Available online at www.sciencedirect.com
ScienceDirect
Computer Networks 51 (2007) 4377–4395
Computer Networks
www.elsevier.com/locate/comnet

SSVP: A congestion control scheme for real-time video streaming

Panagiotis Papadimitriou *, Vassilis Tsaoussidis

Democritus University of Thrace, Electrical and Computer Engineering Department, 12 Vas. Sofias Street, Xanthi, Greece

Received 1 September 2006; received in revised form 1 May 2007; accepted 13 June 2007
Available online 29 June 2007

Responsible Editor: Buyurman Baykal

Abstract

In this paper, we present a new end-to-end protocol, namely *Scalable Streaming Video Protocol* (SSVP), which operates on top of UDP and is optimized for unicast video streaming applications. SSVP employs *Additive Increase Multiplicative Decrease* (AIMD)-based congestion control and adapts the sending rate by properly adjusting the inter-packet-gap (IPG). The smoothness-oriented modulation of AIMD parameters and IPG adjustments reduce the magnitude of AIMD oscillation and allow for smooth transmission patterns, while TCP-friendliness is maintained. Our experimental results demonstrate that SSVP eventually adapts to the vagaries of the network and achieves remarkable performance on real-time video delivery. In the event where awkward network conditions impair the perceptual video quality, we investigate the potential improvement via a layered adaptation mechanism that utilizes receiver buffering and adapts video quality along with long-term variations in the available bandwidth. The adaptation mechanism sends a new layer based on explicit criteria that consider both the available bandwidth and the amount of buffering at the receiver, preventing wasteful layer changes that have an adverse effect on user-perceived quality. Quantifying the interactions of SSVP with the specific adaptation scheme, we identify notable gains in terms of video delivery, especially in the presence of limited bandwidth.

Keywords: End-to-end protocols; Congestion control; Quality of Service; Video streaming

1. Introduction

Time-sensitive applications, such as streaming media, gain popularity and real-time data is expected to compose a considerable portion of the overall data traffic traversing the Internet. These applications generally prefer timeliness to reliability. Real-time video streaming, in particular, calls for strict requirements on end-to-end delay and delay variation. Furthermore, reliability parameters, such as packet loss and bit errors, usually compose an impairment factor, since they cause perceptible degradation on video quality. Unlike bulk-data transfers, video streaming seeks to achieve smooth playback quality rather than simply transmit at the highest attainable bandwidth.

* Corresponding author. Tel.: +30 25410 84382.
E-mail addresses: ppapadim@ee.duth.gr (P. Papadimitriou), vtsaousi@ee.duth.gr (V. Tsaoussidis).

1389-1286/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2007.06.018

Such stringent requirements necessitate explicit management techniques in order to preserve the fundamental *Quality of Service* (QoS) guarantees for video traffic. In this context, *Internet Engineering Task Force* (IETF) attempted to facilitate true end-to-end QoS on IP networks by defining *Integrated* (*IntServ*) and *Differentiated Services* (*DiffServ*) models [2,19]. IntServ follows the signaled-QoS model, where the end-hosts signal their QoS need to the network, while DiffServ works on the provisioned-QoS model, where network elements are set up to service multiple classes of traffic with varying QoS requirements. However, both models are associated with high implementation costs and limited applicability; hence, they have not yet received wide appeal from the majority of users. Essentially, most end-users still rely on the best-effort services of the Internet which strives to meet the high demands of the emerging multimedia applications.

Today's Internet is governed by the rules of *Additive Increase Multiplicative Decrease* (AIMD) [5], which effectively contribute to its stability. Essentially, the goal of such algorithms is to prevent applications from either overloading or under-utilizing the available network resources. Although *Transmission Control Protocol* (TCP) provides reliable and efficient services for bulk-data transfers, several design issues render the protocol unsuitable for time-sensitive applications. More precisely, the process of probing for bandwidth and reacting to observed congestion causes oscillations to the achievable transmission rate. With TCP's increase-by-one and decrease-by-half control strategy, even an adaptive and scalable source coding scheme is not able to conceal the flow throughput variation. Furthermore, TCP occasionally introduces arbitrary delays, since it enforces reliability and in-order delivery. In response to standard TCP's limitations, several TCP protocol extensions [1,9] have emerged providing more efficient bandwidth utilization and sophisticated mechanisms for congestion control. *TCP-friendly* protocols, presented in [9,23,24], achieve smooth window adjustments, while they manage to compete fairly with TCP flows. In order to achieve smoothness, they use gentle backward adjustments upon congestion. In [21,26] we showed that this modification has a negative impact on responsiveness.

User Datagram Protocol (UDP) has been widely used instead of TCP by real-time applications, since it allows for transmission attempts at application rate and consequently, induces minimal fluctuations in the transmission rate. However, UDP poses a threat to network stability, as it lacks all basic mechanisms for flow/congestion control. Furthermore, as the success of the Internet primarily relies on self-regulated TCP, it is crucial to enforce compatible traffic regulations for non-TCP flows. In this context, Internetworking functionality evolves towards punishing free-transmitting protocols.

Congestion control algorithms are, therefore, necessary for multimedia applications in order to deal with the diverse and constantly changing conditions of the Internet. An overview of Internet's current congestion control paradigm reveals that routers play a relatively passive role: they merely indicate congestion through packet drops or *Explicit Congestion Notification* (ECN). It is the end-systems that perform the crucial role of responding appropriately to these congestion signals. Numerous video streaming applications have implemented their own congestion control mechanisms, usually on a case-by-case basis on top of UDP. However, implementing application-level congestion control is difficult and not part of most applications' core needs.

Time-sensitive application constraints and the limitations of existing congestion control schemes circumscribe a framework for potential improvements. We hereby identify distinct cases that motivate end-to-end protocol design for real-time traffic, especially if efficiency is considered on the basis of the application requirements:

TCP's insistence on reliable delivery without timing considerations has an adverse effect on the performance of the system, especially for time-sensitive applications where data packets bear information with a limited useful lifetime.

Multiplicative decrease with a factor of 1/2 (e.g. TCP, Rate Adaptation Protocol [16]) causes transmission gaps that hurt the performance of real-time applications, which experience jitter and degraded throughput.

Slow-responding TCP-friendly protocols [23,24] yield sufficient performance in stationary environments. However, responsiveness is critical for the Internet which operates in the transient than in the stationary regime.

TCP-Friendly Rate Control (TFRC) [9] arguably enables smoother delivery than TCP; however, TFRC's throughput model is quite sensitive to parameters (e.g. packet loss rate, Round-Trip Time), which are often difficult to measure efficiently and to predict accurately. TFRC is

also less responsive to short-term network and session dynamics. In addition, TFRC and TCP flows may have substantially different average sending rates, when they share common network resources. This long-term throughput imbalance between the two protocols causes coexisting TCP and TFRC flows to experience different loss rates, amplifying the throughput difference. Authors in [17] provide an analytical study of the limitations of equation-based congestion control and particularly TFRC.

Source-based decision on the transmission rate, based on the pace of the acknowledgments (e.g. TCP Westwood [13]), inevitably incorporates the potentially asymmetric characteristics of the reverse path. Thus, bandwidth asymmetry may result in inaccurate bandwidth estimates, confining protocol efficiency and subsequently application performance.

