adaptivity issues. No communication with external entities involved in the delivery process, such as other networks or the streaming server, is required. We provide a reference implementation and a performance evaluation through simulations.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: [Network management]

Keywords

Adaptive Streaming; DASH; application-awareness; control plane

1. INTRODUCTION AND RELATED WORK

A great deal of research effort is recently being devoted to the optimization of massive video delivery, with the purpose of improving user engagement by delivering the maximum available quality of experience (QoE) to users. Even though an accepted definition of video QoE is still not available in the literature, three factors mainly impact QoE [5] [6]: *buffering ratio*, which is the ratio between playback interruptions and playback time, should be kept equal to zero; *average bitrate*, which corresponds to the encoding quality of the video, should be maximized to match the available bandwidth; *startup time,* which is the time elapsed from a user video request and the time instant the video playback starts, should be minimized. Modelling the relation of the user perceived QoE with the available network and computational resources is very complex, being it influenced by several factors, such as users sensitivity and video resolution, that are specific to the video flow. For this reason the research community is heading towards the study of application-aware resource management techniques, i.e. dynamic allocation techniques that are based on the perceived QoE.

In general, main causes of issues in the video delivery process are: 1) network bottlenecks due to poor planning or network congestion; 2) significant end-to-end bandwidth variations, which can cause buffering events; 3) and servers or Content Delivery Infrastructures (CDI) overload. QoE-aware resource management aims at optimally allocating network and computational resources to address such issues. At the same time, however, it should not be forgotten that the resource owner, i.e. the network or the CDN provider, is interested in increasing its revenues by avoiding resource underutilization. Thus, an optimal policy should jointly maximize QoE and resource utilization.

Many researchers have explored the possibility of using the emerging paradigm of Software Defined Networks (SDN) to implement application-aware resource allocation. SDN can be considered as a natural enabler for such applications, since it separates the network control plane from the forwarding data plane and provides a centralized view of the distributed state of the network.

An in-depth analysis of video delivery inefficiency issues is done in [5]. Authors make a case for a video control plane that can use a global view of client and network conditions to dynamically optimize the video delivery in order to provide a high quality viewing experience over an unreliable delivery infrastructure. A classification of the factors impairing user engagement and some techniques are given to improve performance. In [8] authors propose the virtualization of the ISPs access infrastructure using open APIs supported through SDN. Content providers programmatically provision capacity to user devices to ensure QoE by employing network resources slicing. Moreover, an algorithm is proposed for optimally allocating network resources, leveraging bulk transfer time elasticity and access path space diversity. In [4] an SDN-based application-aware bandwidth allocation approach is taken to maximize YouTube flows QoE. An OpenFlow testbed is set to evaluate performance when using several application-recognition techniques. In [3] an OpenFlow-assisted QoE Fairness Framework is proposed to fairly maximize the QoE of multiple competing clients in a Home Access Network.

VideoNext'14, December 2, 2014, Sydney, Australia.

http://dx.doi.org/10.1145/2676652.2676655.

In this work we propose a *Network Control Plane* (NCP) for video streaming to jointly maximize users QoE and network utilization by allocating bandwidth on a per-flow basis. To the purpose, we provide a control architecture and a reference implementation. The proposed control framework can be, at least conceptually, applied to any network involved in the delivery process (Home/Enterprise, Access, Content Delivery Networks) in order to maximize users QoE and network utilization in an adaptive and scalable manner in presence of external bottlenecks, overload or any other unknown source of traffic limitation. The proposed framework does not require any communication exchange of the NPC with external entities, such as the streaming servers or the other networks over which the flow is delivered. The NPC is placed on top of the controlled network and cooperates with distributed adaptation techniques implemented at the client. We point out that, even though SDN emerges as the natural enabler of the proposed framework, its adoption is not restricted to the employment of the SDN technology. The performance of the proposed framework has been evaluated by means of simulations.

