Distributed Rate Allocation in Switch-Based Multiparty Videoconferencing System

STEFANO D'ARONCO, École polytechnique fédérale de Lausanne (EPFL)
SERGIO MENA, Cisco Systems
PASCAL FROSSARD, École polytechnique fédérale de Lausanne (EPFL)

Multiparty videoconferences, or more generally multiparty video calls, are gaining a lot of popularity as they offer a rich communication experience. These applications have, however, large requirements in terms of both network and computational resources and have to deal with sets of heterogenous clients. The multiparty videoconferencing systems are usually either based on expensive central nodes, called Multipoint Control Units (MCU), with transcoding capabilities, or on a peer-to-peer architecture where users cooperate to distribute more efficiently the different video streams. Whereas the first class of systems requires an expensive central hardware, the second one depends completely on the redistribution capacity of the users, which sometimes might neither provide sufficient bandwidth nor be reliable enough. In this work we propose an alternative solution where we use a central node to distribute the video streams, but at the same time we maintain the hardware complexity and the computational requirements of this node as low as possible, e.g. it has no video decoding capabilities. We formulate the rate allocation problem as an optimization problem that aims at maximizing the Quality of Service (QoS) of the videoconference. We propose two different distributed algorithms for solving the optimization problem: the first algorithm is able to find an approximate solution of the problem in a one-shot execution, whereas the second algorithm, based on Lagrangian relaxation, performs iterative updates of the optimization variables in order to gradually increase the value of the objective function. The two algorithms, though being disjointed, nicely complement each other. If executed in sequence, they allow to achieve both, a quick approximate rate reallocation in case of a sudden change of the system conditions, and a precise refinement of the variables which avoids problems caused by possible faulty approximate solutions. We have further implemented our solution in a network simulator where we show that our rate allocation algorithm is able to properly optimize users' QoS. We also illustrate the benefits of our solution in terms of network usage and overall utility when compared to a baseline heuristic method operating on the same system architecture.

CCS Concepts: • **Networks** → **Application layer protocols**; *Network resources allocation*; • **Information systems** → **Multimedia streaming**; • **Mathematics of computing** → *Mixed discrete-continuous optimization*;

Additional Key Words and Phrases: Videoconference; rate allocation; QoS optimization

ACM Reference format:
Stefano D'Aronco, Sergio Mena, and Pascal Frossard. 2017. Distributed Rate Allocation in Switch-Based Multiparty Videoconferencing System. 0, 0, Article 0 (2017), 23 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Nowadays video conferencing applications are getting more and more popular, and this trend is expected to continue in the next years according to Cisco Visual Networking index [3]. These applications allow several users to communicate together using audio/video streams and thus to

This work has been supported by the Swiss Commission for Technology and Innovationnder grant CTI-13175.1 PFES-ES and by the Swiss National Science Foundationnder grant CHISTERA FNS 20CH21 151569.
2017. XXXX-XXXX/2017/0-ART0 $15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

provide a rich communication experience. Ideally, all users aim at sending their own video data to all the other participants; at the same time they would like to receive the video data from all the other participants. When the number of participants becomes large, and the network resources are scarce, the transmission of the video data among all the endpoints might be extremely challenging. At this point, it becomes extremely important to optimize the rates of the video streams in order to provide a good Quality of Service (QoS) to the users while meeting the heterogenous bandwidth constraints imposed by the network.

The video distribution problem takes different forms depending on the network architecture that is used for the data transmission. Since multicast technology is not widely deployed in the Internet, the most naïve implementation of a videoconferencing system is the one where each user sends his own video flow directly to all the other users. If the number of participants is equal to N, then $N-1$ video flows share the download link of each endpoint and $N-1$ replicas of the source stream share the upload link. This architecture is particularly problematic in the case of asymmetrical connections, such as asymmetrical digital subscriber lines (ADSL), where the upload capacity rapidly becomes the main bottleneck for large values of N. An alternative solution consists in using a Multipoint Control Unit (MCU) with transcoding capabilities. The MCU is an endpoint used as a communication bridge by the videoconference participants. In this case every sender sends its video only to one node (the MCU) which mitigates the constraints on the upload link. The MCU, which is usually connected to the internet with some high bandwidth links, transcodes the video of each sender and forwards it to all the other participants [1], possibly in different versions. This solution, however, exhibits important scalability problems due to the huge computational load required by the MCU for the transcoding operations. In order to alleviate the scalability problem derived from a single MCU some works proposed to adopt a peer-to-peer solution, [6, 7, 17]. Instead of having a unique central node, the users' endpoints compose a peer-to-peer network in order to improve the video delivery capacity. Although this solution is extremely scalable it strongly relies on the upload bandwidths of the users' endpoints. In practice these may not provide sufficient bandwidth and induce large communication delays.

In this work we aim at designing a multiparty videoconferencing system that offers a tradeoff between a fully centralized solution with transcoding capabilities and a complete peer-to-peer solution that relies only on peers' resources. We focus on an architecture that keeps the computational requirements of the central node extremely low compared to a MCU with transcoding capabilities. Similarly to peer-to-peer systems the intelligence resides completely in the users' endpoints and the rate optimization process is fully distributed preserving system scalability.

In more details, we make the following key assumptions: *i*) the central node can enable application layer multicast communication, *ii*) the users' endpoints are capable of encoding their video streams at multiple rates (single rate encoding is included as trivial scenario), *iii*) a suitable transport protocol is available for real-time multimedia applications. The central node, also called the *switch node* in the remainder of the work, is used by the users as a communication hub. First, it offers a video packet forwarding service. In this way every sender can implement a 2-hop application layer multicast tree for the video delivery, alleviating the constraint on the upload link of the users. Second, it provides a coordination service among the senders and receivers in order to reach an effective QoS aware rate allocation. The complexity of the central node is kept as low as possible: the computation of the layer rates, as well as the forwarding policies of the video packets, are computed by the videoconference participants in a distributed manner. The central node is thus simply a sort of "application layer" switch, hence the name *switch node*. Having low complexity at the central node is not only important for preserving the scalability of the system in case of multiple parallel videoconference sessions. From the perspective of videoconferencing providers, a

low complexity central node requires less computation capabilities making the hardware cheaper and eventually decreasing the operative costs of the videoconferencing service.

More in detail, the operation of our system is the following. We first associate to every user a utility function, which depends on the activity (or importance) of the users and on the video characteristics. The utility functions and the upload/download bandwidth constraints are used to define an optimization problem that reflects the entire videoconference setup. We then propose two methods for solving this optimization problem. The first one provides a fast and efficient way to obtain an approximate solution. The second one is an iterative method that gradually improves an initial guess in order to achieve a higher objective value. Thanks to the structure of the problem, both methods can be implemented in a distributed way preserving the overall scalability of the system. The two proposed algorithms are actually executed in sequence every time the parameters of the original optimization problem change, e.g., when the relative importance of the users varies. In this way we can exploit the features of both methods: the fast algorithm modifies the rate allocation in a single step trying to reach quickly a good approximate solution, the iterative algorithm then refines the approximate solution in order to further improve the objective function.

We carefully evaluate the performance of our system in a network simulator (NS3) that replicates realistic network settings. We use the NADA congestion control [25] to send the media data streams and we perform the proposed rate allocation methods on top of it. Our experiments show that our system is able to properly allocate resources for optimizing the QoS of each user. The fast algorithm provides a quick good approximate solution to the rate allocation problem, while the iterative method provides a better solution at the price of slower convergence.

The remainder of the work is composed as follows. In Section 2 we discuss the related work on multiparty videoconferencing systems. The system model used in this work is described in Section 3. In Section 4 we introduce the optimization problem that we aim to solve, and we present the approximate and the iterative solution methods in Section 5. We discuss the algorithm implementation in Section 6. In Section 7 we provide some simulation results of the implemented system. Finally conclusions are given in Section 8.

2 RELATED WORK

Most of the existing works in the literature tackle the design of a multiparty videoconference problem based on peer-to-peer architectures. Compared to our system that relies on the existence of a central helper node, the peer-to-peer architectures have a higher number of decision variables in the rate allocation process. An example of such variables is the video stream route. Most of these works are based on the construction of a set of 2-hop multicast trees that employ user's nodes as central node to redistribute packets. In [6] the authors consider a peer-to-peer multiparty videoconferencing system and show that under specific assumptions, such as the peer uplinks being the only bottlenecks, the rate region achievable by using a limited number of mutualcast trees [14] is equal to the rate region achievable using inter-session network coding, making peer-to-peer solutions extremely attractive. In [7] the authors adopt a similar framework as the previous work but consider general topologies where the bottlenecks can be located anywhere in the network. In both studies, however, every user encodes his video in a single stream. This might not provide sufficient performance in the scenario where the users' download links have heterogenous values. The work in [17] focuses instead on a multi-rate scenario with upload link constraints only. In this work, every user is required to encode $N-1$ video layers in order to perfectly match the link constraints and use the network resources in an efficient way. Finally, a recent work [13] extends the previous framework to the case with both upload and download capacity constraints. The last two works however assume no constraint on the number of encoded video versions that a user can

generate. In reality, the computational load required to every client node for the encoding process might become too heavy when the number of participants to the videoconference grows large.

