Distributed Congestion Control of Scalable Video Streams

Jean-Paul Wagner and Pascal Frossard

Ecole Polytechnique Fédérale de Lausanne (EPFL)

Signal Processing Laboratory - LTS4

Switzerland - 1015, Lausanne

Email: {jean-paul.wagner, pascal.frossard}@epfl.ch

Abstract— The paper addresses the problem of distributed congestion control in networks where several video streaming sessions share joint bottleneck channels. Based on the rate-distortion characteristics of the video streams as well as the requirements of heterogeneous streaming clients, the proposed algorithm determines the utility of rate increments in terms of overall quality benefits. We build on the utility-based congestion control framework initially proposed by Kelly [1], [2] and extend it such that it could efficiently handle heterogeneous delays in the network as well as the specific requirements of video streams. We then describe an original implementation of the proposed distributed congestion control algorithm using the common RTP/UDP/IP protocol stack for the efficient streaming of scalable video streams. Each video session independently adapts its streaming rate based on receiver feedbacks, which permits extension to large scale system without central coordination. Finally, we provide extensive simulation results that demonstrate the performance of the proposed solution. It is shown to successfully distribute the network resources among the different sessions such that the overall video quality is maximized. Experiments further demonstrate that the proposed scheme fairly cohabits with TCP, which is certainly an important advantage in today's network architectures.
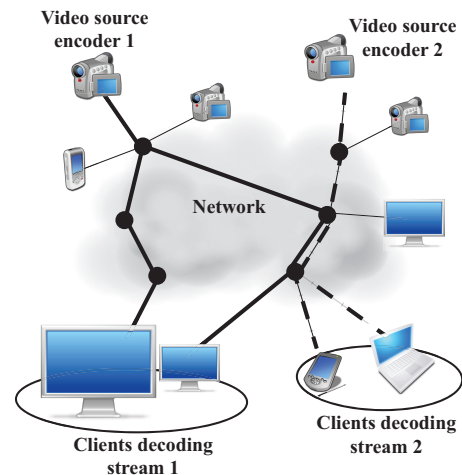
Fig. 1. A multi-session media streaming system whose objective is to deliver the best possible media quality to a set of heterogeneous clients while keeping the network stable in dynamic scenarios.

I. INTRODUCTION

The Internet has been rapidly evolving into a transport medium for the distribution of data such as audio and video streams due to the emergence of attractive multimedia applications. Multimedia streaming applications impose different requirements than those underlying typical FTP or HTTP traffic that is controlled by the Transport Control Protocol (TCP) [3]. On the one hand, the media streams have to be delivered at a rather high and sustained rate in order to deliver an acceptable Quality of Service (QoS) to the media client. On the other hand, an encoded media stream generally carries significant redundancy and therefore presents some inherent robustness to packet loss. The experience of the the end user is mostly driven by the delay and the fluctuations of quality in media streaming applications. Unsurprisingly, TCP does not perform very well in regulating the rate of media streams since it has not been designed for controlling such data flows. In particular, the typical sawtooth behavior in the controlled transmission rate profiles and the delays experienced by some packets due to the retransmission and exponential back-off policies are not ideal for media streaming. TCP-friendly rate control (TFRC) [4] permits to reduce the fluctuations of the transmission rates. It is more appropriate for delay-sensitive data and still offers a fair distribution of network resources. However, both TCP and TFRC do not consider the relative characteristics of the media streams and thus cannot lead to optimized performance in terms of overall quality of experience.

In this paper we explore an alternative rate and congestion control algorithm that is adapted to the specific characteristics of media streams receivers. We consider the framework illustrated in Figure 1. The media sources are captured and encoded in various points at the edges of an overlay network. Clients with heterogeneous needs in terms of frame rate or display resolution capabilities can connect to any media source from anywhere in the network. Our objective is to jointly achieve network stability with a fair distribution of resources in terms of video quality for all decoding clients, even in the presence of changes in the network.

We propose a *media-friendly* control algorithm, which

adapts the rate in the network to the needs of the media streams. Our work is based on a class of distributed congestion control algorithms that have been first proposed by Kelly [1], [2]. Their general objective is to maximize the aggregate *utilities* the end users retrieve from their network usage. The family of congestion control algorithms described in [1], [2] have the benefit of i) potentially allocating smooth rate profiles without abrupt transitions along time and ii) distinguishing each flow based on a *utility* criterion. Since we target a fair distribution of media quality in the client population, we propose to use a utility criterion that captures the quality benefit of an increment of bandwidth for each media stream. We extend the utility-based congestion control algorithms to practical systems with heterogeneous feedback delays and we show that the algorithms remain stable in these imperfect settings. Then we compute generic utility curves for scalable video streams, which permit to define several levels of quality by changing the spatial, temporal or SNR resolutions and hence to fully use the flexibility of scalable video streams. The use of a distortion-based criterion has been proposed in [5] but in more restricted settings that do not consider the different scalability dimensions. We finally propose a novel implementation of the media-friendly distributed congestion control scheme for various streaming scenarios where H.264/SVC streams are transmitted using RTP/UDP/IP protocols equipped with a new client feedback mechanism for network state inference. We provide extensive simulation results that demonstrate the convergence of the distributed congestion control algorithm, even in dynamic environments. The results also show that the bandwidth is properly distributed among the clients such that the quality is properly adapted in all concurrent streaming sessions. Finally, the proposed algorithm is shown to cohabit fairly with TCP flows.

The remainder of this paper is organized as follows. In Section II we provide some background on Kelly's theoretical framework [1], [2]. We extend it in Section III with an important stability proof for imperfect network settings. In Section IV we describe the proposed distributed congestion control algorithm and show how we can incorporate media-friendliness through an effective choice of utility functions adapted to scalable video streams. In Section V we describe the careful and fully scalable implementation of the proposed algorithm that can significantly boost the performance of the congestion control in practical scenarios. In Section VI we validate our findings through extensive NS-2 simulations. Finally we conclude with Section VII.

II. BACKGROUND

In this section, we provide some background and references on Kelly's Network Utility Maximization (NUM) problem that has been first stated in [2]. We give first a brief overview on distributed algorithms that solve the aforementioned problem. Then we provide a stability proof that is of high importance for the practical framework studied in this paper. Namely, we show that a system

controlled using the NUM framework is stable in the case of general rate utility functions and arbitrary round-trip delays. Finally, we discuss the *fairness* between flows resulting from the rate allocations computed in the NUM framework.

A. Network Utility Maximization

Let $s_i(t)$ be the rate assigned to flow i at time t. Further, assume that we assign to each flow i a *utility function*. This function describes by a utility value $U_i(s_i(t))$ the usefulness of the streaming rate $s_i(t)$ for an application that is served by flow i [6]. We suppose that the utility functions are continuous functions of the rate and that they are concave. Let $\mathcal{I}$ be the set of all flows in a given network, characterized in turn by a set of network links l with their respective capacity constraints.