2. CONTROL ARCHITECTURE

In this section, we describe the proposed control architecture. We consider a typical video delivery context, in which four main actors are involved: 1) the user, which runs a player to watch the video; 2) several intermediate communication networks (Home Network, Access Network, WAN, CDN), over which the video is delivered; 3) the streaming server, usually located in a CDN, which stores the video content and sends it to the user; 4) the origin server, which generates the video.

The *Network Control Plane* can be employed on top of any of the networks that are involved in the process and does not need any communication with the other involved entities. In order to achieve its optimization objectives, the control framework makes use of two cooperating techniques: *Network Bandwidth Reservation* and *Playout Buffer Control*. Before describing in detail the proposed control framework, we provide some background about: 1) the *Network Bandwidth Reservation* and *Playout Buffer Control* techniques and 2) some typical scenarios where the *Network Control Plane* might be used.

2.1 Background

Network Bandwidth Reservation (NBR), or *Slicing*, is performed at the network and provides, upon request, a reserved bandwidth slice to a single flow or to an aggregate of flows. Slicing can be static, i.e. done only once when the flow starts, or dynamic, i.e. adapted on-the-fly. A trivial strategy to always provide a flow with the maximum possible QoE is to statically assign a bandwidth slice equal to the maximum video bitrate. However, this leads to underutilization in presence of external traffic limitations, such as bottlenecks or CPU overload. Moreover, this trivial strategy cannot be applied to the case of limited resources, i.e. when the sum of the maximum bitrates of the active flows accessing a shared channel exceeds the available capacity (which often occurs unless the capacity has been overprovisioned). In this case several approaches can be used: admission control techniques, accepting or rejecting reservation requests based on the current reservations state, prioritization of some flows over others based on some static or dynamic sharing policies. These policies can in principle leverage any feature of the video flows, for instance: video resolution, users class, video popularity, video categories.

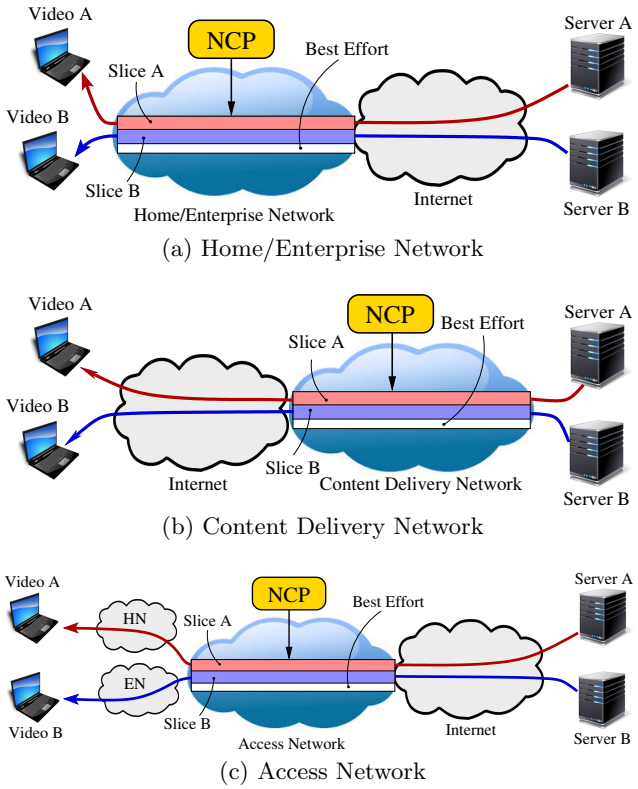


Figure 1: Use cases for the network control plane

Playout Buffer Control(PBC) is performed at the client by acting either on the sending rate (rate-actuated approach) or on the video bitrate (level-actuated approach). The rate-actuated is the mainstream approach, and employs an ON-OFF traffic pattern to adapt the sending rate to the selected video level. Typically, such controllers select the closest video level to the estimated available bandwidth. On the other hand, with the level-actuated approach the sending rate is not shaped and the video level is dynamically throttled to match, on average, the available bandwidth.