In our case we have a fixed central node that is used to redistribute the packets among the different participants, thus our problem is simpler than the peer-to-peer ones from this perspective. In addition, we do not have limitations on the location of the bottleneck links. However, differently from all the previous multi-rate works, we can further impose a specific constraint on the number of encoded streams that every user can generate, making our solution more applicable in scenarios with a large number of users.

The authors in [23] analyze how some of the commercial video conferencing solutions (specifically: Google+, Ichat and Skype) implement multiparty videoconferences. The analysis showed that both, the fully peer-to-peer, with a single encoded stream per sender, and the server based solution, with multiple encoded streams, are used by commercial solutions. Furthermore, another commercial solution [10] uses a simple central communication bridge in order to enable a 2-hop application layer multicast tree for the video delivery. The users encode the video with a finite set of rates and send them to a central node. The central node then decides which layers to forward to each receiver according to the download link bandwidth. Similarly to our system, this method offers a good compromise with a reliable central node to improve communication with no strong computational requirements for transcoding. The solution of the optimal rate allocation and the transmission policy are however not known nor available for all the analyzed commercial solutions. Finally, in [11] the authors develop a multiparty system with an architecture similar to the aforementioned solution and to our proposed system. Analogously to our work, this study is also motivated by the advantage and efficiency of having a simple central node with no transcoding capabilities that applies different forwarding policies to different flows. In order to limit the network usage, the central node forwards to the endpoints only a subset of the videoconference flows. The forwarding decision is made according to the users' importance level (based on the audio activity). However, as for the aforementioned commercial solutions, this study does not tackle the specific problem of the bitrate selection in the presence of heterogenous bandwidths among the videoconference participants. This is one of the gaps that we aim to fill in this paper.

3 SYSTEM MODEL

We target a scenario where N users participate in a videoconference. All the participants are both senders and receivers, thus the video of every user should ideally be received by all the other users. In the remainder of this work we use the terms user and participant interchangeably.

The proposed system architecture is a hub topology with the hub node corresponding to the switch node. Fig. 1 depicts a simple topology example. Every user establishes a bidirectional connection with the switch node using a general Congestion Control (CC) algorithm suitable for real-time communications. Multiple streams of video data are sent from a single user node to the switch node, and vice versa. In our solution, all streams from the switch to a user share a single download session. Another session is active on the reverse path for uploading the user's video streams. This way we can improve the responsiveness of the system when the rates of the video streams need to be modified. The idea of coupling several Real-time Transport Protocol (RTP) flows in order to gain more flexibility is similar to the one described in the internet draft [22] developed in the context of the RTP Media Congestion Avoidance Technique (RMCAT) working group [2]. Real-time CC algorithms maximize the sending rate and at the same time try to limit the end-to-end delay experienced by the user. We identify with $d_n(t)$ the transmitting rate achieved by the CC algorithm from the switch node to the endpoint of user n at time t. Similarly we denote with $u_n(t)$ the transmitting rate achieved by the CC algorithm from the user endpoint to the switch node

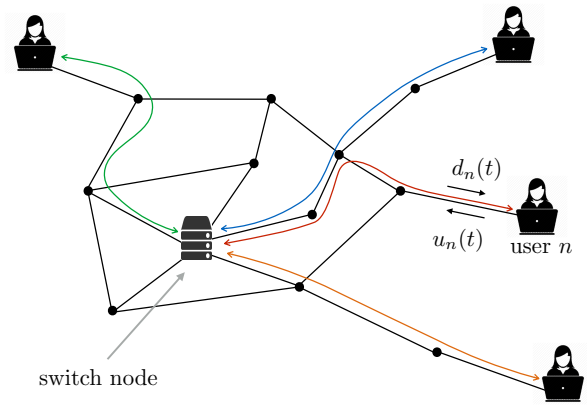


Fig. 1. System example.

at time t. As a result $d_n(t)$ and $u_n(t)$ represent the download and upload rate for the user n. The download and upload rates are obviously time-varying, however, in the remainder of this work we drop the time dependency in the notation to make it lighter.

We assume that every user is able to encode its video into one or multiple streams at different bitrates. Encoding a larger amount of streams is obviously more hardware demanding. In this work we let the maximum number of encoded streams be different among the videoconference participants, so that we can emulate the possible hardware heterogeneity of the endpoints. Although the proposed framework can easily be adapted to the case of Multiple Descriptor Coding (MDC) or multiple independent video streams, we consider the specific case of a Scalable Coding (SC), e.g., Scalable Video Coding (SVC) [21], which offers a compromise between adaptation to receiver bandwidth heterogeneity and overall resource requirements. In SC, a video is encoded using different layers, namely a base layer, and one or more enhancement layers. Users can increase video quality by stacking several enhancement layers on top of each other. The advantages of SC with respect to independent coding is illustrated by a simple example. Consider the case where one user wants to serve a video available at two rates, e.g., 0.5 Mbps and 1 Mbps. With SC the sender can encode the video progressively in two layers: one base layer of 0.5 Mbps; and an enhancement layer of 0.5 Mbps, the user will then send a total of 1 Mbps. Both layers are necessary at the receiver to decode the full quality stream. In the case of independent coding, the user needs to send a total of 1.5 Mbps, since the two streams are independent. It is easy to understand that if the total rate is an important constraint, SC enables a much more efficient bandwidth utilization. Finally, we indicate with L_m the maximum number of different SC layers that the user m can encode, and we denote with r_{ml} the rate required to decode the l-th layer of user m. Thus, r_{ml} is not the rate of the single l-th layer, but the cumulative rate of all the layers that are required to decode the layer l. The coding rate of layer l can thus be written as $r_{ml} - r_{m(l-1)}$ for $1 < l \leq L_m$ (with $r_{m0} = 0$).

Similarly to other works [7, 17], we treat the video streams of the users differently based on their content. In a videoconference, not all the video streams are equivalent: *i*) the video content of some users might be more complex and require a higher encoding bitrate than the one of other senders for the same quality, or *ii*) some users might be more active than others in the videoconference. When

the rate allocation of the users is properly computed, a larger rate should be reserved to the most demanding and high priority users in order to maximize the overall QoS of the videoconference.

In order to design our QoS aware rate allocation system, we need to define a mathematical model to measure the QoS of a user.

We define $U_m(r)$ as the utility of receiving the stream of user m at rate r. The utility function $U_m(r)$ is assumed to be a strictly concave increasing function of the rate, and embeds both the video complexity of the scene and the importance that the user m plays in the videoconference. Then, the total utility of the receiver n, modeled as a sum of the utilities of the different streams, can be written as:

$$\mathcal{U}_n = \sum_{m=1, n\neq m}^{N} U_m\left(r_m\right). \tag{1}$$

An intuitive choice for the utility function could be any sort of video quality metric multiplied by a scalar gain that reflects the user's importance in the videoconference. In this work we model the utility, or QoS, of receiving a video stream exclusively from its rate. In interactive communication however, the communication delay is also a critical quantity for the overall QoS of the communication and it should ideally be taken into account in the QoS metric. We however neglect this quantity in the present work, because our rate allocation is carried out on top of the real-time congestion control algorithm that is actually responsible for achieving a low communication delay. In our scenario we assume that each real-time congestion control session seeks for the best tradeoff between the overall transmitting rate and the experienced delay independently of the layer rate allocation. Therefore in this case the overlay rate allocation carried out by our algorithm has no effect on the experienced delay of the congestion control and we can ignore this value in our QoS model.

Finally note that there is usually a small distortion penalty with SC coding: the coding efficiency of the SC decreases with the number of encoded layers for a given total encoding rate. In order to model this effect the utility function should also depend on the layer number l, and the utility should ideally decrease when l increases. For the sake of simplicity we neglect this dependency in our model as the distortion penalty is negligible as long as the layers are large enough. It is however possible to include this behavior in the utility function for a more precise model if necessary. In the rest of the paper, we show how to properly set the coding rates and how to select video layers to maximize the global QoS.