The original NUM problem consists in finding for every time t a set of rates $s_i(t)$ that maximizes the sum of utilities for all flows, $\sum_{i \in \mathcal{I}} U_i(s_i(t))$, while satisfying the rate constraints on each link in the network. The direct solution of this optimization problem is difficult to find for large networks, and it further assumes that a central entity controls the rate of each flow. Instead, Kelly has proposed in [2] to solve a relaxed version of the original problem, by adjusting the rate of the flows in order to satisfy the following differential equations at any time t:

$$\frac{d}{dt} s_i(t) = \kappa_i s_i(t) \left(U_i'(s_i(t)) - p_i(t) \right). \qquad (1)$$

The term $s_i(r) \cdot U_i'(s_i(t))$ is referred to as the *willingness to pay* for the resource (i.e., the available bandwidth) and $p_i(t)$ is a pricing function, updated by the network, which indicates the price of the offered resource. In practice, $p_i(t)$ is often a congestion indication function that is fed back to the controller from either the network or the end user and κ_i is a constant gain factor.

Note that this system of differential equations drives each rate $s_i(t)$ into a steady state in which there is an equilibrium between the willingness to pay, and the flow's contribution to the resource's price. The former depends on the utility function $U_i(s_i(t))$ through its gradient, while the latter depends on the current network state, which is expressed by the congestion indication function $p_i(t)$. It is worth noting that Equation (1) can be straightforwardly discretized and turned into a rate update equation that leads to a practical implementation of a distributed control algorithm for solving the original NUM problem. To date, this has yielded the most promising implementations of control algorithms, even if there are other ways of solving the original NUM problem in a distributed way [7].

B. Fairness

The control algorithms based on Kelly's framework provide a fair distribution of the network resources among the different competing flows in the network. The fairness characteristic might however be defined in several different ways. In particular, the fairness could be linked to the distribution of the network bandwidth, or rather to

the distribution of network resources that balance utilities among flows. We refer to [8], [9] for a detailed study on the fairness in Kelly's framework. For the sake of completeness we provide a short summary here:

- If the controller relies on end-to-end feedback measures only, the control algorithm provides *proportional fairness*, meaning that flows congesting multiple routers are allocated less bandwidth in the stable state than flows congesting less routers.
- If the controller has access to the congestion levels of each router, which is however not a realistic scenario in today's Internet, the controller can adjust the rate for each flow according to the most congested router, thus providing *max-min fairness* [10].
- If additionally each flow is characterized by a potentially different utility function, the resulting rate allocations are weighted by the respective utility values in the stable state.

The practical system that we propose in this paper relies on end-to-end measures and uses different utility function for each flow. Hence, the control algorithm based on Kelly's framework provides in this case a *utility-proportional* fairness in the bandwidth allocation.

Finally, both stability and convergence of the distributed congestion control algorithm have been extensively studied for systems based on the differential equations given in (1) (see for example [2], [11]–[13] and references therein).

III. STABILITY WITH RANDOM FEEDBACK DELAYS

The ideal system described in the previous section has only marginal practical relevance since it assumes that network or client feedbacks are immediately available at each source in order to update the rate of the corresponding flow. In practical systems however, each flow i has a fixed starting point (the sender) as well as a fixed end point (the receiver) in the network. The route that connects them may change in time, which therefore affects the round-trip delay between sender and receiver. Moreover, the congestion level of each router that is traversed by the flow is dynamically changing. The delay experienced by feedback information is not only heterogeneous across flows in the network, but it has also a random distribution for each flow. When the system experiences some delays in the feedback loops, the differential equation of Eq. (1) reads as:

$$\frac{d}{dt}s_i(t) = \kappa_i \left(s_i(t) U_i'(s_i(t)) - s_i(t - D_i^R) p_i(t - D_i^B) \right),$$
(2)

In this case, D_i^R is the round-trip delay and D_i^B is the delay on the back-trip from the point in the network that created the feedback to the sender. Note that the congestion indication function $p_i(t - D_i^B)$ is synthesized D_i^B time units earlier, and that it relates to the rate allocated by the controller D_i^R time units earlier.

In the last years, a substantial amount of research has been devoted to providing stability results for controllers with delayed feedbacks. For example the authors in [11]

provide results for the case of equal round-trip delays for each flow. More recently, authors in [14] have studied the stability of systems with arbitrary but constant round-trip delays. Further stability results are provided in [10], [15]–[19] and references therein. In each of these works, the scenario corresponds to particular applications: either each flow uses the same utility function, or the delays are different for the various flows but constant in time, or the parameters of the rate update equation are dynamically adjusted with respect to the experienced delay. The latter implies that a precise measurement of the experienced round-trip delay is available at the controller, so that oscillations can be avoided in the system. The application considered in this paper does however not correspond to any of these scenarios. Therefore we need to extend the stability proofs to a more generic case.

The application we are targeting specifically calls for stability results for a scenario with *general* concave utility functions for each media stream and arbitrary round-trip delays. In [20], the author provides an elegant stability proof for the case where general utility functions are used, and where the round-trip delays in the network are arbitrary across flows, but constant in time. The author conjectures that the system remains stable if the round-trip delays take on arbitrary values. In what follows we borrow the framework from [20] and we extend the stability result to the case where generic, concave utility functions are used and where the feedback is arbitrarily delayed for each flow.

We consider a network made up of L links, indexed by l. We call $\Delta_{l,i}$ the round trip delay experienced by data transmitted from source i to traverse link l and get back to the source. Hence the experienced round trip delay D_i^R for user i takes on values in the set $\{\Delta_{1,i}, \ldots, \Delta_{L,i}\}$, depending on the receiver of flow i. Further we denote by Δ_l the maximum round trip delay for data to get sent from any source, traverse link l and get back to the source: $\Delta_l = \sup_i\{\Delta_{l,i}\}$. Let $\overline{D}$ be an upper bound on all experienced round trip delays in the network: $\overline{D} = \sup_i\{D_i^R\}$. Clearly, the following relation holds:

$$\overline{D} \geq \Delta_l, \;\; 0 \leq l \leq L.$$
(3)

The main result of [20] states that the system of delayed differential equations (2) is asymptotically stable if $\kappa_i > 0$, $U_i(\cdot)$ is a concave function for each i, and if all the round trip delays in the network coincide with a single scalar, i.e., $D_i^R = D, \forall i$. A sufficient condition for this to be the case is that the spectral radius of matrix N, given as

$$N = D \sum_{0 \leq l \leq L} M^{(l)}$$
(4)

is smaller than 1. This is shown to be the case *for any scalar D*, if the matrix M, which does not depend on the round trip delays, is positive definite. This condition is in turn verified if $\kappa_i > 0$ and the utility functions $U_i(\cdot)$ are concave. We now extend this stability result to arbitrary feedback delays.

Theorem 1: Assume that the round trip delays D_i^R take on arbitrary values bounded by the scalar $\overline{D}$. Then the system given in Eq. (2) is asymptotically stable under the assumptions that $\kappa_i > 0$ and the Utility functions $U_i(\cdot)$ are concave $\forall i$.

Proof: Following the same reasoning as in [20], a sufficient condition for this to be true is that the spectral radius of matrix N' is smaller than 1, with

$$N' = \sum_{0 \leq l \leq L} \Delta_l M^{(l)}, \qquad (5)$$

and M the same as in the previous development. Let us introduce the following matrix:

$$\overline{N} = \overline{D} \sum_{0 \leq l \leq L} M^{(l)}. \qquad (6)$$

From Eq. (4) we know that the spectral radius of $\overline{N}$ is less than 1. Using Eq. (3) we can further bound the elements of matrix N' as follows:

$$N' \leq \sum_{0 \leq l \leq L} \overline{D} M^{(l)} \qquad (7)$$

The spectral radius of the right-hand side of (7) is smaller than 1. The proof is concluded using the following Lemma, which states that the spectral radius of the right-hand side of (5) is smaller than the one of the right hand side of (7), hence it is also smaller than 1.