We argue that, even though application-aware bandwidth reservation might be applied in many of the networks domains involved in the video delivery, the typical scenario is the control of Access Network links since they typically represent the bottleneck [9].

Figure 1 shows different network domains where the proposed framework can be employed: (a) two video flows, which are streamed from two different servers and share the link over a Home/Enterprise Network; (b) two different servers belonging to the same Content Delivery Network delivering two flows; (c) two clients, accessing the Internet from either a Home Network (HN) or an Enterprise Network (EN), sharing the last mile of an Access Network.

2.2 The Control Architecture

The overall objective of the control architecture is to enable the joint maximization of the users QoE and the network utilization in an adaptive and scalable manner. This

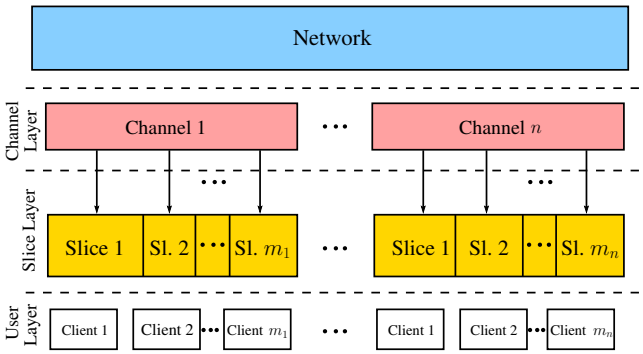


Figure 2: Control architecture

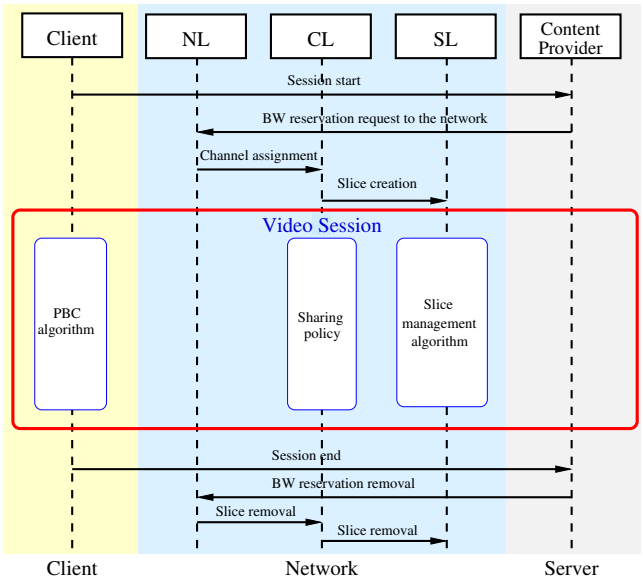


Figure 3: Flow diagram

goal is achieved by means of per-flow bandwidth reservation and playout buffer control.

In our framework the controlled network is modelled as a set of *channels* $\mathscr{C} = \{C_1, C_2, \ldots, C_n\}$. The i-th channel is characterized by a tuple (s_i, d_i, c_i) composed of a source node s_i, a destination node d_i and a given channel capacity c_i. For instance, an ingress or egress link of a Home Network can be considered a channel. This network model can be implemented by using well-known techniques such as network virtualization (see for instance [7]). In the following we make the assumption that a flow can be assigned to only one channel in the network, i.e. we do not consider multi-path delivery. In order to implement per-flow bandwidth reservation, each channel is dynamically divided in a set of *slices*, one for each video flow.

The proposed control architecture is composed of four layers, as shown in Figure 2: *Network Layer*(NL), *Channel Layer*(CL), *Slice Layer*(SL) and *User Layer* (UL). They are, respectively, responsible for: the network management, the single channel management, the single bandwidth slice management and the client-side playout buffer control. The three higher layers are placed at the network controller, the lower layer is placed at the client.