4 PROBLEM FORMULATION

Given the settings described in the previous section, the rate allocation problem for maximizing the QoS in the videoconferencing system can be stated as the following optimization problem:

$$\underset{\{\mathbf{z}_n\},\{\mathbf{r}_m\}}{\text{maximize}} \quad \sum_{n=1}^{N}\sum_{m=1}^{N}\sum_{l=1}^{L_m} z_{nml}U_m(r_{ml}) \tag{2a}$$

$$\text{subject to} \quad \sum_{m=1}^{N}\sum_{l=1}^{L_m} z_{nml}r_{ml} \le d_n, \quad \forall n \tag{2b}$$

$$r_{ml} \le u_m, \quad \forall\, l, m \tag{2c}$$

$$\sum_{l=1}^{L_m} z_{nml} = 1 - \delta_{nm}, \quad \forall n, m \tag{2d}$$

$$r_{ml} \in [R_{\min}, R_{\text{MAX}}], \quad \forall m, l \tag{2e}$$

$$z_{nml} \in \{0, 1\}, \quad \forall n, m, l \tag{2f}$$

The above optimization problem aims at maximizing the overall QoS of the users by selecting the video layers that every user has to receive and by allocating the rates of the different layers that each sender has to encode. The variables of the optimization problem are *i*) z_{nml}, the decision variables for selecting which layer l of user m should be received by user n ($\mathbf{z}_n$ denotes a matrix of size $M \times \max\{L_m\}$ containing the layers selection of user n); and *ii*) r_{ml}, the rates of the different video layers ($\mathbf{r}_m$ denotes a vector of size L_m containing the layer rates of user m). The objective function of the optimization problem corresponds to the sum of the receivers' utility functions defined in Eq. (1).

The first set of constraints (2b) represents the download capacity constraints, which restrict the sum of the rates of the received layers to be no larger than the download capacity d_n. The second set of constraints (2c) defines the limit on the upload bandwidth of the users; practically the largest cumulative layer rate of user m has to be smaller than or equal to the upload capacity u_m. In (2d) we impose that every user gets exactly one version of every other source stream and no version of his video stream (δ_{nm} denotes the Kronecker delta). The next constraints define the limits on the values that the variables can take: constraints (2e) define some possible maximum and minimum encoding rates for each sender, while constraints (2f) limit the value of the decision variables to belong to the set $\{0, 1\}$.

Note firstly that in the case where independent video coding is used instead of SC, the constraints (2c) have to be changed. Since the encoding streams are independent in this case, the new constraints become $\sum_{l=1}^{L_m} r_{ml} \le u_m$. Finally note that the parameters of the above optimization problem, e.g., download/upload bandwidths and users' weights, are time-varying. The optimization Problem (2) needs to be reevaluated whenever parameters change. The quick computation of the new optimal allocation is therefore fundamental to guarantee a good QoS to the users in dynamic conditions.

The above problem represents a Non-convex Mixed-Integer Nonlinear Programming (MINLP) problem. Non-convex MINLPs are generally $\mathcal{NP}$-hard [5] and therefore it is extremely difficult to find the global optimal solution. We therefore focus in the next section on designing fully distributed algorithms in order to find good suboptimal solutions of Problem (2) and at the same time preserve the scalability of the system.

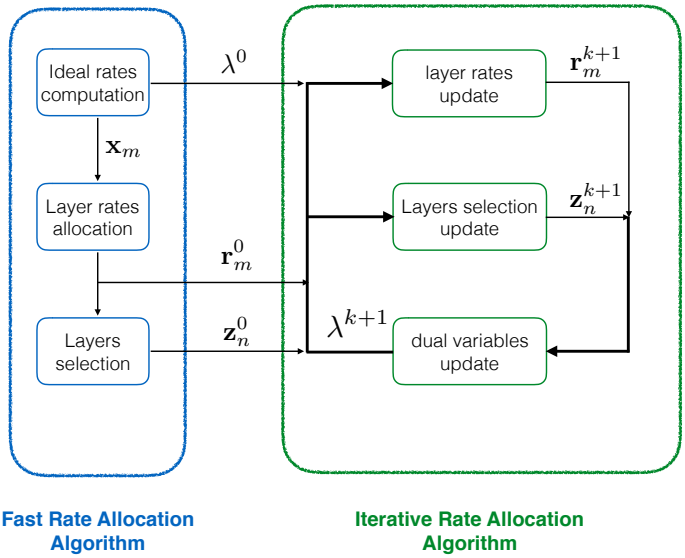


Fig. 2. Steps of the proposed rate allocation algorithm.

5 PROPOSED SOLVING METHODS

We describe now in detail the two methods that we use for finding a suboptimal solution to the problem. The first method provides a fast and approximate solution and is based on decomposing the original problem into three easier subproblems. The second method is an iterative method based on a Lagrangian relaxation: the coupled download capacity constraint is plugged as a penalty to the objective function, allowing then for a distributed update of the optimization variables.

5.1 Fast Rate Allocation Algorithm

The intuition behind the first method is the following. When the available layer rates of the senders are fixed, every receiver can easily and independently compute the best combination of the layers to maximize its utility function. This layer selection problem is actually the discrete version of the one where receivers are free to choose the rate of every sender as a continuous variable. By asking the receiver to solve the continuous version we can collect a list of *ideal rates* that the receiver wants to get; using these ideal rates, the senders can then prepare a list of available rates in order to best satisfy the different receivers. Finally receivers can select from the available rates computed by the senders, the ones that maximize their utility.

The original rate allocation problem is therefore divided into three subproblems as shown in Fig. 2. We now briefly describe the individual steps of this method, for a more detailed description we refer the reader to our former paper [9].

5.1.1 Ideal Rate Computation. The ideal rates of user n, denoted by $\mathbf{x}_n$, can be computed using the download bandwidth and the importance of the different participants. In mathematical terms,

the ideal rates of user n are the solution to the following optimization problem:

$$\mathbf{x}_n = \underset{\mathbf{x}'_n}{\text{argmax}} \quad \sum_{\substack{m=1 \\ n \neq m}}^{N} U_m(x'_{mn}) \tag{3a}$$

$$\text{subject to} \quad \sum_{\substack{m=1 \\ n \neq m}}^{N} x'_{nm} \leq d_n. \tag{3b}$$

Since we assume concave and non-decreasing utility functions the solution to this optimization problem is unique and can be easily computed using basic convex optimization algorithms [4]. The objective function of this problem corresponds to the receiver utility function as defined in (1), while the constraint simply imposes that the sum of the rates is smaller than the downloading rate set by the congestion control of user n. This optimization problem is solved independently by every receiver. By merging the results of Problem (3) for all the N receivers we obtain for each sender a set of $N-1$ ideal rates that can be used as guideline to decide the layer rates to make available.

5.1.2 Layer Rates Allocation Problem. This subproblem is the most complex to solve among the three steps composing the fast rate allocation method. It corresponds to the rate allocation of the different video layers at each sender based on the ideal rates collected from the receivers. The method is strongly based on the multicast rate allocation algorithms described in [15, 24]. The intuition behind this method is the following: if the receiver n requests an ideal rate x_{nm} from sender m, then the associated receiver utility is maximized if the sender m encodes a video layer with a cumulative rate exactly equal to x_{nm}. Considering an ideal scenario where all the receiver's ideal rates are available, any sort of deviation from the ideal rates will surely cause the receiver utility to decrease. The senders can follow this intuition in order to compute an approximate solution of the layer rates.

We first need to introduce a second concept of utility, namely the layer utility, written as:

$$U_{\text{layer}}(x_{nm}, r_{ml}) = \begin{cases} x_{nm}/r_{ml} & : x_{nm} < r_{ml} \\ r_{ml}/x_{nm} & : x_{nm} \geq r_{ml} \end{cases} \tag{4}$$

This function attempts to model the loss of utility when a receiver n with ideal rate x_{nm} receives instead a video layer with cumulative rate equal to r_{ml}. The choice of the utility function in Eq. (4) is not unique. It is however important that *i*) its maximum value is achieved when $x_{nm} = r_{ml}$, *ii*) it is non-decreasing for $0 \leq r_{ml} \leq x_{nm}$ and *iii*) non-increasing for $x_{nm} \leq r_{ml}$. Following the above intuition we can assume that, when multiple encoded rates are available, every receiver will select the one that is closest to its ideal rate. We can therefore define the overall layer allocation utility as:

$$\mathcal{U}_{\text{layer}}(\{x_{nm}\}, \mathbf{r}_m) = \sum_{\substack{n=1, \\ n \neq m}}^{N} \max_{r_{ml}} U_{\text{layer}}(x_{nm}, r_{ml}) \tag{5}$$

where $\mathbf{r}_m$ denotes a vector containing the L_m layer rates of sender m. Note that the higher the value of $\mathcal{U}_{\text{layer}}()$ the better the layer rates fit the ideal rates received.

Now that we have introduced the overall sender layer allocation utility, we can use this function to select the values of r_{ml}. In particular we need to select the values of r_{ml} that maximize $\mathcal{U}_{\text{layer}}()$. We provide a brief intuition about how it is possible to solve this problem by using a dynamic programming approach. Consider the case where all the layers up to $l-1$ are fixed, and we want to add layer l on top of them. Due to the shape of U_{layer}, the only receivers that might be interested in switching to this new layer are the ones that are using the second highest layer $l-1$. As a result

all the receivers that are interested in layers lower than $l-1$ will not be affected by the rate of layer l. Therefore, the rate allocation problem between layer l and all the rates smaller than $l-1$ is decoupled. Following this intuition, it is possible to build a dynamic programming approach to find the optimal solution layer after layer. For a more detailed description of the algorithm we refer the reader to our previous work [9].