Following these observations, there is still room for emerging end-to-end protocols for time-sensitive applications, especially if they incarnate attractive properties, such as unreliable delivery with built-in congestion control. In this context, we have been working on a congestion control mechanism to adapt the rate of outgoing video streams to the characteristics of the end-to-end network path. We had the option to rely on the unreliable UDP datagrams or modify TCP to provide unreliable semantics. However, the latter seems particularly inappropriate considering the TCP semantics and its reliance on cumulative acknowledgments. Consequently, we considered UDP as a better choice, due to its unreliable and out-of-order delivery. Along these lines, we designed a new congestion control scheme, namely *Scalable Streaming Video Protocol* (SSVP), which is optimized for unicast streaming video applications. We note that SSVP congestion control is purely end-to-end and does not rely on QoS functionality in routers, such as *Random Early Drop* (RED), *ECN* or other *Active Queue Management* mechanisms.

SSVP is designed to provide efficient and smooth rate control, while maintaining friendliness with corporate flows. The protocol has the following salient attributes:

It operates on top of the light-weight UDP which is already preferred by the majority of streaming applications and Internet telephony. High-bandwidth UDP applications may rely on SSVP, avoiding the difficult task of implementing congestion control by themselves.

It employs AIMD-oriented congestion control by adjusting the inter-packet-gap (IPG), depending on the occurrence of congestion. IPG adjustments generate a smooth data flow by spreading the data transmission across a time interval, avoiding the burstiness induced by window-based mechanisms, such as TCP.

It provides an unreliable transport service, as retransmissions are often a wasted effort for time-sensitive traffic: they usually deliver delayed packets which are either discarded, or at the worst they obstruct the proper reconstruction of incoming packets.

SSVP segments include header information which allows the manipulation of outgoing video streams (e.g. frame prioritization).

We note that the rationale behind the AIMD adoption is to establish a friendly behavior for SSVP. AIMD flows are the Internet's main bandwidth consumer (which is not expected to change drastically in the near future), and the safest way to achieve inter-protocol friendliness is to employ an AIMD mechanism. Recent alternatives, including TFRC and delay-based congestion control (e.g. TCP Vegas [3], FAST TCP [22]), allow for smoother transmission patterns than AIMD. Besides the aforementioned TFRC's limitations, using delay as a measure of congestion may cause undesirable effects in terms of bandwidth utilization and network stability, especially if it is not augmented with loss information. For example, dominating TCP flows may cause such fluctuations that delay-based congestion avoidance cannot measure the prevailing network conditions on time. Furthermore, the presence of many short flows may result in potential starvation of competing long-lived Vegas/FAST TCP flows.

SSVP composes a viable alternative to existing congestion control schemes, in the context of media delivery with timing considerations. Despite AIMD's oscillatory nature, SSVP manages to reduce the variations in the sending rate, as the combined effect of the following: (i) the protocol enforces a smoothness-oriented modulation of the additive increase and multiplicative decrease factors, while at the same time maintains the required TCP-friendliness, (ii) the transmission rate is adapted once per *Round-Trip Time* (RTT), and (iii) the SSVP sender invokes IPG adjustments spacing outgoing packets evenly.

Besides the detailed description of protocol mechanisms and specification, we exploit a receiver-buffered layered adaptation scheme to adjust the quality of congestion-controlled video on-the-fly. Receiver buffering reduces jitter, and depending on the amount of buffered data, the receiver is enabled to sustain temporary drops in the sending rate. In order to prevent wasteful layer changes which impair the perceptual video quality, the mechanism delivers additional layers based on certain criteria that consider the available bandwidth and the amount of buffered data. The layered scheme is designed to effectively interact with AIMD mechanisms: layered encoding allows video quality adjustments over long periods of time, whereas AIMD congestion control adjusts the transmission rate rapidly over short-time intervals. We quantify the efficiency of the receiver-buffered scheme focusing on the interactions between layered adaptation and SSVP congestion control. Our simulations reveal that this combined approach alleviates most of the impairments induced by limited bandwidth and transient errors.

The remainder of the paper is organized as follows. Section 2 provides an overview of related work and congestion control mechanisms for time-sensitive applications. In Section 3 we discuss the design and implementation details of the proposed congestion control scheme. We also study the relation between the additive increase rate α and multiplicative decrease ratio β, which can be translated into the additive and multiplicative parameters to adjust IPG. In Section 4 we elaborate on the receiver-buffered layered scheme. Section 5 includes our evaluation methodology followed by Section 6, where we provide extensive performance studies of our mechanisms based on simulations. Finally, Section 7 concludes the paper and refers to future work.

2. An overview of related work and congestion control schemes for time- sensitive traffic

The literature includes numerous studies and proposals towards efficient congestion control for time-sensitive applications in the Internet. *Rate Adaptation Protocol* (RAP) [16] is a rate-based protocol which employs an AIMD algorithm for the transmission of real-time streams. The sending rate is continuously adjusted by RAP in a TCP-friendly fashion, using feedback from the receiver. However, since RAP employs TCP's congestion control parameters (i.e. 1, 0.5), it causes short-term rate oscillations, primarily due to the multiplicative decrease. In [15] the specific protocol exploits layered encoding and the corresponding component, namely layer manager, tries to deliver the maximum number of layers that can fit in the available bandwidth. Rate adaptation takes place on a timescale of round-trip times, but layers are added and dropped on a longer timescale.

Datagram Congestion Control Protocol (DCCP) [12] is a new transport protocol that provides a congestion-controlled flow of unreliable datagrams. DCCP is intended for delay-sensitive applications which have relaxed packet loss requirements. The protocol aims to add to a UDP-like foundation the minimum mechanisms necessary to support congestion control. DCCP provides the application with a choice of congestion control mechanisms via *Congestion Control IDs* (CCIDs), which explicitly name standardized congestion control mechanisms. Currently, two CCIDs have been developed, supporting TCP-like and TFRC congestion control. We point out that SSVP composes an experimental congestion control scheme which is not directly comparable with DCCP. Although both congestion control mechanisms emerge from a common incentive, DCCP constitutes a generalized framework for delay-sensitive data transport, while SSVP explicitly addresses real-time video delivery.

Authors in [7] analyze the impact of selected congestion control algorithms on the performance of streaming video delivery. They concentrate on binomial congestion control [1] and especially on SQRT, which responds to packet drops by reducing the congestion window size proportional to the square root of its value instead of halving it. In [6] a *Real-Time Transport* (RTP) [18] compatible protocol (i.e. *SR-RTP*) is proposed, which adaptively delivers high quality video in the face of packet loss. SR-RTP enables selective reliability, retransmitting only the important data.

Since TCP is rarely chosen to transport real-time traffic over the Internet, TCP-friendly protocols constitute an elegant framework for multimedia applications. We consider as TCP-friendly any protocol whose long-term arrival rate does not exceed the one of any conformant TCP in the same circumstances [8]. The differences between standard TCP and TCP-friendly congestion control lie mainly in the specific values of α and β, while their similarities in their AIMD-based congestion control. Standard TCP is therefore viewed as a specific case of AIMD

(α, β) with $\alpha = 1$ and $\beta = 0.5$. Some TCP-friendly protocols (e.g. GAIMD [24]) are designed to satisfy the requirements of delay-sensitive applications. However, they may exhibit further weaknesses, when bandwidth becomes available rapidly [21]. Apparently, the tradeoff between responsiveness and smoothness can be controlled to favor some applications, but it may cause some other damages. The choice of parameters α and β has a direct impact on the responsiveness of the protocols to conditions of increasing contention or bandwidth availability.