Let us begin by analyzing the flow diagram shown in Figure 3. It illustrates how a video session is set up, managed and terminated in the proposed control architecture. The NCP exposes a set of APIs to receive requests from the Content Provider to reserve bandwidth for a new video session. Identification information about the flow is communicated, for instance the video level set, the video resolution, and the user class. When the request is received, the NL can decide to either admit or reject based on its admission control policy. If the request is admitted, the NL assigns it to a channel. The corresponding instance of the CL creates a new slice which is managed by an instance of the SL. At this point, the video session starts. The sharing policy algorithm, the slice management algorithm and the client-side PBC algorithm cooperate to optimize the delivery as described in the following. When the client decides to terminate the session, it sends a message to the NL, which issues a slice removal request to the CL.

Let us now have a closer look at the functionalities provided by each layer:

1. The *Network Layer*(NL) manages the network. Upon a new video session request is received, the NL decides to either admit or reject the request. The NL assigns a channel to the admitted session. Thus, high level network control functionalities such as flow admission control and load balancing between channels have to be implemented here.

2. The *Channel Layer* (CL) manages the single channel according to the adopted *sharing policy*. To the purpose, it assigns to each slice a maximum bandwidth B_M according to the constraints imposed by the policy. This policy algorithm is executed asynchronously to the session arrival process, every T_M seconds.

3. The *Slice Layer* (SL) manages the single bandwidth slice. Its goal is to reserve a bandwidth slice B_R to the flow based on its average effective utilization. It is important to notice that the bandwidth slice B_R is not necessarily equal to the maximum bandwidth B_M that is sent by the CL. In fact, B_R has to belong to the interval $[0, B_M]$. The *SL* slicing algorithm is executed every T_R seconds with $T_R < T_M$.

4. The *User Layer* (UL) corresponds to the client-side adaptive video streaming algorithm and sets the video bitrate l based on the measured bandwidth and the video buffer length. This algorithm is typically executed when a chunk download is completed and it is decoupled from the NCP.

The underlying design choices are:

- the adoption of a hierarchical approach to reserve bandwidth. The proposed approach employs a centralized strategy cooperating with a distributed reactive strategy implemented, respectively, at the CL and at the SL. By leveraging the different control algorithms time scales of the two layers, scalability of the dynamical resource management policy can be achieved.

- decoupling of the client-side adaptive algorithm. This means that no information on the buffer status is required to be exchanged between the network and the client. In our view, in fact, this communication would

r	Video levels bitrate (kbps)	a_r	b_r	c_r
1080p	100, 200, 600, 1000, 2000, 4000, 6000, 8000	-3.035	-0.5061	1.022
720p	100, 200, 400, 600, 800, 1000, 1500, 2000	-4.85	-0.647	1.011
360p	100, 200, 400, 600, 800, 1000	-17.35	-1.048	0.9912

Table 1: Video levels and Utility Functions coefficients for each resolution $r \in \mathscr{R}$

make the system more complex without adding significant benefits.

- the adoption of a level-actuated strategy to control the buffer and select the video level. This choice is meant to prevent subtle interactions between the NBR and the PBC algorithms. With level-actuated algorithms the sending rate set by the TCP congestion control, in fact, will always match on average the bottleneck available bandwidth B. On the contrary, when rate-actuated algorithms are employed, the sending rate can be lower than B due to the ON-OFF phases. This can lead to slice underutilization, which would trigger a slice reduction at the SL.

We claim that the proposed architecture has the following advantages:

- no communication with external entities, except during the initial set up of the video flow, is required;
- scalability in presence of a large number of flows due to the employed hierarchical NBR strategy. This will be better clarified in Section 3 where a reference implementation of the proposed control framework is provided;
- adaptivity in bandwidth allocation at the SL in presence of external bottlenecks.