We now briefly discuss the performance of the above method that selects the best set of video layers. In order to measure the overall layer utility we should know which layer is going to be chosen by every receiver. However, this information is not available. In fact, the selection of the rate for the sender m made by receiver n depends also on the final rate of the layers of all the other senders, as all the streams compete for the same download capacity at the receiver side. In the above procedure the sender assumes that the receiver n will select the layer that leads to the highest layer utility, which may actually not be the choice that maximizes the real utility of the receiver. In order to make the algorithm fast the sender needs to rely only on local information, as a result, we need to make the aforementioned assumption.

5.1.3 Layers Selection Problem. After the layer rates allocation step, every sender m has fixed the rates of the L_m layers using the ideal rates computed by the receivers in the first step. The final step consists in the selection, by the receivers, of the layers they want to receive based on the download capacity and the importance of the users. The optimization problem can be stated as follows:

$$\mathbf{z}_n = \underset{\mathbf{z}'_n}{\operatorname{argmax}} \quad \sum_{m=1}^{N} \sum_{l}^{L_m} z'_{nml} U_m(r_{ml}) \tag{6a}$$

$$\text{subject to} \quad \sum_{m=1}^{N} \sum_{l=1}^{L} z'_{nml} r_{ml} \leq d_n \tag{6b}$$

$$\sum_{l=1}^{L_m} z'_{nml} = 1 - \delta_{nm} \quad \forall\, m \tag{6c}$$

$$z'_{nml} \in \{0,1\} \quad \forall n, m, l. \tag{6d}$$

This problem represents the discretized version of Problem (3). In fact, as for (3), the objective function of this problem represents the utility function of the receiver n, the download capacity constraint establishes the limit of the amount of data that can be downloaded, whereas the second constraints impose the limit of receiving at maximum one layer from each sender, as imposed in the original problem. This problem is a constrained Integer Linear Programming (ILP) problem [20] and can be solved exactly by using an ILP solver.

With respect to the original Problem (2), the rate allocation provided by the fast algorithm is not guaranteed to be optimal. The assumptions made in order to solve the second subproblem are not always valid. This solution method makes however the problem much simpler and permits to solve it in a distributed manner and in a single round of operations which is a large advantage in practice.

5.2 Iterative Rate Allocation Algorithm

The method described in the previous section is obviously not the only algorithm that can be used to find an approximate solution of Problem (2). In this section we develop a second iterative algorithm, which is based on a Lagrangian relaxation of the original problem. Compared to the previous method, which is able to find the solution in a single round of operations, the new procedure

iteratively modifies the optimization variables in the attempt to improve the value of the objective function while respecting the constraints.

As we want to have a distributed algorithm, we decompose the rate allocation problem among the different users following the decomposition methods that are usually used for Network Utility Maximization (NUM) problems [16]. The fundamental feature that permits to decompose NUM problems is the fact that the objective function corresponds to a sum of utilities, which each depends on a single sending rate. As a result, every sender is able to compute the derivative of the objective function with respect to its sending rate, hence to optimize that rate locally and independently of the other variables. In our case, this is only partially true however: the objective function derivative computed with respect to the layer rate r_{ml}, is independent of all the other layer rates but depends on the layer selection variables $\{\mathbf{z}_n\}$ (similar reasoning is true for layer selection variables: optimizing over $\mathbf{z}_n$ can be done independently of the layer selections of the other users but not of the variables $\{\mathbf{r}_m\}$). In order to have a distributed iterative algorithm we need to update the layer rates and the layer selection variables alternatively, by fixing the other ones.

In a distributed algorithm it is not only necessary to decouple the objective function but also the constraints. We can recognize different types of constraints in Problem (2): (2e)-(2f) are bounds on individual variables and can be handled locally. Constraints (2c) and (2d) impose conditions on the vectors $\mathbf{r}_m$ and $\mathbf{z}_n$ respectively, since $\mathbf{r}_m$ and $\mathbf{z}_n$ are computed by a single client node they can also be handled locally. However, the download capacity constraint (2b) is the constraint that relates all the variables of the optimization problem and needs to be replaced in a distributed iterative algorithm. An efficient method to handle such coupled constraints in distributed algorithms uses Lagrangian relaxation [16]. Such constraints are basically added to the objective function and a constraint violation corresponds to a penalty value that ultimately decreases the value of the objective function. The original optimization Problem (2) can be rewritten as follows:

$$D(\boldsymbol{\lambda}) = \underset{\mathbf{z}_n, \mathbf{r}_m}{\text{maximize}} \quad \sum_{n=1}^{N}\sum_{m=1}^{N}\sum_{l=1}^{L_m} z_{nml}U_m(r_{ml}) - \sum_{n=1}^{N}\lambda_n\left(\sum_{m=1}^{N}\sum_{l=1}^{L_m} z_{nml}r_{ml} - d_n\right) \tag{7a}$$

$$\text{subject to} \quad r_{ml} \le u_m \quad \forall\, l, m \tag{7b}$$

$$\sum_{l=1}^{L_m} z_{nml} = 1 - \delta_{nm} \quad \forall\, n, m \tag{7c}$$

$$r_{ml} \in [R_{\min}, R_{\text{MAX}}] \quad \forall m, l \tag{7d}$$

$$z_{nml} \in \{0, 1\} \quad \forall n, m, l. \tag{7e}$$

where $\lambda_n \ge 0$ corresponds to the Lagrangian multiplier, or dual variable, associated to the download capacity constraints of user n ($\boldsymbol{\lambda}$ denotes the entire vector of dual variables). The dual variables have different interpretations, in the NUM framework they are often thought of as prices associated to the utilization of the network resources [12]. We now give a brief intuitive explanation of this price interpretation and explain how dual variables can be useful for finding a solution to our original Problem (2). Firstly, note that the utility functions are strictly concave increasing functions, which means that larger rates will always improve the objective function of Problem (2). In Problem (7) this is not true anymore because of the penalty added to the objective function. Let us assume a fixed $\lambda_{n'} > 0$ and ignore for the moment the other constraints, moreover let us consider a layer of rate r_{ml} that is selected by user n' ($z_{n'ml} = 1$). When the rate r_{ml} grows excessively the download constraint of user n' becomes violated, the penalty value increases progressively reducing the gain of Eq. (7a). Since the utility functions are strictly concave, whereas the penalty grows linearly with the rate r_{ml}, at some point the gain in the utility will be counterbalanced by the penalty and a

further increase of the rate would make the value of the objective function decrease. Similarly to prices, the dual variables can be varied in order to control the users' resources consumption: large values would restrict the network usage enforcing the rates to respect the download capacity constraints, while smaller values incentivize the use of the network links. This intuition poses the basis for our iterative algorithm: we can alternate between updating the primal variables (layer rates and layers selection) and the dual variables in order to find a solution that optimizes the original problem and at the same time respects the constraints. If the primal variables violate one constraint, then we increase the dual variable of that constraint. In the next step the primal variables will be pushed towards the feasible set. On the other hand, if a constraint is not violated then the dual variable will be reduced. Note that this rule for the price update simply corresponds to update $\boldsymbol{\lambda}$ in the direction of the negative gradient computed with respect to $\boldsymbol{\lambda}$ of Eq. (7a).

From a more formal point of view, it can be proven that, for any fixed set of values $\boldsymbol{\lambda}$ the solution to (7) provides an upper bound to the original Problem (2). The value of this upper bound depends on the dual variables $\boldsymbol{\lambda}$. The problem of minimizing the upper bound is known as the dual problem, which under specific assumptions has the same optimal value of the primal problem [4]. The iterative procedure described above actually corresponds to an iterative method for minimizing the upper bound.

The use of the dual variables permits to design a distributed algorithm that improves the objective value of the original problem while respecting the download capacity constraints at equilibrium. Moreover, the updates of the primal and dual variables can be executed in a distributed way. We describe now the primal-dual method used, where at every step we slightly modify the primal and dual variables of the optimization problem in the gradient direction.

5.2.1 Layer Rates Iterative Update. The update of the layer rates is simply done by taking a small step in the direction of the gradient of the objective function of Eq. (7a) computed with respect to $\mathbf{r}_m$:

$$\Delta_{ml} = \alpha \left(\sum_{n=1}^{N} z_{nml}^k U_m'(r_{ml}^k) - \sum_{n=1}^{N} z_{nml}^k \lambda_n^k \right) \tag{8a}$$

$$r_{ml}^{k+1} = \max\{R_{\min}, \min\{r_{ml}^k + \Delta_{ml}, R_{\text{MAX}}, u_m\}\}, \tag{8b}$$

where $U_m'(\cdot)$ denotes the derivative of the utility function, Δ_{ml} represents the variation of the rate variable r_{ml} and $\alpha > 0$ is a simple parameter that controls the step length of the update equation. The second equation is needed in order to respect the constraints in Eq. (7b)-(7d). The primal variables $\mathbf{z}_n$ and the dual variables $\boldsymbol{\lambda}$ are fixed in this step.