Lemma 1: Let A and B be two matrices, the entries of which, indexed by couples (n, m), all satisfy

$$|A_{n,m}| \leq B_{n,m} \qquad (8)$$

and hence, the $B_{n,m}$ are real, nonnegative. Then, the spectral radius, i.e., the largest positive eigenvalue of A, is smaller or equal to that of B. [20] ∎

This result of stability is important for the design of the congestion control algorithm proposed in the next section. It is moreover most relevant to any practical system as in practice the round trip delays that are observed are always bounded.

IV. Distributed Rate Allocation Algorithm

We propose now a distributed rate allocation algorithm for scalable video sequences based on the utility maximization framework described above. We first present a discretized version of the system of differential equations given in the relation (2), which leads to discrete-time rate update equation. Then we propose utility functions that are adapted to practical scenarios for video streaming.

A. Rate update equation

The distributed control algorithm can only act at discrete time instants in practice. An approach based on finite differences can be used to obtain a discrete-time version of the Eq. (2) as suggested in [11]. This results in the following update equation:

$$
\begin{aligned}
s_i(t) \;=\; & s_i(t-1) + \kappa_i\big(s_i(t-1)U_i'(s_i(t-1)) \\
& - s_i(t - D_i^R)p_i(t - D_i^B)\big),
\end{aligned} \qquad (9)
$$

The congestion indication function (pricing function) for flow i can be computed based on the *received* rate $r_i(t - D_i^B)$ measured at the client at time $t - D_i^B$. In this case, it reads

$$p_i(t - D_i^B) = \frac{s_i(t - D_i^R) - r_i(t - D_i^B)}{s_i(t - D^R)}. \qquad (10)$$

Note that delays induce a temporal shift between the time at which the reference sending rate, the actual sending rate, and the received rate are computed in the system. This slows down the convergence of the system. Using Eq. (9), the sender updates the rate $s_i(t)$ using the last taken control $s_i(t-1)$ as reference rate and a delayed feedback term. This feedback represents pricing information about the control taken D_i^R time units earlier. For example, if there is no congestion, the received rate is equal to the sending rate at time $t - D_i^R$. In that case the update equation will lead to a pure increase of the rate. In the event of a congestion, only part of the sending rate is received so that $r_i(t - D_i^B) < s_i(t - D_i^R)$. Hence the price of the resource for flow i is increased and the rate will be re-adjusted downwards accordingly. Note that the receivers provide periodic feedbacks to the servers and that these feedbacks do not correspond to specific frames. The feedbacks are generally not instantaneous; if they are frequent enough the servers however react properly to changing network conditions and the system stays stable.

In order to cope with delays and improve the stability of the system, we can rather use $s_i(t - D_i^R)$ as the reference rate for the update equation [16], [21], so that the temporal drift between reference rate and pricing function is virtually cancelled. However, differently from [21] we do not intend to use a constant *willingness to pay* throughout the network, as the rate allocation should reflect the relative utilities of the various streams. Hence, in order to eliminate the temporal drift between the term of Eq. (9) that increases the reference rate and the feedback term that penalizes the reference rate, we need to evaluate the Utility function at the rate given by $s_i(t - D_i^R)$ as well. Finally, the proposed controller use the following discrete-time rate update equations:

$$
\begin{aligned}
s_i(t) \;=\; & s_i(t - D_i^R) + \kappa_i s_i(t - D_i^R) \\
& \cdot \big[U'\big(s_i(t - D_i^R)\big) - p_i(t - D_i^B)\big].
\end{aligned} \qquad (11)
$$

From a networking point of view, it can be inferred from the rate update equation (11) that the average experienced loss rate $p_i(\cdot)$ is equal to the gradient value of the utility function, evaluated at the rate operation point that is reached in the stable state. The controller then maximizes the system's aggregate utility by iteratively allocating more rate to streams that have a larger benefit at the current rate operation point.

As the normalized congestion signal $p_i(\cdot)$ takes values in the interval $[0, 1]$, the utility gradient values of any stream in the system are normalized by a system-wide constant. This constant corresponds to the largest gradient value for utility functions in the system. It is important to note that the normalization constant should not be

stream-dependent in order to avoid distortions between the relative utility curves assigned to different streams. The maximum gradient value in the system also drives the maximum observed stable-state loss rate per stream throughout the system. Hence, by correctly scaling all the gradients, we can bound the maximum loss rate π to the maximum value that can be tolerated by the target application. Hence, we can finally rewrite the Eq. (11) as:

$$
\begin{aligned}
ccs_i(t) \;=\; & s_i(t - D_i^R) + \kappa_i s_i(t - D_i^R) \\
& \cdot \; \left[\pi \cdot U'\left(s_i(t - D_i^R)\right) - p_i(t - D_i^B) \right] \quad (12)
\end{aligned}
$$

B. Video Utility Functions

In the analysis about the stability of the distributed rate allocation algorithm, the only assumption on the continuous utility functions relies in their concavity. It leaves quite some flexibility in the choice of these functions, such that they correspond to the characteristics of the target application. An obvious choice for utility functions is to relate the utility of a media stream to the rate-distortion characteristics of the encoded sequence. In that case the control algorithm iteratively allocates the rates among streams proportionally to their contribution to the average video quality computed on all the streams. We focus in the rest of this paper on video encoders that provide the possibility to generate traffic that can be tuned to achieve any rate within a bounded rate interval. Such encoders include for example Motion-JPEG2000, in which each frame is progressively intra-coded: the rate of each frame can thus be adapted by selectively dropping wavelet coefficients, while increasing the distortion gracefully. Another example is given by the progressive refinement (PR) slices of the scalable video coding (SVC) amendment to the H.264/MPEG-4 AVC standard. Using SVC, a video can be decoded at the full encoding rate, yielding the highest possible decoded quality in terms of SNR, frame rate and resolution. Aside from this, one has the option to decode a number of sub-streams that have been specifically included while encoding. Each of the sub-streams represents a version of the video that is degraded in either SNR, frame rate or spatial resolution, or any combination of these. Typically it is up to the application to decide which sub-stream is most useful. Finally, each of these sub-streams can be encoded using progressive refinement slices, providing fine granularity scalability (FGS). These slices can be cut at any point in order to finely tune the rate within a substream, while gracefully increasing the distortion. Both of the aforementioned encoding choices rely on fine grained rate adaptation that results in SNR scalability. Hence they are able to generate the kind of traffic we aim for: any rate within a bounded rate interval can be achieved. As this results in SNR scalability, the resulting rate-distortion curves over that interval are concave and can be used as utility functions for the respective streams.