3. A REFERENCE IMPLEMENTATION

In this Section we provide a reference implementation of the proposed control architecture in order to show the feasibility of the proposed framework, which allows an incremental approach to design a video streaming delivery network. We consider the basic case of a single channel network.

3.1 The sharing policy

We employ the QoE Fairness (QF) policy proposed in [3] as the CL *sharing policy* (see Figure 3). The QF policy aims at providing a fair QoE share to all the video flows. The QF policy makes the assumption that each video is encoded at a single given video resolution $r \in \mathscr{R}$ with different bitrates. For each video resolution r, QF employs a nonlinear Utility Function $f_r(x)$ mapping the video bitrate x to the user perceived QoE. The employed Utility Functions $f_r(x)$ are two-term power series of the form $f_r(x) = a_r x^{b_r} + c_r$, which are based on objective video quality assessment models. The CL is assumed to know in advance the Utility Functions that

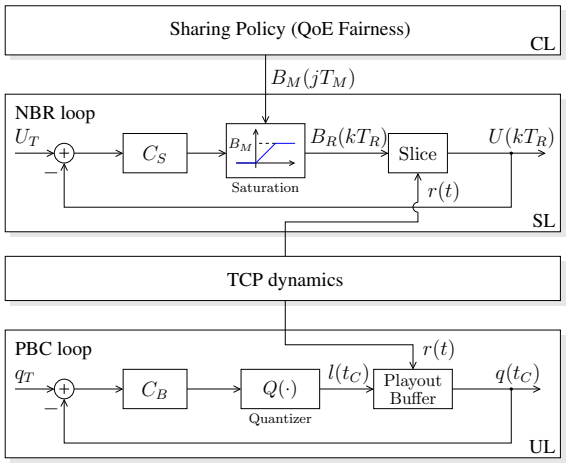


Figure 4: Reference Implementation

is associated to every resolution $r \in \mathscr{R}$[1]. The adopted levels and coefficients for each resolution are shown in Table 1. When a new video session is started, it communicates its video level set and video resolution r to the CL.

The fairness optimization problem to solve is the following:

$$Q_{fair} = \max(\min(f_r(x_i))) \quad s.t. \sum x_i < C \tag{1}$$

where x_i is the bandwidth reserved to the i-th flow and C is the portion of channel capacity reserved to video flows. x_i is then calculated for each flow as $x_i = f_r^{-1}(Q_{fair})$.

3.2 The proposed reference implementation

Figure 4 provides a block diagram of the proposed reference implementation showing how the control algorithms handle a video session. For simplicity, and without loss of generality, we do not include an admission control module at the NL. Three algorithms cooperate at different layers and on different time scales: the *Optimization Module* is executed by the CL every T_M seconds, the *NBR loop* is executed by the SL every T_R seconds, and the *PBC loop* is executed at the UL when a chunk is downloaded.

When a video requests arrives, the CL starts the video session and assigns it to a default slice, named *arrival slice*. Then, when the next iteration of the *Optimization Module* algorithm is executed, it creates a new channel slice and computes its size B_M by solving the optimization problem (1)[2]. From now on, the video session is assigned to the newly created channel slice of size B_M. Then, the CL communicates B_M to the *NBR loop*.

The goal of the *NBR loop* is to drive the *average slice utilization* U measured in the last sampling interval to the target U_T. Towards this end, the control algorithm C_S computes and reserves the bandwidth slice B_R. The rationale of the control algorithm is the following. Ideally, the band-

[1]This requires that a database storing Utility Functions for the resolutions is constructed.