Note that in some cases the derivative of the objective function of Eq. (7a), computed with respect to some layer rates, might vanish, which results in a null variation of the rate in Eq. (8). This happens when the layer is not selected by any receiver, i.e., $\sum_{n=1}^{N} z_{nml}^k = 0$. In this case, the layer does not contribute to the value of the objective function and the derivative is therefore null. If a layer is not selected in one iteration, it is very unlikely that it will be selected in the future iterations if its rate is kept fixed. Therefore, in order to avoid this pitfall, we can randomly change the rate of the layers that are not selected in order to promote exploration of new solutions.

5.2.2 Layer Selection Iterative Update. The layer selection variables are discrete and it is not possible to apply a gradient ascent step as above. We rather solve an integer programming problem similar to Problem (7) with respect to $\mathbf{z}_n$ with an additional constraint that limits the variation of

$\mathbf{z}_n$ with respect to its previous value. The problem is the following:

$$\mathbf{z}_n^{k+1} = \underset{\mathbf{z}'_n}{\text{argmax}} \quad \sum_{m=1}^{N} \sum_{l}^{L_m} z'_{nml} U_m(r_{ml}^k) - \lambda_n^k \left(\sum_{m=1}^{N} \sum_{l=1}^{L_m} z'_{nml} r_{ml}^k \right) \tag{9a}$$

$$\text{subject to} \quad \sum_{l=1}^{L_m} z'_{nml} = 1 - \delta_{nm} \quad \forall\, m, n \tag{9b}$$

$$\sum_{l=1}^{L_m} \sum_{m=1}^{N} (z'_{nml} - z_{nml}^k)^2 \leq 2 \tag{9c}$$

$$z'_{nml} \in \{0, 1\} \quad \forall n, m, l. \tag{9d}$$

This problem represents an integer programming problem with quadratic constraints and it is solved independently by every receiver. In order to limit the variable variations we introduce the quadratic constraint (9c). This constraint limits the l^2-norm of the difference between the old layer selection variable and the new one. This is equivalent to limiting the variation of the received layer for only one sender at each iteration. This constraint is meant to avoid abrupt variations of the layer selection variables and to improve the stability of the algorithm.

5.2.3 Dual Variables Iterative Update. The last update concerns the dual variables. As mentioned previously the update of the dual variables coincides with a step in the direction of the negative gradient of Eq (7a) computed with respect to $\boldsymbol{\lambda}$ followed by a projection onto the positive orthant:

$$\lambda_n^{k+1} = \left(\lambda_n^k + \beta \left(\sum_{m=1}^{N} \sum_{l=1}^{L_m} z_{nml}^{k+1} r_{ml}^{k+1} - d_n \right) \right)^+, \tag{10}$$

where $\beta > 0$ is a simple parameter that limits the step length of the dual variable update, and $()^+$ denotes the projection onto the positive orthant. We would like to stress the fact that dual variables are updated according to a very intuitive rule: if a constraint is violated the associated lambda will grow; on the other hand, if a constraint is neither violated nor tight the dual variable will decrease. Finally note that the dynamic update in Eq. (10) does not admit equilibrium points that are infeasible. This can easily be understood by setting the dual variable variation to zero in Eq. (10), in this case we see that the download capacity constraint must be respected.

5.3 Summary of the Rate Allocation Algorithms

From the descriptions of the two algorithms, we can identify the differences in their design. Whereas the fast algorithm is based on the similarity between Problem (3) and Problem (6), the iterative algorithm is based on the Lagrangian relaxation similarly to the usual NUM problems. Also, the features of the algorithm are radically different. The fast algorithm does not need an initial guess and provides a solution that is likely to be good, but it does not offer a way to further improve this solution. On the other hand, the iterative algorithm is able to gradually improve the solution but it requires an initial guess. Moreover, since the problem is not convex, local maxima are possible. Hence, a good initial guess is extremely important to achieve good performances. From this perspective, the two algorithms are not mutually exclusive but actually can be combined in order to achieve better solution. As depicted in Fig. 2, we can first run the fast algorithm to obtain a good initial guess of the solution, and then refine this guess using the iterative algorithm and improve the value of the objective function. In the next section we describe more in detail how a system that uses both algorithms can be implemented in a realistic environment.

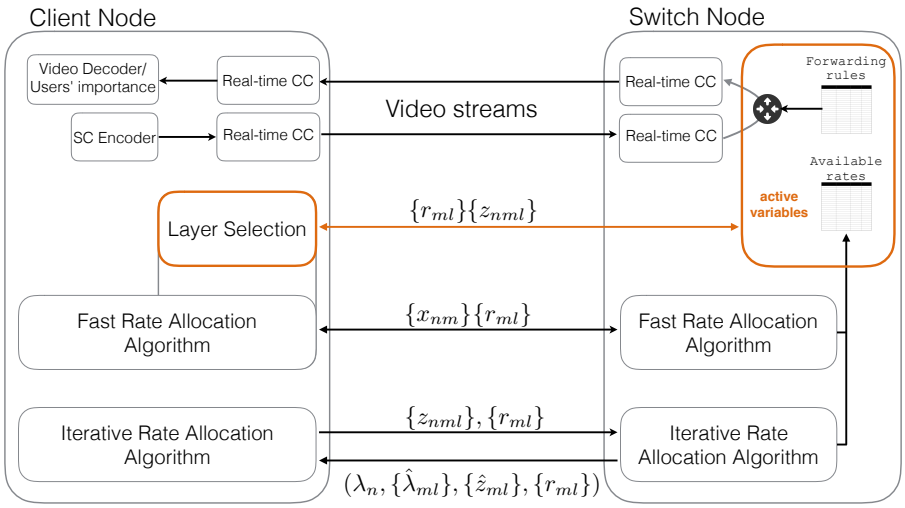


Fig. 3. System architecture.

6 PRACTICAL IMPLEMENTATION

We explain now the practical implementation of the algorithms described in Section 5. The system is composed of two types of applications: the client application and the switch node application. In Fig. 3 a simplified block diagram with the client and the switch node applications is depicted. Before continuing with the implementation description, we recall which information can be measured locally by the different components of the system. The client nodes have access to the value of their own upload and download bandwidth exclusively and to the utility functions of all the videoconference participants. The bandwidth values can be obtained from the CC whereas the utility can be extrapolated from the audio and video data received from the different users. For example, the video quality can be estimated by using some no-reference techniques and the relative importance of each stream can be inferred by detecting the current speaker (see for example speech activity detection methods, e.g., [19]). The switch node instead knows all the upload and download bandwidths of all users via the CC algorithm, but it is not able to measure the utility function since it has no decoding capabilities.

6.1 Fast Rate Allocation Algorithm Implementation

We first describe the implementation of the fast rate allocation algorithm proposed in Subsection 5.1. The fast algorithm is triggered by the client nodes that independently solve the ideal rate computation problem described in Subsection 5.1.1. If the computed rates are different from the old ones, they are communicated to the switch node using a reliable protocol.

Upon reception of the new ideal rates from any of the receivers, the switch node quickly schedules the transmission of the updated ideal rates to all the senders. The small waiting time before the transmission of the ideal rates permits to collect new ideal rates from other receivers before proceeding to the next rate allocation step. This ultimately reduces the communication overhead of the distributed algorithm. The users do not only communicate the new ideal rates to the switch node, but also the dual variable associated to the download capacity constraint of Problem (3). This information will be used later by the switch node in the execution of the iterative

rate allocation algorithm. When the users receive the new ideal rates forwarded by the switch node they independently compute the new layer rates using the algorithm described in Subsection 5.1.2 and send the new rates to the switch node. When the switch node has collected the corresponding values from each client, it forwards the complete list of available layers to all receivers.

When the users receive the new values of the available rates, they perform two operations: *i*) the sender subpart adapts the actual layer rates to the new values, *ii*) the receiver subpart solves the Layer selection problem described in Subsection 5.1.3 and sends the output of this optimization problem to the switch node. The switch node then updates the forwarding table according to the new rules communicated by the receivers. The forwarding table is a data structure that is maintained by the switch node. It contains a list of pair values (user identifier and layer identifier) for every user that indicates which layers should be forwarded to each endpoint. Every time a video packet is received by the switch node, the application checks which receivers have requested this pair of user-layer identifier and forwards the packet accordingly. Note that if users have different Round-Trip Times (RTTs) the updates of the layer rates and the layers selections are slightly desynchronized. As a result, the capacity constraints might be violated for a short period of time causing an increase in the communication delay. However, as it is shown in Section 7, the desynchronization effect for typical values of RTTs causes a limited delay increase. In the case where the delay increase may lead to a severe quality degradation of the interactive communication it is possible to control and limit the delay increase by properly dropping some packets or by improving the system implementation in order to take into account explicitly the desynchronization in the layer rates update.