TFRC [9] is a representative TCP-friendly protocol, which adjusts its transmission rate in response to the level of congestion, as estimated based on the calculated loss rate. Multiple packet drops in the same RTT are considered as a single loss event and hence, the protocol follows a more gentle congestion control strategy. More precisely, the TFRC sender uses the following TCP response function:

$$T(p, \mathrm{RTT}, \mathrm{RTO}) = \frac{1}{\mathrm{RTT}\sqrt{\frac{2p}{3}} + \mathrm{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p(1+32p^2)}, \quad (1)$$

where p is the steady-state loss event rate and *RTO* is the retransmission timeout value. Eq. (1) enforces an upper bound on the sending rate T. According to [9], TFRC's increase rate never exceeds 0.14 packets per RTT (or 0.28 packets per RTT when history discounting has been invoked). In addition, the protocol requires 5 RTTs in order to halve its sending rate. Consequently, the instantaneous throughput of TFRC has a much lower variation over time. However, this smoothness has a negative impact, as the protocol becomes less responsive to bandwidth availability [21]. MULTFRC [4] is a recent extension to TFRC for wireless networks, establishing multiple TFRC connections on the same path when a single connection is not able to utilize the wireless resources efficiently.

TCP-Real [20,25] is a high-throughput transport protocol that incorporates a congestion avoidance mechanism in order to minimize transmission-rate gaps. As a result, the protocol is suited for real-time applications, since it enables better performance and reasonable playback timers. TCP-Real approximates a receiver-oriented approach beyond the balancing trade of the parameters of additive increase and multiplicative decrease. The protocol introduces another parameter, namely γ, which determines the window adjustments during congestion avoidance. More precisely, the receiver measures the receiving rate and attaches the result to its acknowledgments (ACKs), directing the transmission rate of the sender. When new data is acknowledged and the congestion window (*cwnd*) is adjusted, the current data-receiving rate is compared against the previous one. If there is no receiving rate decrease, *cwnd* is increased by 1 *Maximum Segment Size* every *RTT*. If the magnitude of the decrease is small, the *cwnd* remains temporarily unaffected; otherwise, the sender reduces the *cwnd* multiplicatively by γ. In [25] a default value of $\gamma = 1/8$ is suggested. However, this parameter can be adaptive to the detected conditions. Generally, TCP-Real can be viewed as a TCP (α, β, γ) protocol, where γ captures the protocol's behavior prior to congestion when congestion boosts up.

TCP Westwood [13] is a transport protocol that emerged as a sender-side-only modification of TCP Reno congestion control. TCP Westwood exploits end-to-end bandwidth estimation in order to adjust the values of slow-start threshold and *cwnd* after a congestion episode. The protocol incorporates a recovery mechanism which avoids the blind halving of the sending rate of TCP Reno after packet loss and enables TCP Westwood to achieve high link utilization in the presence of wireless errors. The specific mechanism considers the sequence of bandwidth samples *sample_BWE*[n] obtained using the *ACKs* arrival and evaluates a smoothed value, *BWE*[n], by low-pass filtering the sequence of samples, as described by the following pseudocode:

Algorithm 1. TCP-Westwood

If an ACK is received **then**
  set sample_BWE[n] = (acked * pkt_size * 8)/(now − last_ACK_time)
  set BWE[n] = (1 − beta) * (sample_BWE[n] + sample_BWE[$n-1$])/2 + beta * BWE[$n-1$]
end if

where *acked* is the number of segments acknowledged by the last *ACK*; *pkt_size* is the segment size in bytes; *now* is the current time; *last_ACK_time* is the time the previous *ACK* was received; *beta* is the pole used for the filtering (a value of 19/21 is suggested). However, in [14] we showed that TCP Westwood tends to underestimate the available bandwidth, due to ACKs clustering. *TCP Westwood+* is a recent extension of TCP Westwood,

based on the *Additive Increase/Adaptive Decrease* (AIAD) mechanism. Unlike the initial version of Westwood, TCP Westwood+ computes one sample of available bandwidth every RTT, using all data acknowledged in the specific RTT [10].

3. SSVP design and implementation

3.1. Sender and receiver interaction

SSVP, in a complementary role, operates on top of UDP and supports end-to-end congestion control relying on sender and receiver interaction. SSVP acknowledges the datagrams received by transmitting control packets (containing no data). In accordance with the relaxed packet loss requirements of streaming video and considering the delays induced by retransmitted packets, SSVP does not integrate reliability into UDP datagrams. Hence, control packets do not trigger retransmissions. However, they are effectively used in order to determine bandwidth and RTT estimates, and properly adjust the rate of the transmitted video stream.

We have encapsulated additional header information to UDP datagrams[1] (Fig. 1), including packet type, sequence number, packet length, frame type, congestion indicator (CIn) and timestamp. *Packet type* field denotes whether a segment with video-data or a control packet is transmitted. *Frame type* can be exploited in order to augment a prioritized transmission (where I frames can be prioritized). *CIn* field is used by control packets as a congestion indicator (i.e. marked with 1 if packet loss is detected). Although SSVP currently provides one bit of information for the state of congestion, the specific field accommodates two bits in the protocol header to allow for possible future extensions.

Timestamp field is used to handle RTT computation. More precisely, when the sender transmits a video-packet, it updates the specific field with current time Ts_n. As soon as the receiver acquires the packet, it generates a control packet attaching Ts_n to the *timestamp* field. Upon the receipt of the corresponding feedback, the sender subtracts the included *timestamp* from current time in order to estimate the RTT sample. If Ts'_n denotes the time

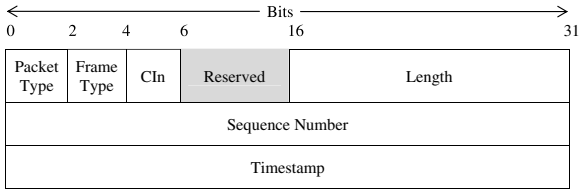

Fig. 1. SSVP header.

the packet is being received, the sender gets the observed value of each RTT, as follows:

$$\text{SampleRTT} = Ts'_n - Ts_n. \tag{2}$$

Therefore, SSVP obtains an accurate approximation of RTT, which does not require synchronization between sender's and receiver's clocks.

Since SSVP is a protocol without reliability, some datagrams may be lost due to congestion or inability of the receiving host from reading the packets rapidly enough. The receiver uses packet drops or reordering as congestion indicator. Consequently, congestion control is triggered when:

> a packet is received carrying a sequence number greater than the expected sequence number and the receiver does not acquire any packets within a timeout interval.

The flowchart of SSVP receiver's responses to the above events is illustrated in Fig. 2.

The proper adjustment of the timeout interval is critical. A timeout interval that is set too short will claim false packet drops resulting in a wasteful reduction of the transmission rate. On the other hand, a long and consequently conservative timeout interval will inevitably impact the protocol responsiveness. In order to properly adjust the timeout, we exploit RTT measurements (*SampleRTT*) and based on this quantity we further compute the weighted average of RTT:

$$\text{EstimatedRTT} = \gamma \times \text{EstimatedRTT} + (1-\gamma) \times \text{SampleRTT} \tag{3}$$

setting the smoothing factor γ to 0.9. After RTT estimation, the timeout interval for SSVP (STO) can be calculated by:

$$\text{STO} = \text{EstimatedRTT} + \delta \times \text{Deviation}, \tag{4}$$

where δ is set to 4 and *Deviation* is the smoothed estimation of the variation of RTT which is represented as:

[1] We do not use RTP avoiding its overhead. SSVP header includes all the necessary fields (i.e. sequence number, timestamp) to add RTP functionality and enable congestion control.

$$\text{Deviation}_n = \varepsilon \times \text{Deviation}_{n-1} + (1-\varepsilon) \times |\text{EstimatedRTT} - \text{EstimatedRTT}'|, \quad (5)$$

where $Deviation_{n-1}$ and *EstimatedRTT* are the variation of RTT and the estimated RTT in the last round respectively, while ε is set to 0.25.