[2]Actually, a preliminary check is done to address the case of unlimited resource. In fact, there is no need to solve (1) if the sum of the maximum bitrates of each video session is less than C.

```
1: B_R = B_M
2: while session_running do
3:     U = r/B_R;
4:     if U > U_T then
5:         B_R = B_R + incr
6:     else
7:         B_R = r/U_T
8:     end if
9:     if B_R > B_M then
10:        B_R = B_M
11:    end if
12:    sleep T_R
13: end while
```

Figure 5: *NBR loop* pseudocode

width slice B_R should be set equal to B_M. However, if an external bottleneck $R < B_M$ exists, such a setting would determine an undesired slice underutilization. Thus, in order to prevent slice underutilization, B_R should be set as:

$$B_R = \min(B_M, R). \quad (2)$$

However, the setting (2) cannot be implemented since R is time-varying and unknown. Hence, we employ the following adaptive algorithm. When the video session starts, B_R is set equal to B_M. If a slice underutilization is measured, i.e. $U < U_T$, an external bottleneck is present. Since the TCP rate r quickly matches the end-to-end available bandwidth, then the bottleneck bandwidth R is equal to r. On the other hand, when $U > U_T$ the reserved bandwidth slice B_R is lower than R but, in this case, it is not possible to estimate R. Thus, a blind probing strategy is required to iteratively increase B_R to match R. The saturation block shown in Figure 4 prevents B_R from exceeding B_M set by the sharing policy. Finally, it is worth noting that a strictly positive margin $1 - U_T$ is necessary to trigger the probing strategy.

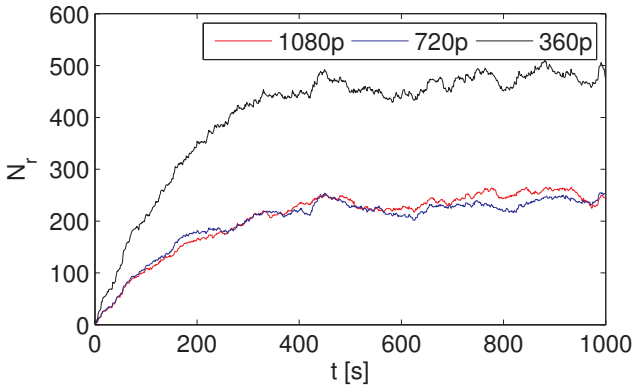
An implementation of the control strategy described above is given in Figure 5. An infinite loop is executed each T_R seconds. The blind probing phase that holds when $U > U_T$ is implemented in line 5: the reserved bandwidth is increased by the *incr* parameter at each sampling time. When $U < U_T$, B_R is set equal to r/U_T so that U matches U_T (line 7). The *saturation* block is implemented in lines 9-10. The unused bandwidth $B_M - B_R$ is assigned to a channel slice reserved for best effort flows.

To summarize, the channel is divided into three bandwidth slices groups: 1) the *arrival slice*; 2) the *video slices*, one for each video session, whose size B_R is computed by the proposed *NBR loop* (see Figure 5); 3) the *best effort* slice, whose size is the sum of the differences $B_M - B_R$ of all the active sessions.

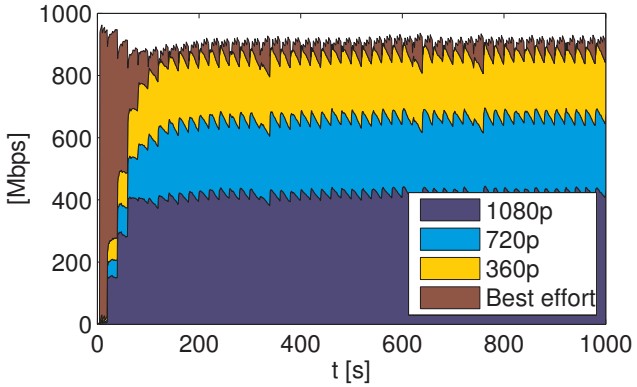
Finally, the *PBC loop* selects the video level l in order to steer the playout buffer to the target q_T (level-actuated approach). The adaptive algorithm Elastic [1] has been employed (the C_B block in Figure 4).