At this point the fast rate allocation algorithm is completed and in order to improve the problem solution we trigger the iterative rate allocation algorithm.

6.2 Iterative Rate Allocation Algorithm Implementation

We now describe the implementation of the iterative rate allocation algorithm of Subsection 5.2, which uses the output of the fast rate allocation algorithm as initial guess. The switch node first executes a dual variable update according to Eq. (10)[1]. Using the new values of the dual variables and the current layer rate selection, the switch node computes the following quantities which are needed by the senders to compute the layer rates update of Eq. (8):

$$\hat{z}^k_{ml} = \sum_{n=1}^{N} z^k_{nml} \qquad \hat{\lambda}^k_{ml} = \sum_{n=1}^{N} z^k_{nml}\lambda^k_n. \tag{11}$$

The variable $\hat{z}^k_{ml}$ simply corresponds to the number of users receiving the layer l of sender m, whereas $\hat{\lambda}^k_{ml}$ is the sum of the dual variables associated to all the download links that are used by layer l of user m. The switch then communicates to each user the complete list of available rates ($\{\mathbf{r}^k_m\}$), the dual variable of the download capacity λ^k_m, the cumulative dual variable $\hat{\lambda}^k_{ml}$ and the total receivers $\hat{z}^k_{ml}$ for all the layers generated by the user m.

Each user, using the received information, updates the primal variables solving Eq. (8) and the integer programming Problem (9). The users then communicate the updated variables to the switch node. When the switch node receives the new variables from all the users the dual variables are updated and the next iteration starts.

When we apply a primal-dual algorithm, we know that, if the equilibrium point exists it has to be feasible. We might however achieve feasibility only asymptotically. In a realistic implementation we would like to obtain a solution that achieves feasibility in a finite amount of time. Secondly, it is

[1]The initial values for the dual variables are the ones collected from the ideal receiving rates computed in Problem (3).

also advisable, in case the download bandwidth is noisy, to have a small safety margin between the total downloading rate and the download bandwidth. In our implementation the switch node therefore uses a discounted value for the download capacities in the dual variables update, namely γd_n, with $\gamma = 0.98$.

The time taken to execute one step of the iterative algorithm depends on the RTTs between the switch node and the users' endpoints. In fact, when new dual variables are computed the next iteration is executed when all the users communicate the updated primal variables. This time is roughly equal to the maximum RTT plus the time required to solve the integer programming Problem (9). Lower communication delays do not only improve videoconference Quality of Experience but also the convergence time of the iterative algorithm. Unfortunately, communication delays result from the congestion control algorithm in place and do not depend on the proposed algorithm.

The iterative algorithm can be executed continuously in order to track small variations of the utility functions or variations of the upload/download capacities. Note that the users' RTTs pose an upper limit to the speed of the algorithm but the algorithm can obviously be executed at a lower speed. For example, the iteration speed can be decreased in order to save network resources if the objective function has reached a sufficiently high value.

6.3 Further Implementation Details

We briefly list here some other implementation details in the design of our videoconferencing system.

(1) Every stream flowing from the switch node to one of the receivers goes through a sending buffer. These buffers are drained according to the sending rate of the congestion control algorithm. They help to handle situations where the download capacity changes suddenly, or where minor synchronization errors among endpoints may generate a sporadic surplus of packets. This occurs when the layer rates and the layers selections change, for example. In our particular implementation we simply use a droptail management policy for the output buffers [8]. More advanced scheduling techniques can also be used at this stage without affecting the design of the switch node. The maximum size of the buffers is set in such a way to limit the maximum queuing delay to 100 ms; when this limit is reached, other incoming packets are discarded. It is worth noting that packet losses at this stage are not taken into account by the congestion control, since the sessions from the sender to the switch node, and from the switch node to the receiver(s) are completely decoupled.

(2) The forwarding rules that are actually used by the switch node can be updated at any time by the users. For example, if a sudden reduction of the download bandwidth occurs, then the users can recompute the optimal selection using the current layers and send the new rules to the switch node. The new rules will immediately become effective. This fast update is completely decoupled from the optimization algorithms, this is a sort of emergency action in case of sudden capacity (or utility) variations.

(3) In Fig. 3 it can be seen that in the proposed implementation the variables of the iterative algorithm are not the same as the active variables used in the videoconference, the switch node is responsible for deciding when to update the active values. Obviously, there are different possibilities for this update choice. One could decide that the primal variables computed by the algorithm at every iteration are always the active ones. This method is dangerous because the variables might actually provide an infeasible solution of the original problem. Therefore we prefer to adopt a different solution. In our implementation the users solve Problem (5.1.3) periodically, every second for example with the most updated layer rates of the iterative algorithm $\{\mathbf{r}_m^k\}$ and they communicate the optimal utility to the switch

Table 1. Settings for the 10 users scenario

Download capacity	[4 5 3.5 7 10.5 9 12.5 13 13.5 14] Mbps
Upload capacity	[0.7 0.7 0.7 1 1.4 1.5 2.1 1.8 2.0 1.8] Mbps

node. The switch node updates the active layer rates only if the expected total utility is larger than the current one, basically, we never perform a rate update that causes a drop of the overall utility function.

(4) If system conditions change remarkably it is convenient to restart the allocation process from the fast rate allocation algorithm. The restart of the fast algorithm is simply triggered by sending new ideal rates to the switch node. The decision whether to restart the process or not has to be taken by looking at the variations of the utility functions and of the upload/download bandwidth. In our implementation we restart completely the process if *i*) a bandwidth variation larger than 250 kbps occurs *ii*) the speaker of the videoconference changes.

7 SIMULATIONS

We study now the performance of the proposed algorithm. The system has been implemented in the network simulator NS3 [18]. The simulation results illustrate the behavior of the system and show performance comparisons between the fast and iterative algorithms. We also show the benefits of the proposed solution over a heuristic and non-optimized rate allocation method using the same system architecture.

7.1 Experimental Setup

The system described in Section 6 has been fully implemented in the network simulator. We use the NADA congestion control algorithm to send the media packets between nodes, with the implementation described in [25]. It is worth noting, however, that the operation of the proposed rate allocation method is independent of the underlying CC algorithm used by the system.

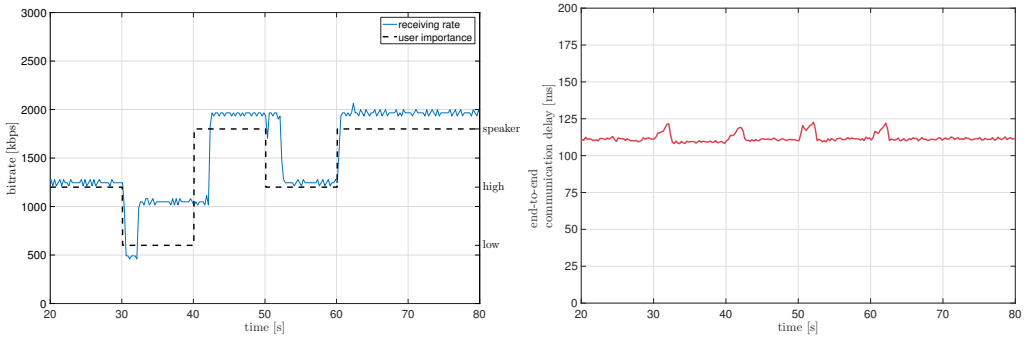
Similarly to many other works on NUM we use the logarithm of the rate in order to model the utility functions:

$$U_m(r_m) = w_m \log(r_m). \tag{12}$$

The coefficient w_m embeds the dependency of the utility function on the video content of user m. In the following simulations we assume that w_m represents only the users' activity and we assume to have the same video complexity for the different streams. The coefficient w_m can take the following values: 1 (low activity user), 2 (high activity user) and 3 (speaker). The weight of the user m can be obtained from the video-audio information coming from user m or can be signaled in the media packets. Receivers become aware of the importance of the other users simply by inspecting the incoming packets.

We consider a scenario with 10 users that participate in a videoconference. In order to have realistic bandwidth settings we set the capacities of the links according to different speed tiers provided by a large internet service provider company[2]. The values of the capacity for the upload/download links are listed in Table 1, we use heterogeneous capacity values in order to stress the performance of the proposed system. We finally assume that all the users are able to encode up to 3 video layers.