Examples of the utility functions used in this paper are illustrated in Figure 2 for a number of test sequences at different frame rates and resolutions. They have been
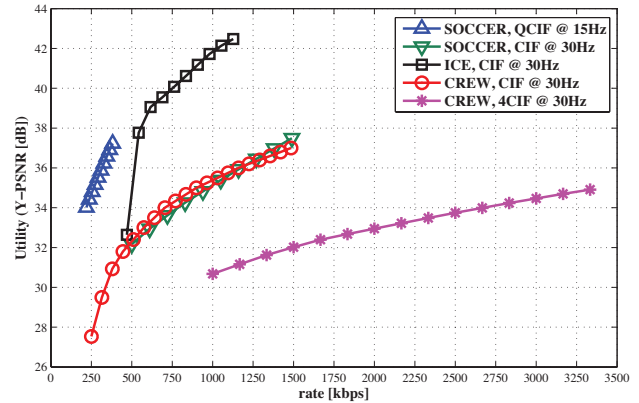

Fig. 2. Utility curves can reflect the rate-distortion characteristics. Here we show the rate-distortion curves of several H.264-SVC(FGS) encoded test sequences (SOCCER, CREW, ICE).
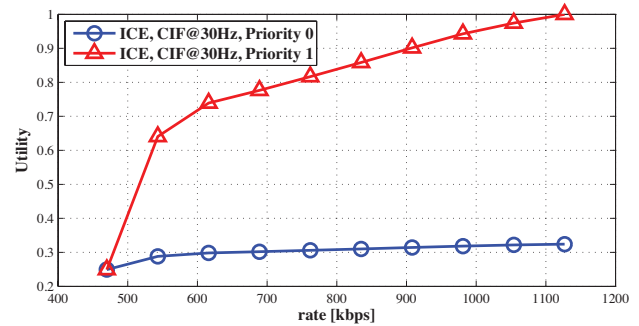

Fig. 3. Utility curves can also reflect traffic prioritization. Here priority class 0 uses the (scaled) R/D information for the SOCCER sequence, whereas users in priority class 1 use a utility function that has a gradient five times as large at each rate point.

computed on video sequences encoded using the H.264 SVC extension and using PR slices. The sequences have first been segmented into Groups of Pictures (GOPs) of equal size, and each GOP has been encoded independently into PR slices. The utility functions for the complete sequence have finally been extracted by decoding each GOP at a number of fixed rate points, and by averaging the resulting Y-PSNR value at each rate point for all the GOPs of the sequence. Note that we do not rely on an analytical model to specify the utility curves for each stream contrarily to [5], [22], but we rather use the exact rate-distortion information.

Finally, it should be noted that our framework is very generic and that any concave continuous function is valid as utility function. In particular, priority or service classes can also be incorporated in the utility functions. For example, we can design utility functions for several groups of streaming clients interesting in the same video sequence. The rate for receivers in the lower priority class is adapted using to the rate-distortion function. The receivers in the higher priority class use a different utility function, whose gradient corresponds to a scaled version of the rate-distortion curve at any rate point. Clients in the higher priority class hence see the quality of their

video streams be adapted faster and reach a higher level. Such utility functions are illustrated in Figure 3, where the utility gradient is multiplied by five for the high priority class.

V. PRACTICAL IMPLEMENTATION

We propose now a practical implementation of the control system described above. We have shown that the distributed control algorithm is stable, even with heterogenous feedback delays, which are likely to happen in real scenarios. We outline the scalable nature of the proposed framework and we explain in detail the design choices for the rate update equation and the distributed control in a video streaming system.

A. Design issues

From the above development, it is clear that the rate update equation (see Eq. (12)) has ideally to be applied as often as possible in order to emulate a continuous time control system. This requires the availability of accurate feedback at each moment in time and at each end-point in the network. At the same time, it implies that the sending rate can be adapted at any time instant. However, when we are dealing with real video streams, it is clear that we cannot adapt the video rate at any arbitrary time instant. For example, dropping random parts of the bitstream results in large and uncontrolled losses of quality due to the inherent decoding dependencies between video elements. Scalable video coding provides an interesting solution for flexible adaptation of the bitstream. For example, encoding formats such as H.264-SVC(FGS) form independently decodable compressed units such as Groups of Pictures (GOP), which are typically groups of 16 or 32 frames. They further offer the possibility to extract a substream of a given rate from any independently decodable entity of the stream. The rate control has therefore to be performed on GOPs, and the rate update equation of the control system should be synchronized with GOP boundaries. It is applied after each transmitted GOP and defines the rate of the substream to be extracted from the next GOP.

The decision of the control algorithm relies on the feedback received from the clients, about the state of the streaming session. In addition, it uses the result of the decisions taken in previous iteration of the algorithm, as seen in the update equation (12). The controller has to know the sending rate that has been computed $t - D_i^R$ time units earlier, where D_i^R is the *arbitrary* round trip delay. Maintaining the history of earlier decisions for each flow at each sender does however not provide a viable solution as it does not scale. In addition, it relies on very accurate round trip Time measurements in order to avoid any drift between the sending rate that is actually used in the update equation and the feedback that is received. The utility function is evaluated on the sending rate, yielding the willingness to pay for the bandwidth that is offered, while the congestion signal that is fed back gives the price of the bandwidth resources. Any temporal drift between the
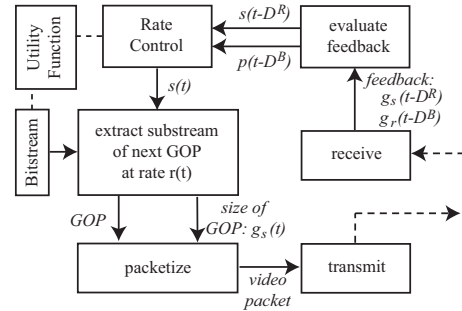

Fig. 4. Schematic view of our sender implementation.
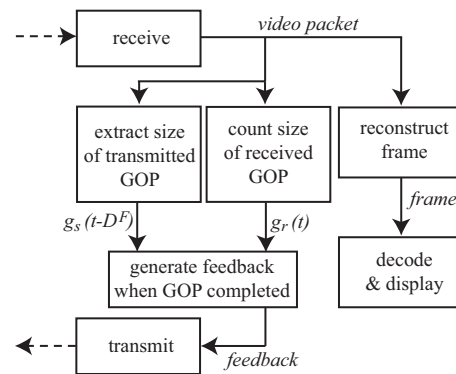

Fig. 5. Schematic view of our receiver implementation.

computation of these values slows down the convergence of the distributed algorithm. A scalable system has therefore to avoid any temporal drift by grouping together the congestion signal and the sending rate it corresponds to. In the next section, we propose a light-weight application layer protocol that respects the above constraints using the time-stamp information included in the transported video streams.

B. Proposed control system

We propose now an implementation of the distributed control system at the application layer, for streaming scalable video over the classical RTP/UDP/IP protocol stack. For the sake of clarity we drop the index i that specifies a particular flow and sender/receiver pair in what follows. We describe in details the behavior of a client and sender pair; all the concurrent streaming sessions in the system adopt the same strategy.