Since STO is calculated on the sender, the receiver needs to acquire this estimate. In order to avoid additional overhead, STO is periodically communicated to the receiver via the *Timestamp* field. Hence, the receiver maintains the most recent value of STO. When the sender attaches STO to an outgoing packet, it marks a specific bit in the *reserved* space of the protocol header. The receiver is therefore able to detect whether an incoming packet carries a timestamp or STO. We note that SSVP estimates the timeout value similarly to TCP to prevent potential misbehaviors when SSVP coexists with TCP flows. For example, [17] uncovers that a main reason for the long-term throughput imbalance between competing TCP and TFRC flows is the different timeout estimation schemes between the two protocols.

3.2. Rate adjustment

SSVP adjusts the sending rate in a TCP-friendly fashion, exploiting the feedback of reception statistics (control packets). Both *binomial* [1] and AIMD congestion control are designed to achieve TCP-friendliness. Although binomial schemes, such as IIAD or SQRT, are quite attractive to multimedia applications for their smooth rate variations, they are not able to achieve TCP-friendliness independent of link capacity. Apart from link capacity, the selection of increase rate and decrease ratio composes another influencing parameter. Along these lines, in order to attain TCP-friendliness, SSVP incorporates AIMD congestion control. Let α, β the values of additive increase rate and multiplicative decrease ratio, respectively. The choice of α and β has a direct impact on protocol responsiveness to conditions of increasing contention or bandwidth availability. In highly multiplexed dynamic networks, AIMD flows with different (α, β) pairs have different response patterns to transient changes of network resources. For example, an AIMD flow

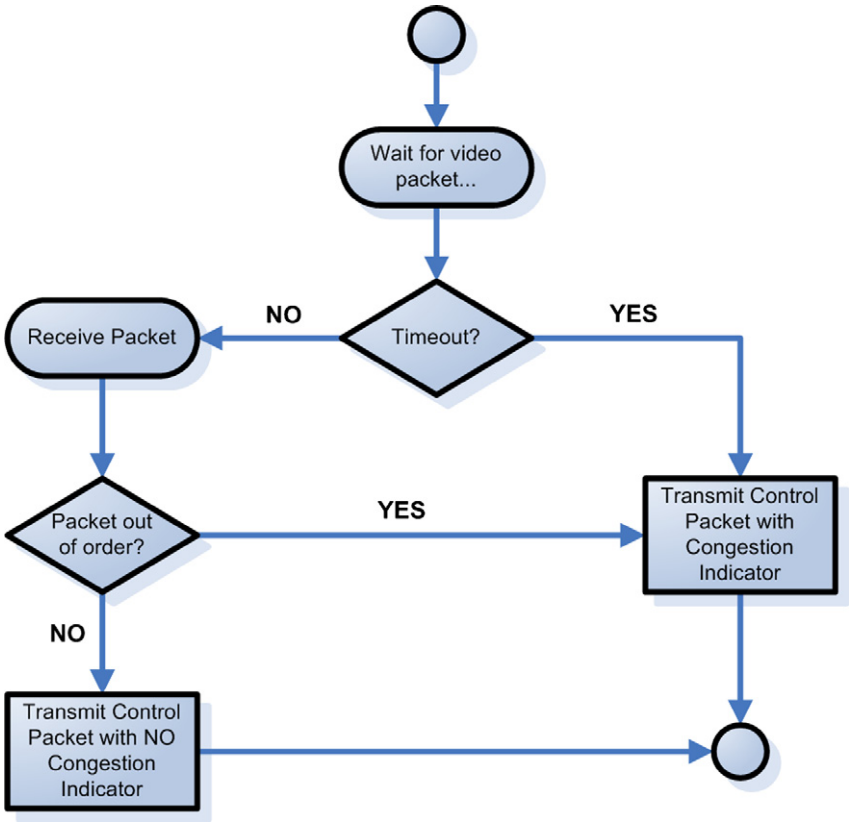

Fig. 2. SSVP receiver flowchart.

with a large α and small β is very sensitive to bandwidth variation, and consequently its instantaneous throughput changes rapidly. Such behavior may cause frequent interruptions on video playback. Generally, an AIMD protocol with a large β is more suitable for such applications, as we show in the sequel.

Fig. 3 illustrates an SSVP flow in the presence of a scalable video coder. Assuming a packet drop at time t_1, the transmission rate is reduced from R to βR and immediately the video coder is notified to reduce the video coding rate. This process (i.e. coding rate reduction) inevitably incurs a delay d, and eventually transmission resumes at time $t_2 = t_1 + d$. We investigate the maximum sending delay with respect to the streaming application requirements. Let a packet P generated at time t_2. P will be enqueued after a number of packets that were generated during the time period: $t_1 \leqslant t_s < t_2$. Consequently, such a packet will suffer the longest delay. During t_s, the queue inside the sender is increased with the amount of data obtained by the area of the shaded rectangular in Fig. 3, i.e. $d(1-\beta)R$. Hence, the sending delay D that denotes the amount of time that packet P will rest inside the server is derived by

$$D \approx \frac{d(1-\beta)R}{\beta R} = d\left(\frac{1}{\beta} - 1\right). \tag{6}$$

Assuming a fixed value of d (for a certain video coder), sending delay D exclusively depends on the decrease parameter β. With respect to Eq. (6), we can enhance video delivery by choosing a suitable value of β. A high decrease ratio can reduce the sending delays along with the magnitude of AIMD oscillation, inline with the requirements of media-streaming applications for smooth patterns of data transmission. However, a large β enforces the selection of a small α, according to the TCP-friendly condition obtained in [24]

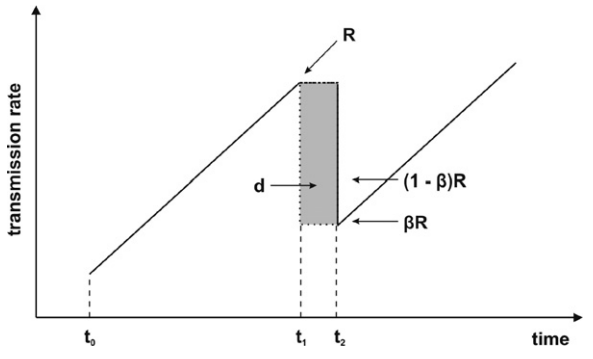

Fig. 3. SSVP transmission rate evolution.

$$\alpha = \frac{4(1-\beta^2)}{3}. \tag{7}$$

Based on this intuitive analysis and with respect to the perceptual video quality, SSVP employs AIMD congestion control with $\alpha = 0.31$ and $\beta = 0.875$. The transmission rate is controlled by properly adjusting IPG. Let S denote the packet length, the instantaneous transmission rate R_i for an SSVP flow is given by

$$R_i = \frac{S}{t_i + \mathrm{IPG}_i} \approx \frac{S}{\mathrm{IPG}_i} \tag{8}$$

if we consider the transmission time t_i of the ith packet negligible (compared to IPG). The sender adjusts the transmission rate once per RTT in order to maintain a smoothed flow. More precisely, each RTT the SSVP source calculates the ratio of the number of control packets with congestion indicator (i.e. *CIn* field 1) over the total number of incoming control packets in order to estimate the level of congestion and subsequently follow the appropriate recovery strategy. If this ratio exceeds a specific *congestion level* threshold, the sender infers congestion and immediately reduces the transmission rate via the multiplicative increase of IPG:

$$\mathrm{IPG}_{i+1} = \frac{\mathrm{IPG}_i}{\beta}. \tag{9}$$

If the sender has received at least one control packet with congestion indicator but the measured ratio does not exceed the *congestion level* threshold, a transient loss (e.g. wireless error) is assumed and the sending rate remains unaffected. The reception of control packets with no congestion indicator within an RTT triggers an increase in the transmission rate by decreasing IPG, as follows:

$$\mathrm{IPG}_{i+1} = \frac{1}{1+\alpha}\mathrm{IPG}_i. \tag{10}$$

We note that the sender-side measurements of the congestion level are not affected by the spacing between the incoming control packets. Furthermore, possible control packet losses do not affect the protocol behavior dramatically; the sender will still be able to calculate the level of congestion. Essentially, the combination of the selected AIMD parameters and IPG adjustments reduces the magnitude of AIMD oscillation and eventually allows

for a smooth transmission pattern that favors real-time applications. The *congestion level* threshold has been set experimentally to 0.005. Certainly, this can be adjusted differently in order to modify the transient behavior of SSVP congestion control. The detailed operation of the SSVP sender is described by the following pseudocode:

Algorithm 2. SSVP Sender-Side Operation

```
if the sender receives a Control Packet then
    increase acks_within_rtt
    if the Control Packet's CIn field is 1 then
        increase congestion_acks_within_rtt
    end if
    if (current_time − adjustment_time) ⩾ current_RTT then
        if congestion_acks_within_rtt / acks_within_rtt > congestion_level_threshold then
            set IPG_{i+1} = IPG_i / β
        end if
        if congestion_acks_within_rtt = 0 then
            set IPG_{i+1} = 1/(1+α) IPG_i
        end if
        set congestion_acks_within_rtt = 0
        set acks_within_rtt = 0
        set adjustment_time = current_time
    end if
end if
```

where *adjustment_time* is the time of the previous rate adjustment (initially set to 0); *current_time* is the running time; *current_rtt* is the last measured RTT; *acks_within_rtt* is the number of control packets received within current RTT; *congestion_acks_within_rtt* is the number of control packets with congestion indicator that were received within current RTT.

During startup, it is crucial for any protocol to explore the available bandwidth rapidly. Increasing the transmission rate once per RTT (with the protocol's standard increase rate) may result in poor startup utilization. In order to overcome this limitation the SSVP sender increases the sending rate, based on Eq. (10), after the reception of each control packet. As a result, multiple rate increases may occur within a single RTT, improving bandwidth utilization during startup. On the occurrence of congestion, the startup phase is terminated and the SSVP sender behaves as described in Algorithm 2.

We note that the similarities between SSVP and RAP lie only in the adjustments of IPG, since both compose spacing-based schemes. In contrast to SSVP, the RAP source receives ACKs infrequently and exploits the redundant information on a single incoming ACK to detect packet loss, inline with TCP's *Fast Recovery* algorithm. Generally, RAP attempts to resemble TCP's functionality, leaving out only the undesired reliability. However, some aspects of TCP design that do not favor real-time delivery are incorporated into RAP. For example, the multiplicative decrease by a factor of 1/2 invokes abrupt rate reductions upon congestion, compromising smoothness. Furthermore, RAP is part of an end-to-end architecture designed specifically for layered video streams. On the contrary, SSVP is decoupled from application-level functionality and operates independently on top of UDP. Certainly, SSVP can be augmented by video adaptation schemes, such as layered encoding. We elaborate on this combined approach in the following section.

4. Layered adaptation

The rationale of the employed adaptation scheme mainly rests on the assumption that a user's perception is sensitive to the changes in video quality and potential interruptions in the stream playback. Despite the degradation in visual quality, we consider smooth video of reduced bitrate more preferable than inconsistent and jerky video at highest quality. Since processing on the original pixels is computationally expensive (full decoding and encoding of the video stream is required), various scalability techniques have been proposed [15], which operate on-the-fly. *Simulcast* uses multiple versions of the stream, encoded at different bitrates. The versions of streams used are often limited in order to avoid high redundancy. The server transmits all the alternate streams and the client switches to a stream version depending on the available bandwidth. *Layered adaptation* has been proposed as a solution to bandwidth redundancy introduced by simulcast. This approach is based on information decomposition. That is, the video stream is encoded at a *base layer* and a number of *enhancement layers*, which can be combined to render the stream high quality. Layered adaptation is performed by adding or dropping enhancement layers depending on the prevailing network conditions.

We employ a quality adaptation mechanism in order to sustain smooth video delivery under

awkward conditions. We specifically adopt the layered approach, where the streaming server coarsely adjusts video quality without the need to implement transcoding. The efficiency of such mechanism can be affected by the frequency of layer changes. Since we do not know in advance how long we will be able to sustain a specific layer, minimal rate variations should not directly trigger video quality adjustments. Generally, switching layers unduly usually induces perceptible video quality variations with frustrating consequences to the end-user.

Along these lines, we concentrate on defining a priori whether a new layer should be added under the properties of AIMD congestion control. We rely on a layered scheme where each layer is multiplexed and supplies a corresponding buffer, as shown in Fig. 4. The server encodes raw video into n cumulative layers using a layered coder: layer 1 is the base layer and layer n is the least important enhancement layer. The layer rates are given by r_i, $i = 1,2,\ldots,n$. The receiver obtains a certain number of layers depending on bandwidth availability. All active layers k ($1 \leqslant k \leqslant n$) are typically multiplexed in a single AIMD flow with rate $R = \sum_{i=1}^{k} r_i = k\bar{r}$, where $\bar{r}$ denotes the average rate among the k layers. We assume that each buffer is drained with a constant rate c_i, $i = 1,2,\ldots,k$. Consequently, the total consumption rate at the receiver buffers is $C = \sum_{i=1}^{k} c_i = k\bar{c}$, where $\bar{c}$ denotes the average consumption rate of the buffers that correspond to the k active layers.

Following [15], certain conditions in the instantaneous transmission rate and the amount of buffering at the receiver can be applied in order to reduce the number of layer changes. The simplest way to perform layer adaptation is to send the maximum number of layers that can be accommodated in the available bandwidth. However, following this approach, the frequency of layer changes will be governed by the magnitude of AIMD oscillation with an adverse impact on user-perceived quality. Alternatively, layer switching can be reduced, if a new layer is added, as soon as the current transmission rate R exceeds the total consumption rate of all currently active layers plus the new one [15]

$$R > (k+1)\bar{c}. \tag{11}$$

However, relying on such rule does not eventually prevent layer changes; oscillations in the congestion control algorithm may still result in layer switching. Only sufficient buffering at the receiver can smooth out the variations in the available bandwidth and sustain a relatively constant number of active layers throughout the connection.

When transmission rate R exceeds the current total consumption rate C, the buffers are being filled with the spare data. In the event of a temporary drop in the sending rate, the adaptation scheme may be able to utilize the buffered data at the receiver in order to keep sending the same number of layers. Since the slope of linear increase for an AIMD mechanism can be easily estimated (based

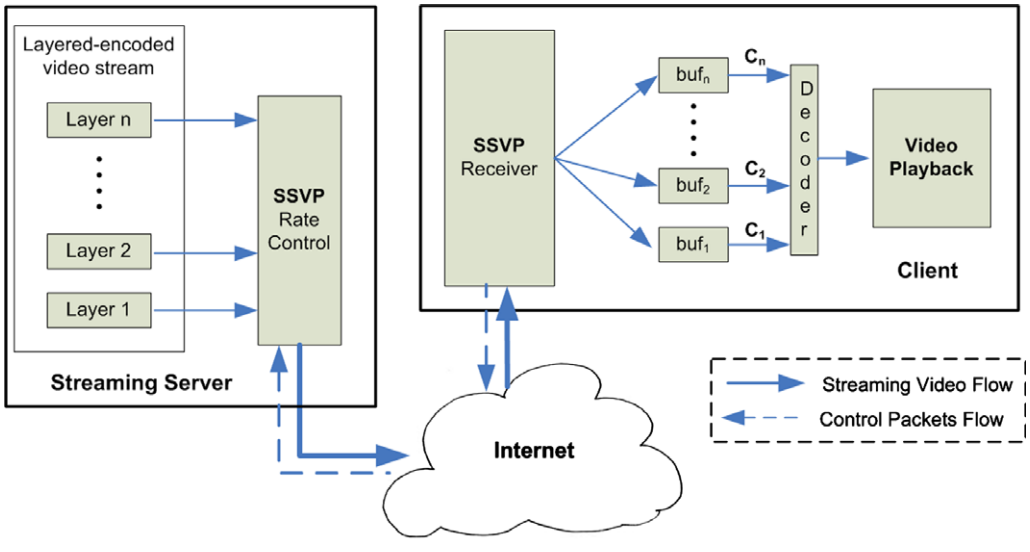

Fig. 4. Layered adaptation with receiver buffering.

on the increase rate α), we can derive a second rule expressing the amount of buffering required to prevent a layer drop, until the transmission rate has reached the total consumption rate again.