[2]available at: http://www.att.net/speedtiers



(a) Receiving rate (b) End-to-end delay

Fig. 4. Transmitted video stream from user 9 to user 5 (user 9 importance is shown)

7.2 Fast Algorithm Evaluation

In the first test we evaluate how quickly the fast rate allocation algorithm can react to users' utility variations. We vary abruptly the users' importance during the videoconference and observe the reaction of the rate allocation algorithm. Due to the large number of participants, it is convenient to focus on a single user and analyze that particular download rate evolution. In Fig. 4 we show the evolution of the bitrate and the end-to-end delay of the video stream sent from user 9 to user 5. The algorithm is able to track the variation of the user importance by providing fast rate adaptation (see Fig. 4a). The time required to adapt the rate corresponds to 2s, which strongly depends on the RTTs between the switch node and the users. It is easy to observe that the time required by the fast algorithm for completing the rate adaptation is approximately equal, in the worst case scenario, to three times the largest RTT between the switch node and the users plus the waiting time adopted by the switch node before forwarding the ideal rates (see Subsection 5.1). Fig. 4b shows the evolution of the end-to-end communication delay between the two considered users. This metric mostly depends on the CC algorithm used rather than the proposed videoconferencing system. However, it is important to verify that the small desynchronization between the adaptation of the layer rates and the layer selection update does not cause large delay variations in the data transmission. When the layer rates change, the experienced delay slightly grows, but this increase is contained within 20 ms (similar results hold for the other users) which confirms that the effect of the desynchronization does not compromise the QoS of the videoconference. Note that at time 30s and 60s the rate adaptations are almost instantaneous. In this case, it means that among the layers that are available at the time where the utilities change, there already exists a combination that can increase the utility function of user 5. Therefore the user selects this solution immediately, before the actual conclusion of the fast iterative algorithm, as discussed in the second paragraph of Subsection 6.3. For further results regarding the fast rate allocation algorithm we refer the reader to our former paper [9].

7.3 Iterative Algorithm Evaluation

In this subsection we evaluate how the layer rates evolve according to the iterative rate allocation algorithm. Since the purpose of the iterative algorithm is to progressively refine the rate allocation, and in order to evaluate the results more easily, we keep the activity of the users constant over time but we assign to each user a different importance level, namely the importance level assigned to each user are: $\mathbf{w} = [1\ 1\ 1\ 2\ 1\ 1\ 2\ 2\ 3\ 1]$. The algorithm works as described in Section 6: we first

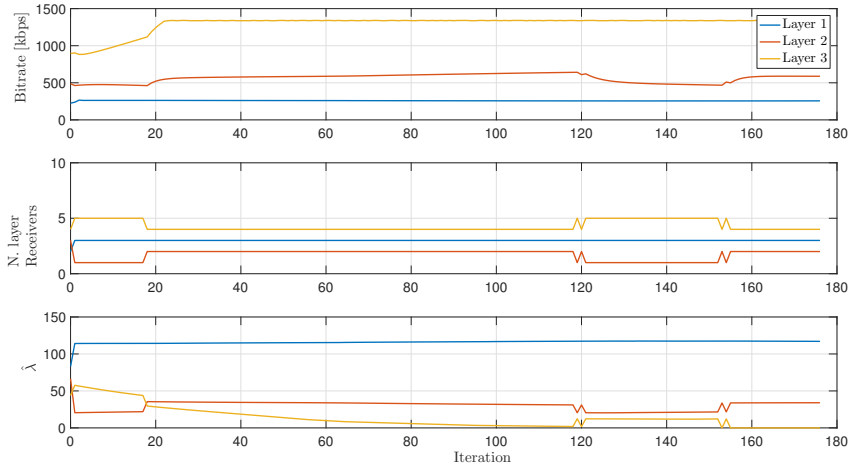


Fig. 5. Encoding bitrate, total receivers and cumulative dual prices for video layers generated by user 6.

execute the fast algorithm described in Subsection 5.1 and then we use the approximate solution as initial guess for the iterative algorithm of Subsection 5.2.

Fig. 5 and Fig. 6 depict the evolution of the layer rates, the total number of receivers and the cumulative dual variables associated to user 6 during the execution of the iterative algorithm. The upper plot in Fig. 5 shows the rate of the layers encoded by the user. In the initial phase we can notice the increase of the rate of layers 1 and 3. The increase of these two layers reveals one of the pitfalls that might affect the fast algorithm and that the iterative algorithm can correct. In this case other videoconference participants ideally want a lower rate from user 6 and some higher rate from other users. However, since other participants have a limited upload bandwidth (see Table 1) and are not able to provide the full requested rate, some free bandwidth becomes available this is ultimately allocated to users with a larger upload bandwidth, such as user 6 by the iterative algorithm, which results in a more efficient use of the upload link. The number of receivers of each layer and the cumulative dual prices are shown in the central and lower plot of Fig. 5 respectively. Note that the total number of receivers for all the layers is equal to 9, which means that all users in the videoconference receive the video stream of user 6, which is a constraint of the original Problem (2). In Fig. 6, some variables related to the receiving side of user 6 are depicted. In particular, the upper plot describes the total receiving rate at every iteration and the user download capacity, whereas the bottom plot illustrates the dual variable associated to the download constraint. As discussed in Subsection 5.2 the dual variable simply evolves according to the overuse or underuse of the capacity. When the rate exceeds the capacity the dual variable increases whereas when the constraint is not violated the dual variable decreases.

Using the same scenario, we show in Fig. 7 the evolution of the objective function of problem (7) as a function of the iteration. The objective function decreases, which is in agreement with the fact that Problem (7) is an upper bound of the original Problem (2) that is minimized by the iterative algorithm. The red line is obtained by solving Problem (2) with respect to the layers selection variables $\{\mathbf{z}_n\}$ with the layer rates fixed to the most recent iteration. At the beginning the objective value drops, because the iterative algorithm needs to violate the constraints in order to build up the dual variables, making all the layer rates too large to provide a good solution. When the dual variables are large enough they push back the solution of the primal-dual algorithm towards the

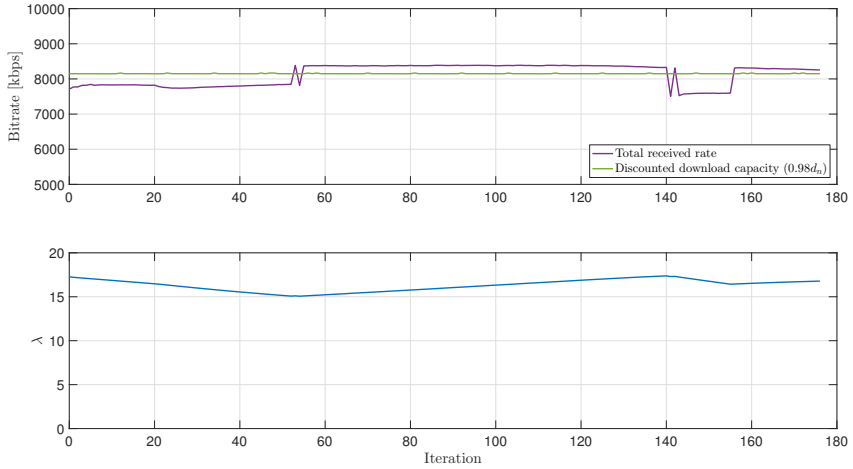


Fig. 6. Total received video bitrate, download capacity and dual variable associated to the download bandwidth of user 6.

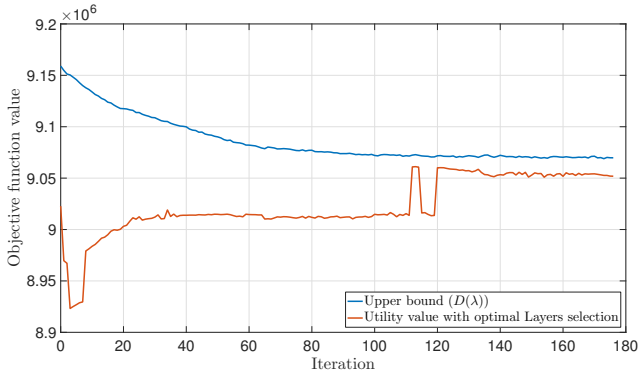


Fig. 7. Evolution of the Objective function of Problem (7).

feasible set of the original problem. At this point the layer rates are very close to the boundary of the feasible set, which translates in an efficient usage of the links and a higher total utility.

Finally a brief discussion on the number of iterations required by the iterative algorithm. In this simulation the iterative algorithm requires about 120 iterations in order to converge to the optimized allocation. If we consider that, a time equal to the largest RTT between the switch node and the users' nodes is required in order to execute one iteration of the algorithms, and assuming realistic RTTs between $100 - 400$ms, then the time required to find the solution is about $12 - 50$s. This is too large to use the iterative algorithm for fast rate adaptation, and this is why we combine it with the fast rate allocation algorithm, which instead is able to quickly converge to a good solution. Nevertheless the iterative algorithm, as we will see in the next subsection, is very effective in terms of utility maximization and channel utilization.