4. SIMULATIONS

In this section we evaluate the reference implementation of the proposed control framework by means of numerical simulations. In our evaluation scenario several video and best effort flows access a controlled channel of capacity 1000 Mbps. The video traffic is generated as follows. Video requests are generated according to an exponential distribution with a rate parameter $\lambda = 1.58$req/s. The session duration is distributed according to the CDF provided in [2]. Video resolutions are assigned to each flow according to the following proportion: 1/4 belongs to the 1080p class, 1/4 belongs to the 720p class and 2/4 belongs to the 360p class of Table 1.



(a) Number of active flows

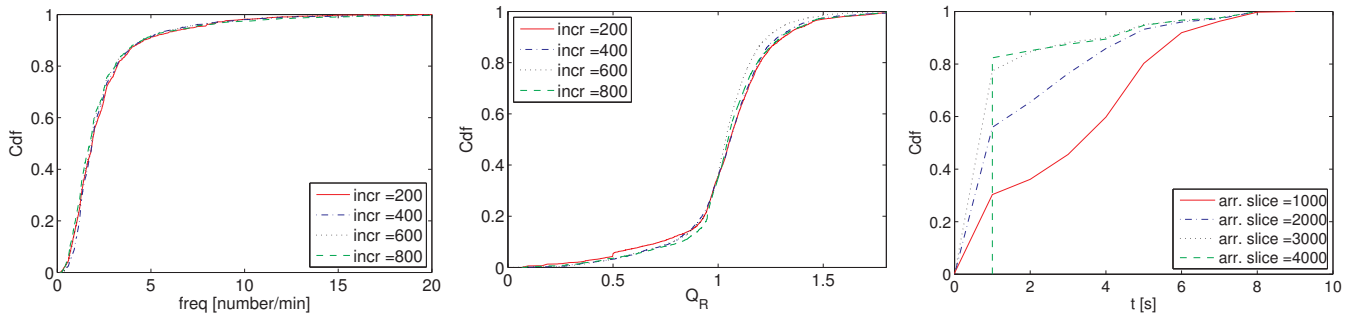


(b) Bandwidth slices reserved to video flows grouped by resolution and best effort slice

Figure 6: Channel performance metrics

In order to assess the adaptation control law shown in Figure 5, the clients are divided in three groups: for 80% of the clients, the bottleneck of the video session can only be due to the reserved slice of the controlled network; in fact, these clients are provided with a downstream bandwidth equal to 10Mbps that is larger than the maximum considered video level (see Table 1); 10% of the clients starts with a downstream bandwidth equal to 10Mbps and at a random instant the bandwidth suddenly decreases to a random value R; the remaining 10% starts with a downstream bandwidth of random value R and increases to 10Mbps. The random value R is generated according to a normal distribution with mean 1.5Mbps and standard deviation 1Mbps. Thus, for 20% of the video sessions the bottleneck is external to the controlled network. Finally, the sampling times of the *Optimization Module* and the *NBR loop* are, respectively, $T_M = 20$s and $T_R = 5$s.

In Figure 6 channel performance metrics are shown. Figure 6(a) shows the evolution of the number $N_r(k)$ of the active flows grouped by the video resolution $r \in \mathscr{R}$. In or-



(a) Switching frequency (b) Video quality ratio $Q_R = \tilde{l}/\tilde{B}_M$ (c) Startup delay

Figure 7: QoE metrics

der to assess that the Optimization Module control goals are met, we have computed the average QoE through the Utility Functions defined in Section 3. The average QoE are equal to 0.958, 0.951 and 0.944 respectively for the three resolution groups 360*p*, 720*p*, 1080*p*. Moreover, Figure 6(b) shows the bandwidth shares obtained by the three groups. As expected, higher resolution flows (violet area) get a larger bandwidth share, since they need higher bandwidth share *wrt* low resolution videos to get the same QoE. Moreover, we remark that the SL is able to provide a channel utilization higher than 0.9. In order to show that the *NBR Loop* algorithm does not significantly affect the performance provided by the UL, we measure the rebuffering ratio, the video level quality $Q_R = \tilde{l}/\tilde{B}_M$ and the video level switching frequency of the video sessions for different values of the *incr* parameter of the NBR Loop algorithm executed the SL.