Fig. 5 depicts the behavior of an adaptive video flow under generalized AIMD (α, β) congestion control. If the AIMD source currently sends k layers, based on condition (11) a new layer can be added, as soon as R exceeds the total consumption rate corresponding to $k+1$ layers, i.e. $C = (k+1)\bar{c}$. At time t_1 we observe that $R > (k+1)\bar{c}$; an additional layer can be sent, provided that there is sufficient buffering at the receiver to survive an eventual packet loss. Therefore, the buffering requirements are satisfied when

$$\sum_{i=1}^{k} \mathrm{buf}_i \geqslant P, \tag{12}$$

where buf_i is the amount of video-data buffered for the ith layer and P is the shaded portion in Fig. 5. If condition (12) holds, the decoder draws an amount of the buffered data (or all buffered data if $\sum_{i=1}^{k} \mathrm{buf}_i = P$), allowing the continuous playback of $k+1$ layers, until R has reached $(k+1)\bar{c}$ at time t_2. Note that we have considered the most extreme scenario: a packet loss immediately after adding the new layer. If the loss occurs later, P (which represents the area of a triangle) will be smaller, allowing for more relaxed buffering requirements.

Based on Fig. 5, we derive P as follows:

$$P = \frac{1}{2}(t_2 - t_1)[(k+1)\overline{C} - (1-\beta)R]. \tag{13}$$

From the time period t_1 to t_2, the AIMD flow is in the congestion avoidance phase and the transmission rate R follows a line segment with slope λ.[2] The evolution of the transmission rate from t_1 to t_2 is expressed as

$$\begin{aligned} (k+1)\overline{C} &= (1-\beta)R + \lambda(t_2 - t_1), \\ t_2 - t_1 &= \frac{(k+1)\overline{C} - (1-\beta)R}{\lambda}. \end{aligned} \tag{14}$$

Combining Eqs. (13) and (14), P is given by

$$\begin{aligned} P &= \frac{(k+1)\overline{C} - (1-\beta)R}{2\lambda}[(k+1)\overline{C} - (1-\beta)R], \\ P &= \frac{[(k+1)\overline{C} - (1-\beta)R]^2}{2\lambda}. \end{aligned} \tag{15}$$

[2] λ represents the rate of linear increase in the transmission rate.

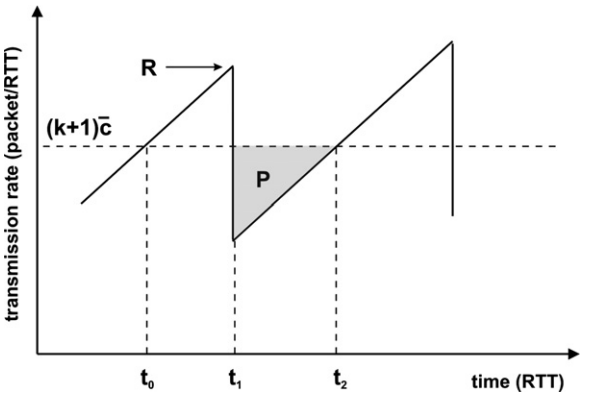

Fig. 5. Layered adaptation with AIMD rate control.

Finally, Eqs. (12) and (15) give the second condition, which defines the minimum amount of receiver buffering to send a new layer:

$$\sum_{i=1}^{k} \mathrm{buf}_i \geqslant \frac{[(k+1)\overline{C} - (1-\beta)R]^2}{2\lambda}. \tag{16}$$

Besides these criteria, the efficiency of the layered scheme is also subject to the transmission rate. Ideally, the cumulative consumption rate of all the buffers should equal the mean transmission rate. On the other hand, empty buffers may cause playback interruptions with frustrating consequences to the end-user. Along these lines, it is critical to combine the layered mechanism with a congestion control mechanism that maintains a relatively high and smooth sending rate. In Section 6.1 we show experimentally that SSVP satisfies these criteria. We rewrite Eq. (16) in the case of SSVP with $\lambda = 0.31$ and $\beta = 0.875$:

$$\sum_{i=1}^{k} \mathrm{buf}_i \geqslant \frac{1}{0.62}\left[(k+1)\overline{C} - \frac{R}{8}\right]^2. \tag{17}$$

Employing both rules for adding new layers enables the adaptation mechanism to smooth out the variations in the sending rate and eventually prevent rapid changes in video quality. Therefore, depending on the amount of buffered data, stable quality in video streaming applications can be attained, enhancing the user experience.

5. Experimental environment

5.1. Experimental settings

The evaluation plan was implemented on the *NS*-2 network simulator. In order to assess the efficiency

of our proposed solution, we implemented an experimental MPEG-4 video streaming server that supports layered adaptation. The traffic generated closely matches the statistical characteristics of an original MPEG-4 video trace. We developed three separate *Transform Expand Sample* (TES) models for I, P and B frames, respectively. The resulting video stream is generated by interleaving data obtained by the three models.

Simulations were initially conducted on a single-bottleneck *dumbbell* topology with a bottleneck capacity of 1 Mbps and a round-trip link delay of 44 ms (Fig. 6a). We also enabled simulations on a network topology (Fig. 6b) which addresses the heterogeneity of the Internet. The specific topology includes multiple bottlenecks, cross traffic, wireless links and diverse RTTs. The propagation delays of the access links from all the source nodes and the links to the FTP sink nodes range from 5 ms to 15 ms, while the corresponding bandwidth capacities range from 2 Mbps to 10 Mbps. Cross traffic includes diverse FTP flows over TCP Reno. The capacity of all access links to the MPEG sink nodes is set to 2 Mbps. By randomizing RTTs, we avoided synchronization effects.

In the *cross-traffic* topology NS-2 error models were inserted into the access links to the MPEG sink nodes with packet error rate adjusted at 1%. In both topologies we used drop-tail routers with buffer size adjusted in accordance with the *bandwidth-delay* product. Furthermore, we set the packet size to 1000 bytes for all system flows and the maximum congestion window to 64 KB for all TCP connections. The duration of each experiment is 60 s. The results are collected after 2 s in order to avoid the skew introduced by the startup effect.