Table 2. Available layers used in the baseline solution

Number of available layers	Cumulative Layer rates
1	$1/4 \cdot u_n$
2	$[1/4\ 1/2] \cdot u_n$
3	$[1/4\ 1/2\ 3/4] \cdot u_n$
4	$[1/8\ 1/4\ 1/2\ 3/4] \cdot u_n$
5	$[1/8\ 1/4\ 3/8\ 5/8\ 7/8] \cdot u_n$

7.4 Performance Comparisons

In order to further compare the quality of the solutions achieved by the fast and the iterative algorithms we test them for different numbers of encoded layers. In this case we use the same network and users' importance as in the previous simulation but we vary the number of encoded layers for the users. The maximum number of layers varies from 1 to 5, and for each scenario we evaluate the total utility, the average upload and download link utilization after convergence for the videoconference participants. We also compare the performance with a baseline solution where the rates of the encoded layers are fixed. The layer rates used for the baseline solution are listed in Table 2. The rates have been selected in order to offer a tradeoff between an efficient use of the upload capacity and having a good probability of guaranteeing that all the users are able to receive all the streams. For this baseline method, the receiver algorithm is the same as in our proposed solutions: every user selects the combination of layers that maximizes his own utility function.

The results of these simulations are depicted in Fig. 8 where we see that the combination of the fast and iterative algorithms always outperforms the other methods. The fast algorithm is also very efficient in terms of total utility, but it is not always able to use the bandwidth in the most efficient way. The baseline solution obviously achieves lower values for all the metrics but the performance penalty decreases with the number of layers, since the optimization of the variables then becomes less important. When only few layers are available for a large user population, a proper optimization of the rates becomes important, and this is exactly the target scenario for our algorithm.

8 CONCLUSIONS

In this work we propose a method for solving the rate allocation problem in a multiparty video conferencing system. The considered system architecture is composed of a set of users, which are able to encode their own video at a limited number of different bitrates, and a central switch node, which enables application layer multicast communication among the videoconference participants. In order to preserve scalability the central node is kept as simple as possible and the system intelligence resides almost exclusively at the users' side. Unfortunately, the complexity of the rate allocation problem prevents us from being able to find easily the optimal solution. Therefore we design two distributed algorithms in order to find good suboptimal solutions. The first algorithm is a fast rate allocation method able to rapidly find an approximate solution of the original problem in a one-shot execution. The second algorithm is an iterative algorithm based on Lagrangian relaxation that gradually updates the variables in order to improve the initial solution. Since the two methods offer complementary features we can combine them in order to design a system that is able to guarantee a quick rate adaptation in the case where videoconference conditions change and a precise refinement of the solution for an efficient network usage. The proposed videoconferencing system has been implemented in a network simulator and its performance have been compared

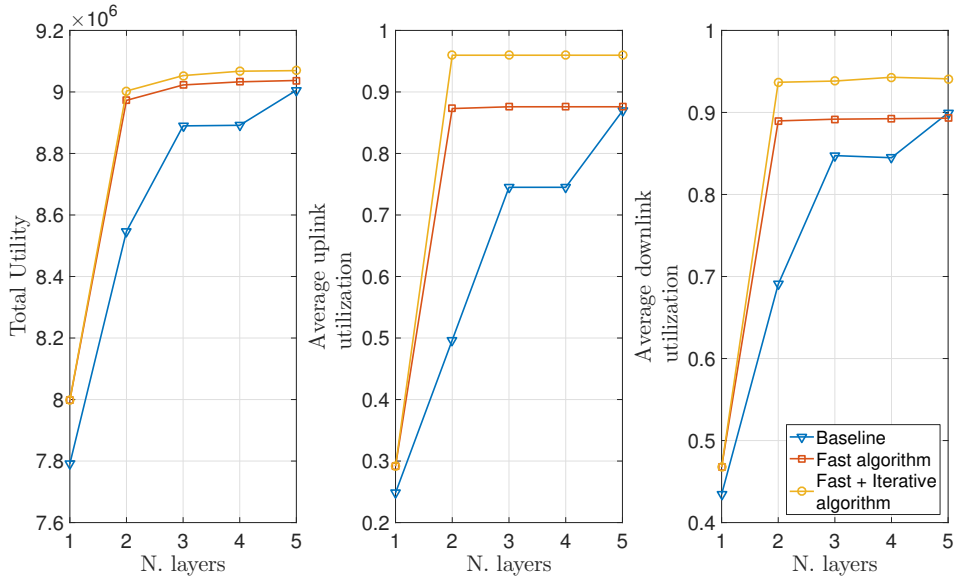


Fig. 8. Total utility, upload link utilization and download link utilization for the three methods under evaluation.

with a baseline solution. The results show the benefits that can be achieved by the combined use of the two algorithms in terms of fast rate adaptation, resource utilization and QoS provided to the users.

REFERENCES

[1] 2012. Cisco Video and TelePresence Architecture Design Guide. *Cisco* (2012).
[2] 2012. RTP Media Congestion Avoidance Techniques (RMCAT). IETF Working Group. (2012).
[3] 2015. The Zettabyte Era: Trends and Analysis. *White Paper, Cisco* (2015).
[4] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.
[5] Samuel Burer and Adam N Letchford. 2012. Non-convex mixed-integer nonlinear programming: a survey. *Surveys in Operations Research and Management Science* 17, 2 (2012).
[6] Minghua Chen, Miroslav Ponec, Sudipta Sengupta, Jin Li, and Philip A Chou. 2008. Utility maximization in peer-to-peer systems. In *SIGMETRICS Performance Evaluation Review*. ACM.
[7] Xiangwen Chen, Minghua Chen, Baochun Li, Yao Zhao, Yunnan Wu, and Jin Li. 2011. Celerity: a low-delay multi-party conferencing solution. In *International Conference on Multimedia*. ACM.
[8] Douglas E Comer. 2006. *Internetworking with TCP/IP*. Vol. 1. Pearson.
[9] Stefano D'Aronco, Sergio Mena, and Pascal Frossard. 2016. Distributed rate allocation in switch-based multiparty videoconference. In *7th International Conference on Multimedia Systems*. ACM.
[10] Alex Eleftheriadis. 2011. SVC and Video Communications. *White Paper, Vydio* (2011).
[11] Boris Grozev, Lyubomir Marinov, Varun Singh, and Emil Ivov. 2015. Last N: relevance-based selectivity for forwarding video in multimedia conferences. In *25th Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM.
[12] Frank P Kelly, Aman K Maulloo, and David KH Tan. 1998. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society* 49, 3 (1998).
[13] Eymen Kurdoglu, Yong Liu, and Yao Wang. 2016. Dealing With User Heterogeneity in P2P Multi-Party Video Conferencing: Layered Distribution Versus Partitioned Simulcast. *IEEE Transactions on Multimedia* 18, 1 (2016).
[14] Jin Li, Philip A Chou, and Cha Zhang. 2005. Mutualcast: An efficient mechanism for one-to-many content distribution. In *SIGCOMM Asia Workshop*. ACM.

[15] Jiangchuan Liu, Bo Li, Y. T. Hou, and I. Chlamtac. 2004. On optimal layering and bandwidth allocation for multisession video broadcasting. *IEEE Transactions on Wireless Communications* 3, 2 (2004), 656–667.
[16] Daniel Pérez Palomar and Mung Chiang. 2006. A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications* 24, 8 (2006).
[17] Miroslav Ponec, Sudipta Sengupta, Minghua Chen, Jin Li, Philip Chou, and others. 2009. Multi-rate peer-to-peer video conferencing: A distributed approach using scalable coding. In *International Conference on Multimedia and Expo*. IEEE.
[18] George F. Riley and Thomas R. Henderson. 2010. The ns-3 Network Simulator Modeling and Tools for Network Simulation. In *Modeling and Tools for Network Simulation*. Springer Berlin Heidelberg.
[19] Kirill Sakhnov, Ekaterina Verteletskaya, and Boris Simak. 2009. Dynamical energy-based speech/silence detector for speech enhancement applications. In *World Congress on Engineering*.
[20] Alexander Schrijver. 1986. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc.
[21] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. 2007. Overview of the scalable video coding extension of the H. 264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology* 17, 9 (2007).
[22] M. Welzl, S. Islam, and S. Gjessing. 2015. Coupled congestion control for RTP media. Internet-Draft. (2015).
[23] Yang Xu, Chenguang Yu, Jingjiang Li, and Yong Liu. 2012. Video Telephony for End-consumers: Measurement Study of Google+, iChat, and Skype. In *Conference on Internet Measurement Conference*. ACM.
[24] Yang Richard Yang, Min Sik Kim, and Simon S Lam. 2000. Optimal partitioning of multicast receivers. In *International Conference on Network Protocols*. IEEE.
[25] Xiaoqing Zhu and others. 2015. NADA: A Unified Congestion Control Scheme for Real-Time Media. Internet-Draft. (2015).