A schematic view of the sender implementation is first given in Figure 4. Upon reception of a feedback message, the rate controller given by Equation (12) updates the targeted sending rate to $s(t)$, which depends on the media stream that is being transmitted through the chosen utility function. The next GOP of the scalable bitstream is then optimally truncated so that its rate matches the target rate. The resulting substream is then packetized according to the video frames and Network Abstraction Layer (NAL) units and injected into the network. The exact size of the extracted GOP, denoted by $g_s(t)$, is added to the header of all the packets in the GOP. Note that the sender knows the
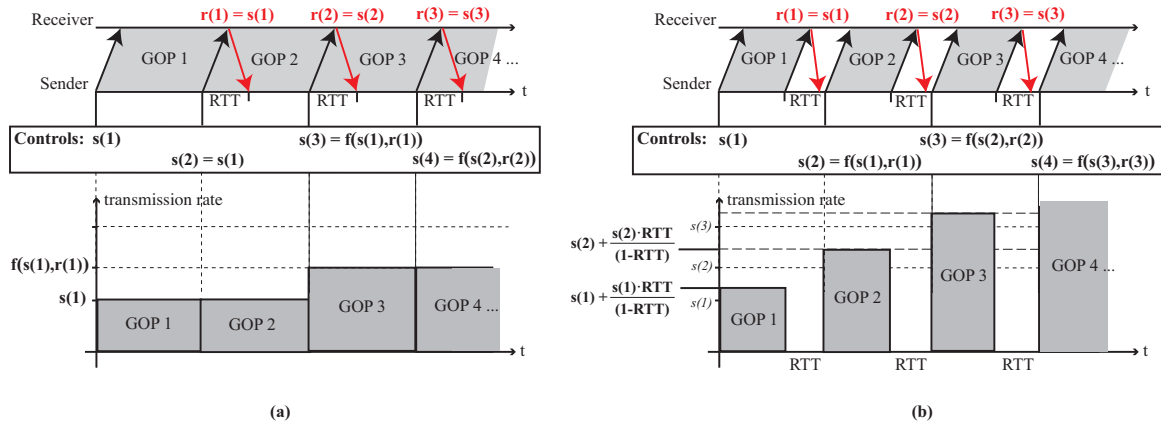
Fig. 6. Guard interval illustration. *Left:* When no guard interval is used, the available feedback is only considered in the control loop at a later stage. For example, the rate for GOP 3 is dependent on the rate and feedback of GOP 1. *Right:* By using a guard interval, the feedback can be integrated more rapidly in the control decisions.

framerate f [Hz] of the stream as well as the length k of the GOP in frames, since those are usually negotiated by the sender and the receiver. The sending rate is therefore simply given by

$$s(t) = \frac{g_s(t)}{k} f \ . \qquad (13)$$

The client behavior is then represented in Figure 5. It receives the packet stream and detects potential packet losses. It further reassembles the media bitstream and sends the reconstructed decodable parts of the bitstream to the decoder. At the same time, the receiver counts the number of bits $g_r(t)$ that are received for the current GOP. Periodically, the receiver sends a feedback to the sender where it includes the received size $g_r(t)$ as well as the GOP size $g_s(t - D^F) \geq g_r(t)$ that is read from packet headers. When the feedback eventually reaches the sender, it accurately reconstructs the control decision taken $D^R = D^F + D^B$ time units earlier. Even if the round trip time D^R is arbitrary, the sender can replicate the past control decision and compute the previous sending rate as

$$s(t - D^R) = \frac{g_s(t - D^R)}{k} f. \qquad (14)$$

Similarly it reconstructs the rate that is actually received by the client as:

$$r(t - D^B) = \frac{g_r(t - D^B)}{k} f \ . \qquad (15)$$

The sender has thus access to all the information necessary to evaluate the congestion signal given in Equation (10). Note that it uses the reference sending rate $s(t - D^R)$ without relying on a stored history of controls nor on exact round trip delay measurements. All the information is rather extracted from the feedback packets and the media stream characteristics that are known at the sender. Furthermore, the information that is transmitted in the feedback packet can be accurately synthesized at each client even when media packets are lost since all the packet headers contain the GOP size information $g_s(t)$. It is therefore sufficient to receive one media packet of

the GOP for the client to generate an accurate feedback signal that stabilizes the control system.

Even if such a system is scalable and robust, it does not perform optimally yet due to network latency. Consider for example the scenario depicted in Figure 6(a). The feedback is only sent after a GOP n is completely received. Hence, it is available at the sender at the earliest one round trip time after the last packet of the GOP has been transmitted. At the time the controller has to decide on the sending rate for the next GOP $n + 1$, it does not have access to any feedback yet. It can thus only decide to keep on transmitting at the same sending rate. Even though an accurate feedback becomes available during the transmission of the GOP $n + 1$, it can not be incorporated in the control system due to the structure of the video streams. Therefore, it can only affects the sending rate of the GOP $n + 2$, which introduces a latency that is almost equal to the duration of one GOP.

In order to avoid such a latency that slows down the convergence of the control system towards a steady-state solution, we propose to increase slightly the actual transmission rate. The data of a GOP with rate $s(t)$ is thus transmitted at a rate $s(t) + \frac{s(t) \cdot RTT}{1 - RTT}$, where RTT is the round trip time in the system. It therefore creates a guard-interval between the transmission of two adjacent GOPs n and $n+1$, so that the feedback information about the GOP n is likely to be received on time for controlling the sending of the GOP $n + 1$. Figure 6(b) sketches the improved control sequence. The small increase in the transmission rate permits to adapt the sending rate of the streaming session faster compared to the case illustrated in Figure 6(a).

VI. SIMULATION RESULTS

A. Setup

We illustrate now the behavior of the distributed rate allocation algorithm with simulation results in different streaming scenarios. We consider the general network topology depicted in Figure 7, where eight different
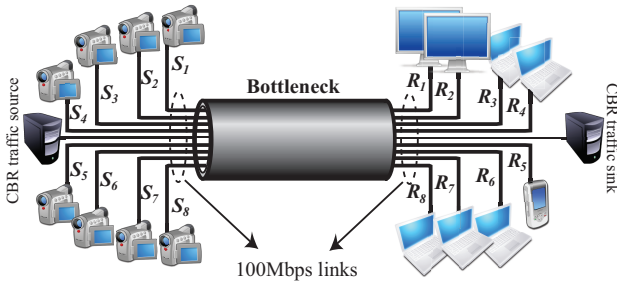
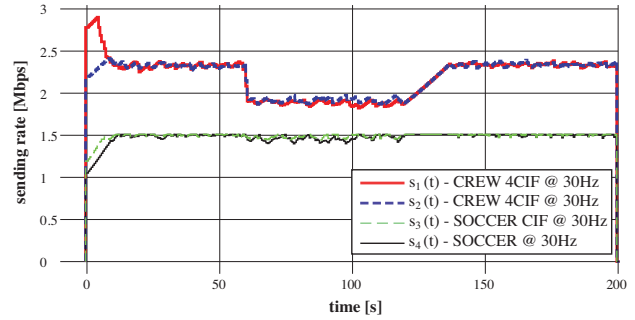Fig. 7. Simple simulation topology, each source streams SOCCER CIF @ 30Hz

Fig. 8. Resulting sending rates/controls for scenario 1.

sender-receiver pairs connect to a network through high-speed links but share a common bottleneck link. Each sender-receiver pair runs its own rate-control loop independently of the other pairs. It relies only on end-to-end feedbacks and the Utility information relative to the video sequence. In particular, each sender and receiver know neither the capacity of the bottleneck link, nor the number and nature of the other streams that use the same bottleneck link. Finally, we also consider a constant bit rate (CBR) source-sink that also uses the same bottleneck link without any adaptive rate control.

The sequences transmitted by the senders are encoded using the H.264 SVC reference software (JSVM). In the encoding, we constrain each GOP of a given stream to span the same range of encoding rate. Unless otherwise stated, the Utility functions are given by the Rate-Distortion information extracted from the encoded sequences, as depicted in Figure 2. The distribution of the test video sequences between the different sender-receiver pairs is given in Table I for the scenarios considered in our simulations.