5.2. Measuring performance

We hereby refer to the performance metrics supported by our simulation model. Since both topologies include MPEG flows competing with corporate FTP flows, the performance metrics are applied separately to the MPEG and FTP traffic. *Goodput* was used to measure the overall system efficiency in bandwidth utilization. Goodput is defined as

$$\text{Goodput} = \frac{\text{Original Data}}{\text{Connection Time}},$$

where *Original Data* is the number of bytes delivered to the high-level protocol at the receiver (i.e. excluding retransmitted packets and overhead) and *Connection Time* is the amount of time required for data delivery. Inter-protocol fairness measure-

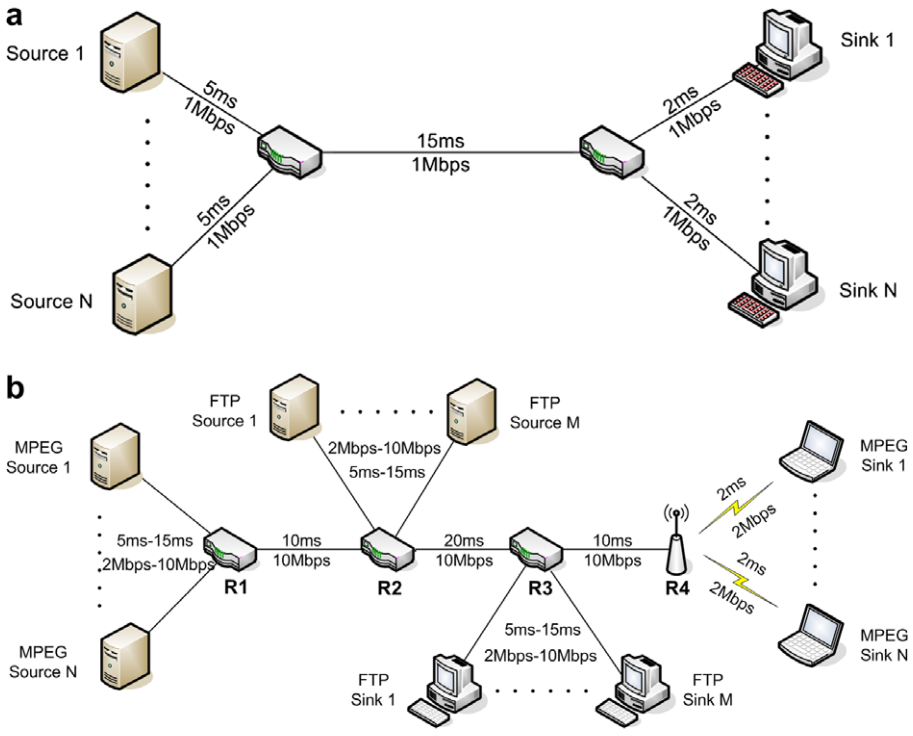

Fig. 6. Simulation topologies: (a) dumbbell topology and (b) cross-traffic topology.

ments were conducted based on *Normalized Throughput*, which is the ratio of the average throughput received by each flow over the bandwidth fair share on each case.

The task of specifying the effects of network QoS parameters on video quality is challenging. Transmission rate fluctuations, increased delays, jitter and packet loss commonly deteriorate the perceptual quality or fidelity of the received video content. However, these parameters do not affect quality in an independent manner; they rather act in combination or cumulatively, and ultimately, only this joint effect is detected by the end-user.

Jitter composes a critical factor in the performance of video delivery. Packet jitter is the delay variation experienced by packets in a single session. Let $D(i,j)$ denote the value of packet spacing at the receiver compared with packet spacing at the sender for a pair of packets i and j. $D(i,j)$ is represented as

$$\begin{aligned} D(i,j) &= (R_j - R_i) - (S_j - S_i) \\ &= (R_j - S_j) - (R_i - S_i), \end{aligned} \tag{18}$$

where S_i, S_j, R_i and R_j denote the sending and receiving times for packets i and j, respectively. In the absence of jitter, the spacings will be the same and $D(i,j)$ will be zero. Packet jitter is calculated continuously as a weighted average of the observed values of $D(i,j)$

$$J(i,j) = \frac{15}{16}J(i,j) + \frac{1}{16}|D(i,j)|. \tag{19}$$

We further exploit two metrics for the evaluation of perceptual video quality: *Peak Signal-to-Noise Ratio* (PSNR) and *Video Delivery Index* (based on [14]). PSNR compares the maximum possible signal energy to the noise energy between a source and destination image, I_S and I_D respectively. PSNR is defined as

$$\mathrm{PSNR}(I_D, I_S) = 20 \log 10 \frac{V_{\mathrm{peak}}}{\mathrm{MSE}(I_S, I_D)} \ [\mathrm{dB}], \tag{20}$$

where $\mathrm{MSE}(I_S, I_D)$ is the mean square error of the two images and $V_{\mathrm{peak}} = 2^h - 1$, with h the bit color depth. We note that representing PSNR frame by frame is more tractable than calculating the average of PSNR values of all frames, since an average PSNR may not map well to the overall subjective impression during video playback. Therefore, we rely on the frame-wise PSNR to assess video quality only for a single MPEG transfer. [11] provides a heuristic mapping of PSNR to *Mean Opinion Score* (MOS), as shown in Table 1. MOS is a numerical measure of perceptual quality at the receiving end. The metric virtually indicates the video quality perceived by the end-user on a scale from 1 (worst) to 5 (best).

Table 1
Possible PSNR to MOS conversion

PSNR	MOS
>37	5 (Excellent)
31–37	4 (Good)
25–31	3 (Fair)
20–25	2 (Poor)
<20	1 (Bad)

For experiments with multiple flows, we use *Video Delivery Index*, which captures the joint effect of jitter and packet loss on perceived quality. The metric monitors packet inter-arrival times and distinguishes the packets that can be effectively used by the client application (i.e. without causing interruptions) from delayed packets according to a configurable packet inter-arrival threshold. The proportion of the number of delayed packets is denoted as *Delayed Packets Rate*. Video Delivery Index is defined as the ratio of the number of *jitter-free* packets over the total number of packets sent by the application

$$\text{Video Delivery Index} = \frac{\#\text{jitter-free packets}}{\#\text{sent packets}} \leqslant 1.$$

In accordance with video streaming delay guidelines, we adjusted the packet inter-arrival threshold at 75 ms. For a system with multiple flows, we present the average of the Video Performance Index of each MPEG flow.

6. Performance evaluation

In this section, we demonstrate extensive performance studies based on simulations. More precisely, we assess SSVP's performance from the perspective of bandwidth utilization, video delivery, and inter-protocol fairness. Further, we quantify the efficiency of the receiver-buffered scheme, focusing on the interactions between layered adaptation and SSVP congestion control.

6.1. Single SSVP flow

Initially, we evaluate the efficiency of SSVP from the perspective of video delivery. We conducted

simulations on the dumbbell topology, where an SSVP flow competes with two FTP flows over TCP Reno. Fig. 7 illustrates an excerpt from an MPEG transfer with SSVP. The protocol is able to sustain a regular transmission rate with oscillations of small magnitude. More precisely, the integrated AIMD $(0.31, 0.875)$ congestion control results in gentle rate reductions in response to packet drops. SSVP may also achieve a certain level of responsiveness, although it does not exhibit TCP's prompt responses to sudden bandwidth availability, due to the small increase rate.

The performance of video delivery is additionally shown in Fig. 8. An SSVP flow shares the bottleneck link with two FTP connections, which are expected to cause noticeable disturbances on perceived video quality. Nevertheless, jitter never exceeds the frustrating limit of 75 ms, since SSVP effectively smoothes transmission gaps validating our choice to apply a multiplicative decrease factor of 0.875. Such a performance does not necessitate the use of deep playback buffers in order to ameliorate the negative effect of jitter. Only in conditions of scarce bandwidth and increased contention, a playback buffer may notably improve the perceptual quality and reduce potential interruptions on video playback.

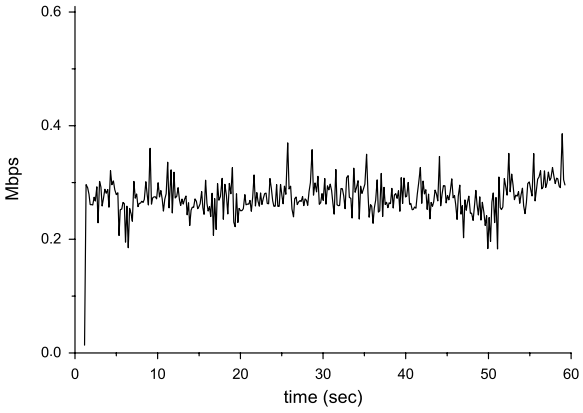

Fig. 7. SSVP sending rate.