We have measured a buffering ratio roughly equal to zero for all the flows regardless of the considered *incr* parameter. Figure 7(a) shows the CDF of the video levels switching frequency computed as the ratio between the number of level switches and the video session duration. The switching frequency is kept under 3 switches/minute for more than the 70% of flows, and it is not sensitive to the *incr* parameter. Figure 7(b) shows the CDF of the ratio between the average level $\tilde{l}$ achieved by a video session and the average bandwidth $\tilde{B}_M$ computed by the Optimization Module. $\tilde{B}_M$ represents the bandwidth share that achieves QoE fairness. Ideally, the ratio $\tilde{l}/\tilde{B}_M$ should be as close as possible to 1, which means that the video quality matches the reserved bandwidth. Only the 10% of the flows presents $\tilde{l}/\tilde{B}_M < 0.8$. The presence of $\tilde{l}/\tilde{B}_M > 1$ for some sessions can be explained with the presence of the playout buffer, which allows video sessions to achieve an average level higher than the reserved bandwidth in the initial phase. If the session is short enough, the level switch-down is never triggered, thus $\tilde{l}/\tilde{B}_M > 1$. Figure 7(c) shows the CDFs of the startup delay for different *arrival slice* sizes. As expected, the startup delay decreases as the *arrival slice* size is increased. Thus, the arrival slice size should be carefully designed based on the estimate of the video request arrival statistics.

5. CONCLUSIONS AND FUTURE WORK

In this paper a scalable network control framework to optimize massive video delivery has been proposed. A layered architecture is proposed, in which each layer is responsible for a specific control goal. As an illustrative example, an instantiation of this architecture is provided. Simulations show that the performance of the control algorithm of each layer is loosely coupled with the control algorithms employed at other layers. This is the key feature of the proposed architecture, which allows an incremental approach to design a video streaming delivery network.

6. ACKNOWLEDGEMENT

This work has been made possible in part by the gift CG #574954 from the Cisco University Research Program Fund, a corporate advised fund of Silicon Valley Community Foundation. This work has been also partially supported by the Italian Ministry of Education, Universities and Research (MIUR) through the MAIVISTO project (PAC02L1_00061). Any opinions, findings, conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect the views of the before mentioned funding agencies.

7. REFERENCES

[1] L. De Cicco et al. Elastic: a client-side controller for dynamic adaptive streaming over http (dash). In *Proc. of the Packet Video Workshop*, Dec. 2013.

[2] A. Finamore et al. Youtube everywhere: Impact of device and infrastructure synergies on user experience. In *Proc. of the ACM IMC*, pages 345–360, 2011.

[3] P. Georgopoulos et al. Towards network-wide qoe fairness using openflow-assisted adaptive video streaming. In *Proc. of the ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking*, pages 15–20, 2013.

[4] M. Jarschel et al. Sdn-based application-aware networking on the example of youtube video streaming. In *Proc. of the European Workshop on Software Defined Networks*, 2013.

[5] X. Liu et al. A case for a coordinated internet video control plane. In *Proc. of the ACM SIGCOMM*, pages 359–370, 2012.

[6] V. Sekar et al. Developing a predictive model of quality of experience for internet video. In *Proc. of the ACM SIGCOMM*, 2013.

[7] R. Sherwood et al. Flowvisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep*, 2009.

[8] V. Sivaraman et al. Virtualizing the access network via open apis. In *Proc. of the ACM CoNEXT*, pages 31–42, 2013.

[9] S. Srikanth et al. Broadband internet performance: A view from the gateway. In *Proc. of the ACM SIGCOMM*, pages 134–145, 2011.