The proposed rate allocation algorithm is implemented in the application layer of the NS-2 simulator platform, and controls the rate of the underlying UDP transport protocol. The sender runs the control algorithm using the rate update equation (i.e., Eq. (12)) based on the utility functions of the transmitted stream and the feedback it gathers from the receiver. Unless otherwise stated, the maximum loss rate factor π is set to 0.05 in order to limit the degradations due to loss and the gain factor κ equals 1. Both values have been selected empirically in order to ensure good system performance. The sender then extracts a SVC substream at the target sending rate with help of the tools available in the JSVM software distribution and sends the appropriate packets to the receiver. Upon re-

ception of each packet, the receiver checks for losses and eventually reconstructs a received packet trace for each GOP. It forms a feedback packet based on the information it has received. The video sequence is finally decoded with the packets that have been correctly transmitted.

B. Adaptation to changing bottleneck capacity

In the first set of simulations, we analyze the effect of a changing bottleneck capacity, or equivalently the effect of varying background traffic in the Scenario 1. We set the bottleneck bandwidth to 8 Mbps and let senders S_1, S_2, S_3 and S_4 start transmitting simultaneously their respective video streams at time $t = 0$. After 60 seconds, we add a 1 Mbps CBR background traffic stream in the same bottleneck channel, which translates into a sudden bottleneck capacity reduction for the four sender-receiver pairs. The background traffic is switched off again after one further minute.

The resulting sending rates are shown in Figure 8 that illustrates some key properties of the proposed algorithm. Clearly, the algorithm converges quickly to a stable state at the beginning of the transmission, as well as around the changes in background traffic. The stable state corresponds to maximization of the utilities of the different streams. Even though each stream is regulated independently of all the other ones, the same video sequences are allocated the same rate in the stable state. Once the CBR traffic joins the bottleneck, the streams react quickly and

$S_{1,2} \to R_{1,2}$	CREW, 4CIF @ 30Hz
$S_{3,4} \to R_{3,4}$	SOCCER, CIF @ 30Hz
$S_5 \to R_5$	SOCCER, QCIF @ 15Hz
$S_{6,7} \to R_{6,7}$	ICE, CIF @ 30Hz
$S_8 \to R_8$	ICE, CIF @ 30Hz (Priority 1)

TABLE I

DISTRIBUTION OF THE TEST VIDEO SEQUENCES AMONG THE

SENDER-RECEIVER PAIRS.

Scenario 1	Mean Y-PSNR at sender	Mean loss at receiver
$S_1 \to R_1$	33.30 dB	0.65 dB
$S_2 \to R_2$	33.28 dB	0.75 dB
$S_3 \to R_3$	36.62 dB	0.52 dB
$S_4 \to R_4$	36.56 dB	0.49 dB
Scenario 2		
$S_5 \to R_5$	36.12 dB	0.05 dB
$S_6 \to R_6$	41.06 dB	0.82 dB
$S_7 \to R_7$	41.24 dB	0.80 dB
Scenario 3		
$S_5 \to R_5$	34.01 dB	0.11 dB
$S_6 \to R_6$	39.99 dB	0.53 dB
$S_8 \to R_8$	41.92 dB	0.91 dB

TABLE II

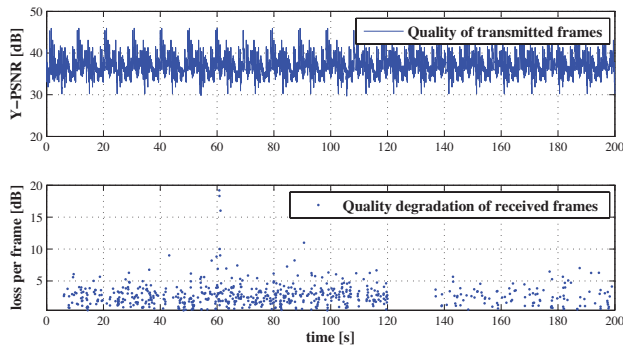PSNR QUALITY FOR THE DIFFERENT STREAMING SCENARIOS [DB].

Fig. 9.   Results for scenario 1. *Top:* Y-PSNR of each frame the SOCCER CIF@30Hz sequence transmitted by sender S_3. *Bottom:* Quality loss for each frame of the sequence, as received by client R_3.

settle in a new stable state almost immediately. Note that this behavior is very different from the sawtooth behavior seen in classical TCP connections, for example. As the CREW sequences carried by S_1 and S_2 have a lower utility gradient than the SOCCER sequences transmitted by S_3 and S_4, the rates of the latter are hardly affected by the drop in bottleneck capacity. In other words, as the background traffic lowers the bottleneck bandwidth, the distributed control system allocates less rate to the CREW streams, as this results only in a minor overall quality degradation. A rate reduction for the SOCCER streams would result in a larger quality drop. Once the background traffic is switched off, the four streams return rapidly to their original stable rates. The average quality of the streams sent by the senders are reported in Table II (Scenario 1) in terms of Y-PSNR, along with the respective quality reduction due to packet loss during the transmission.

Finally, we show the temporal evolution of the quality for one of the test streams in Figure 9. We report the Y-PSNR quality for each frame of the stream transmitted by S_3. We also compute the quality loss at the receiver by subtracting the Y-PSNR of each received frame from the Y-PSNR of the corresponding frame transmitted by the sender. Unsurprisingly, there is no loss when there is spare bandwidth available on the bottleneck link, i.e. during the first 10 seconds and after the CBR source switches off. There is however more packet loss due to congestion when the background traffic is active (i.e., in the timespan from 60 to 120 seconds). However, the steep gradient of the SOCCER Utility function prevents the rate from dropping. We note that loss cannot be completely avoided even in the steady state, as all the streams simultaneously compete for bandwidth shares until saturation of the bottleneck bandwidth. The small quality degradation resulting from these losses could be avoided by the use of error resilient coding methods.

C. Adaptation to new streams

In Scenario 2 we analyze the allocation of bandwidth resources when streams join and leave the bottleneck channel. We set the bottleneck bandwidth to 2 Mbps,
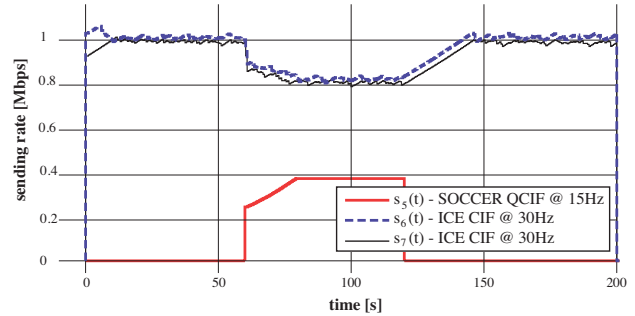

Fig. 10.   Resulting sending rates/controls for scenario 2.

and the sources S_6 and S_7 both start transmitting at time $t = 0$. After 60 seconds, the sender S_5 starts streaming during one minute, and then leaves the bottleneck again. The sending rates for this dynamic join/leave scenario are depicted in Figure 10.

It can again be seen that the system converges rapidly into a stable state, where the two equivalent streams get an equal share of the bottleneck capacity. Once the third source initiates its streaming sessions, the sending rates of the ICE sequences are gracefully decreased for S_6 and S_7 in order to adapt to the new situation. The new stream that contains the SOCCER QCIF sequence has a steep utility function and is therefore aggressive in getting shares of the bottleneck bandwidth. The quality of the transmitted streams and the corresponding quality drops due to packet loss are given in Table II (Scenario 2).

In order to illustrate the benefit offered by a slight increase in the transmission rate, we have run the same simulation where the senders however do not implement the guard interval presented in the previous section (see Figure 6). The corresponding sending rates are given in Figure 11. It can be seen that in this case the algorithm has a slower convergence to the steady state due to the delay introduced by late feedbacks. This penalizes the average quality by about 1 dB per stream with respect to the same simulation scenario where the guard interval is used.
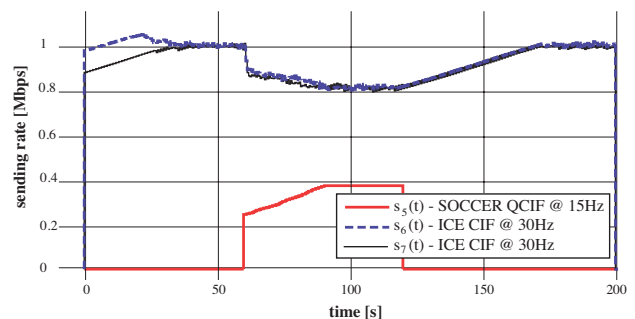

Fig. 11.   Resulting sending rates/controls for scenario 2 when the guard interval scheme is not used.

D. Adaptation to priority classes

As stated earlier, the concept of Utility functions is very general and extends beyond the classical characterization
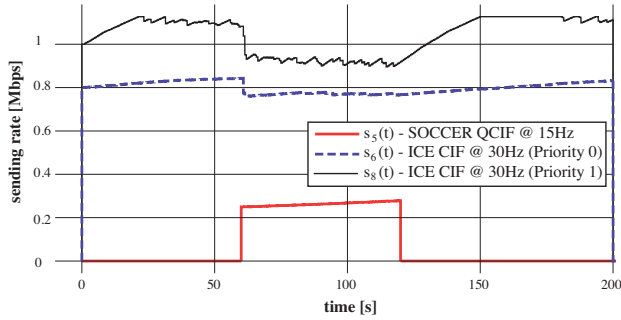
Fig. 12.   Resulting sending rates/controls for Scenario 3. Both S_6 and S_8 transmit the same ICE sequence, but using different Utility curves.

of video streams by their rate-distortion characteristics only. To illustrate this, we consider in Scenario 3 a situation similar to the previous one, but where two streams from servers S_6 and S_8 correspond to the same video content with different priority classes. They are characterized by the 2 Utility curves depicted in Figure 3, which could for example model two clients with differential treatment. The resulting sending rates for this simulation run are shown in Figure 12. They illustrate that the choice of Utility function directly drives the performance of the streaming application, which clearly favor the high priority clients. We report again in Table II (Scenario 3) the average quality of the transmitted streams along with the quality drops due to packet loss. We see that the high priority stream benefits from a 2dB quality gain compared to the low priority stream with the same video content.

Finally, we have run the same scenario several times using different values of π in the rate update Equation (12). Remember that this factor scales all the Utility gradients in the system and should thus bound the loss rate experienced by any stream in the stable state. The result of this simulation is given in Figure 13 where we show the experienced loss (which is equal to the congestion signal $p(t)$), as seen by receiver R_8 for the three cases where π equals 0.03, 0.05 and 0.1 respectively. As expected the packet loss are bounded by π in each of these cases. The parameter π is therefore an essential tool in order to adapt the rate allocation algorithm to a given decoder. A decoder can be characterized by a loss rate that it can cope with while decoding a stream, mostly through the use of error-concealment techniques. By matching π to the maximum tolerable loss rate at the decoder, we therefore ensure that the received stream is decodable when the system is in the stable state.

E. Adaptation to TCP traffic

We finally illustrate the behavior of the rate allocation algorithm when the bottleneck channel is shared with a flow controlled by TCP. We have considered a scenario similar to Scenario 1, but we replace the 4th stream in the system by an FTP data flow, which is regulated by TCP (Tahoe implementation). The results of this experiment
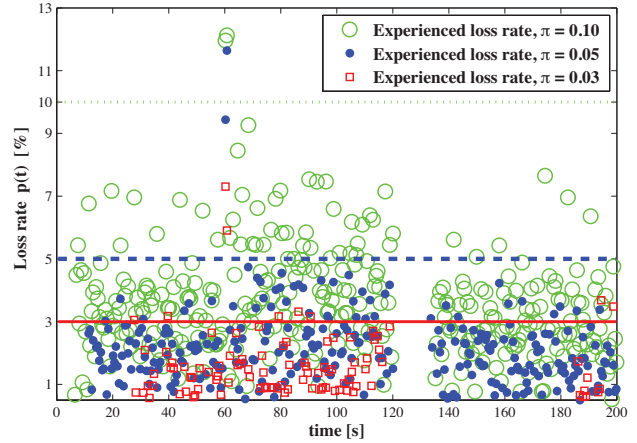

Fig. 13.   Results for scenario 3: experienced loss rates $p(t)$, expressed in percentages, at receiver R_8 for different values of π
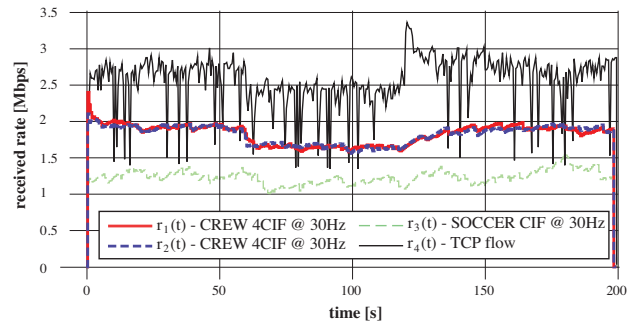

Fig. 14.   Received rate in Scenario 1, where the stream from S_4 is replaced by a TCP-regulated flow.

are shown in Figure 14, which presents the received rates at each of the clients. The goodput of the TCP flow is shown as stream 4. Interestingly, these results show that our proposed congestion control protocol can coexist with TCP on the same bottleneck channel. Compared to the Scenario 1, the sending rates of the media streams converge slower and show slightly larger oscillations around their stable state values. This is due to the rapid changes in available bandwidth, which is mostly driven by the TCP flow. However, one can observe that the rates are still smooth and that the CREW and SOCCER streams are still handled appropriately, according to their respective Utility functions.

While this simulation does not represent any formal proof of proper distribution of resources between TCP flows and the streams controlled by the algorithm proposed in this paper, it still shows that our algorithm does not starve TCP flows of the network resources. This corresponds to the results presented in [23], which state that the long term behavior of TCP Tahoe is equivalent to the one of a controller that maximizes a Utility function of the form $U(s) = \arctan(s)$. As this is a concave utility function, we expect TCP Tahoe streams to be able to compete with streams regulated by any other concave Utility functions in a stable system. However, the quality offered by the streams controlled by TCP is clearly sub-

optimal; TCP blindly optimizes the share of bandwidth resources, without considering the actual utility of the rate in terms of quality of experience.

Finally, we shall note that our algorithm is not *TCP-friendly* in the sense that the allocated rates yield an average per-flow bandwidth that is not equivalent to the one allocated by a TCP connection. TCP is in general more aggressive and tends to fairly share the average rate of each session, without considering the characteristics of each stream. Our algorithm targets a different objective, which is the effective allocation of the resources in order to maximize the aggregate utility, or equivalently to make the best use of bandwidth resources in terms of application requirements.

VII. CONCLUSIONS AND FUTURE WORK

We have presented a distributed rate allocation algorithm that targets the optimal distribution of bandwidth resources for improving the average quality of service of media streaming applications. We have first extended the framework of Network Utility Maximization with an important stability proof, which states that the algorithm is stable under general concave utility functions and randomly delayed feedback. This result is particularly important for the implementation of rate allocation algorithms in real scenarios. We have then proposed an effective and scalable implementation of the distributed control algorithm with a light-weight application-layer protocol. We have further proposed a few practical utility functions for streaming scalable video sequences. Finally, we have analyzed the behavior of the proposed solution with extensive NS-2 simulations, where we have considered H.264 SVC encoded sequences as an illustration. We have shown that the bandwidth allocation actually respects the constraints imposed by the utility functions and that the system converges quite rapidly to a stable state after changes in the network. The proposed algorithm therefore provides an interesting solution for rate allocation in distributed streaming systems. We plan to study the extension of the proposed framework to step-like utility functions rather than concave functions, in order to provide efficient control solutions for a larger variety of streaming applications.

ACKNOWLEDGMENT

REFERENCES

[1] F. P. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–27, 1997.

[2] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252(16), Mar 1998.

[3] V. Jacobson, "Congestion avoidance protocol," *ACM SIGCOMM*, pp. 314–329, 1988.

[4] M. H. S. Floyd and J. Padhye, "Equation-based congestion control for unicast applications," *Proceedings of ACM SIGCOMM*, pp. 43–56, Sep 2000.

[5] J. Yan, K. Katrinis, M. May, and B. Plattner, "Media- and tcp-friendly congestion control for scalable video streams," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 196–206, Apr 2006.

[6] Z. Cao and E. W. Zegura, "Utility max-min: an application-oriented bandwidth allocation scheme," *IEEE INFOCOM*, vol. 2, pp. 793–901, Mar 1999.

[7] M. Chiang, S. Zhang, and P. Hande, "Distributed rate allocation for inelastic flows: Optimization frameworks, optimality conditions, and optimal algorithms," *IEEE INFOCOM*, vol. 4, pp. 2679–2690, Mar 2005.

[8] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, 2000.

[9] W. H. Wang and S. H. Low, "Application-oriented flow control: Fundamentals, algorithms and fairness," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1282–1292, Dec 2006.

[10] Y. Zhang, S. R. Kang, and D. Loguinov, "Delayed stability and performance of distributed congestion control," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 307–318, Aug 2004.

[11] R. Johari and D. Tan, "End-to-end congestion control for the internet: Delays and stability," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 818–832, 2001.

[12] L. Massoulie and J. Roberts, "Bandwidth sharing: Objectives and algorithms," *IEEE INFOCOM*, vol. 3, pp. 1395–1403, Jun 1999.

[13] S. Kunniyur and R. Srikant, "End-to-end congestion control schemes: Utility functions, random losses and ecn marks," *IEEE INFOCOM*, vol. 3, pp. 1323–1332, 2000.

[14] P. Ranjan, R. J. La, and E. H. Abed, "Global stability conditions for rate control with arbitrary communication delays," *IEEE/ACM Transactions on Networking*, vol. 14, no. 1, pp. 94–107, Feb 2006.

[15] R. J. La and V. Anantharam, "Utility-based rate control in the internet for elastic traffic," *IEEE/ACM Transactions on Networking*, vol. 10, no. 2, pp. 272–286, Apr 2002.

[16] Y. Zhang and D. Loguinov, "Local and global stability of symmetric heterogeneously-delayed control systems," *IEEE Conference on Decision and Control (CDC)*, 2004.

[17] T. Alpcan and T. Basar, "A globally stable adaptive congestion control scheme for internet-style networks with delay," *IEEE/ACM Transactions on Networking*, vol. 13, no. 6, pp. 1261–1274, Dec 2005.

[18] J.-W. Lee, R. R. Mazumdar, and N. B. Shroff, "Non-convex optimization and rate control for multi-class services in the internet," *IEEE/ACM Transactions on Networking*, vol. 13, no. 4, pp. 827–840, Aug 2005.

[19] L. Ying, E. Dullerud, and R. Srikant, "Global stability of internet congestion controllers with heterogeneous delays," *American Control Conference*, vol. 4, pp. 2948–2953, Jun 2004.

[20] L. Massoulie, "Stability of distributed congestion control with heterogeneous feedback delays," *IEEE Transactions on Automatic Control*, vol. 47, no. 6, pp. 895–902, Jun 2002.

[21] Y. Zhang, S. R. Kang, and D. Loguinov, "Delay-independent stability and performance of distributed congestion control," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, Dec 2007.

[22] M. Dai, D. Loguinov, and H. M. Radha, "Rate-distortion analysis and quality control in scalable internet streaming," *IEEE Transactions on Multimedia*, vol. 8, no. 6, pp. 1135–1146, Dec 2006.

[23] J. He, M. Chiang, and J. Rexford, "Can congestion control and traffic engineering be at odds?" *Proceedings of the Global Telecommunications Conference. GLOBECOM '06*, pp. 1–6, 2006.

Jean-Paul Wagner received his M.Sc. in Communication Systems as well as his Ph.D. from the Ecole Polytechnique Fédérale de Lausanne (EPFL) in 2004 and 2008 respectively. His main research topics have been in the domain of distributed streaming of scalable multimedia content. He is now working for a firm of intellectual property attorneys in Luxembourg.

Pascal Frossard received the M.S. and Ph.D. degrees, both in electrical engineering, from the Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, in 1997 and 2000, respectively. Between 2001 and 2003, he was a member of the research staff at the IBM T. J. Watson Research Center, Yorktown Heights, NY, where he worked on media coding and streaming technologies. Since 2003, he has been a faculty at EPFL, where he heads the Signal Processing Laboratory (LTS4). His research interests include image representation and coding, visual information analysis, distributed image processing and communications, and media streaming systems.

Dr. Frossard has been the General Chair of IEEE ICME 2002 and Packet Video 2007. He has been the Technical Program Chair of EUSIPCO 2008, and a member of the organizing or technical program committees of numerous conferences. He has been an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA (2004-), the IEEE TRANSACTIONS ON IMAGE PROCESSING (2010-) and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (2006-). He is an elected member of the IEEE Image and Multidimensional Signal Processing Technical Committee (2007-), the IEEE Visual Signal Processing and Communications Technical Committee (2006-), and the IEEE Multimedia Systems and Applications Technical Committee (2005-). He has served as Vice-Chair of the IEEE Multimedia Communications Technical Committee (2004-2006) and as a member of the IEEE Multimedia Signal Processing Technical Committee (2004-2007). He received the Swiss NSF Professorship Award in 2003, the IBM Faculty Award in 2005, the IBM Exploratory Stream Analytics Innovation Award in 2008 and the IEEE Transactions on Multimedia Best Paper Award in 2011.