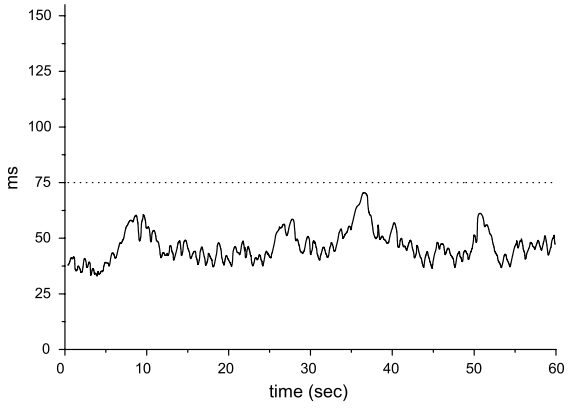

Fig. 8. SSVP packet jitter.

Fig. 9 illustrates frame-by-frame PSNR measurements for an MPEG video transfer under the same network conditions. PSNR values are acceptable throughout the video sequence, with the absence of considerable distortions in the video stream. Note that a certain amount of distortion is unavoidable, as the effect of lossy coding algorithms. Mapping the PNSR values of Fig. 9 to MOS grades, based on Table 1, provides a simplified and widely used numerical description of visual quality perception. All received video frames are characterised by either *good* or *excellent* MOS grade, achieving satisfactory performance on video delivery throughout the entire connection.

Based on the same network topology (i.e. dumbbell), we investigate the impact of SSVP on corporate traffic. We simulated a single SSVP flow competing with a diverse number of FTP flows (1–30) successively. Fig. 10 illustrates the corresponding *Normalized Throughput* measurements. The target sending rate for SSVP is adjusted at 380 Kbps in order to enforce strong contention with interfering TCP flows. Despite the limited bandwidth resources (1 Mbps), TCP flows are allowed to obtain a fair share of the link (in each case they score *Normalized Throughput* of nearly 1). On the other hand, SSVP manages to allocate the remaining resources, since bottleneck link utilization is always more than 80%.

Fig. 11 illustrates the sending rates of an SSVP and an FTP flow over TCP Reno, while sharing a network channel (dumbbell topology). In this case,

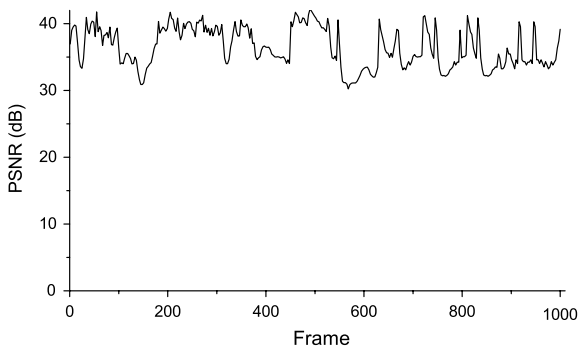

Fig. 9. PSNR.

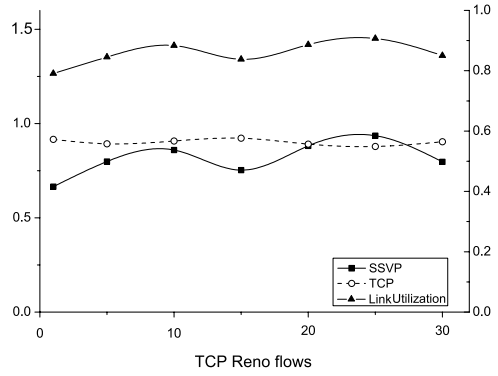

Fig. 10. Normalized throughput.

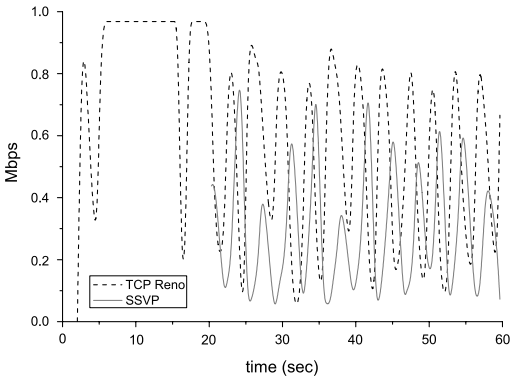

Fig. 11. SSVP coexisting with TCP Reno.

the target sending rate for SSVP is set to 1 Mbps. The video flow joins the network at 20 s and competes with FTP traffic for the remaining 40 s. SSVP allows the evolution of the TCP's sending rate, which periodically approximates the bottleneck capacity. We also observe that TCP achieves a rapid recovery from multiple losses; an indication that SSVP has a friendly behavior to corporate traffic. Furthermore, Fig. 11 depicts that both SSVP and TCP flows eventually reach a steady state, i.e. each flow additively increases and multiplicatively decreases its transmission rate periodically. However, we note that TCP converges faster to the steady state than SSVP, since TCP employs a higher increase rate. A conclusive overview of Figs. 10 and 11 indicates that SSVP coexists fairly with TCP.

6.2. Performance with heterogeneous networks and high link-multiplexing

We additionally carried out a series of simulations on the cross-traffic topology in order to assess the performance of SSVP versus TCP-friendly and UDP traffic. We simulated a wide range of MPEG flows (1–50) with (i) SSVP, (ii) TFRC, (iii) TCP Westwood+ (TCPW+), and (iv) UDP, competing with 10 FTP connections of TCP Reno, successively. The corresponding results appear in Fig. 12. In addition, we present selected traces of the queue length of the router R2 (Fig. 13) in the presence of 40 MPEG flows (plus 10 FTP connections).

Since UDP does not incorporate any close-loop control, the sender keeps transmitting at application rate. As a result, at high contention (30–50 flows) the protocol achieves the highest goodput rates, outperforming the rest of the protocols. SSVP also exhibits high bandwidth utilization, regardless of link multiplexing (Fig. 12a). Inline with the single-flow results, SSVP adapts to the vagaries of the network and is also less susceptible to random packet loss, such as the sporadic wireless errors. Both TFRC and TCPW+ are able to change the sending rate adaptively, although in a different fashion. In the situation of low contention (1–20 flows) where packet loss rate is insignificant, TFRC manages to effectively utilize the available bandwidth. However, during high link multiplexing and compared to SSVP, TFRC exhibits a slight deficiency, as shown in Fig. 12a. TFRC is designed to respond to a loss event (which may include several packet drops) instead of a packet loss. However, in environments with transient errors, TFRC occasionally fails to obtain accurate estimates of the loss event rate, invoking an inappropriate equation-based recovery that impacts its performance. Furthermore, TFRC's gentle backward adjustments in response to congestion may favor smooth video delivery in normal conditions, but occasionally fail to relinquish a considerable amount of the allocated network resources during congestion periods. As a result, queues in the bottleneck buffers are rapidly built up. The average queue-length measurements of router R2, as shown in Table 2, reveal a tendency of TFRC to buffer overflows. TCPW+ yields a notable inefficiency in terms of goodput performance (Fig. 12a). Despite the improvements over the initial version of Westwood, TCPW+'s algorithm still does not obtain accurate estimates in heterogeneous environments, since the estimation filter is slow, needing time to converge to the available bandwidth. This observation is profound in the case of limited bandwidth (high contention), where the goodput rate is diminished.

Since goodput gains do not necessitate improved performance on video delivery, we demonstrate the

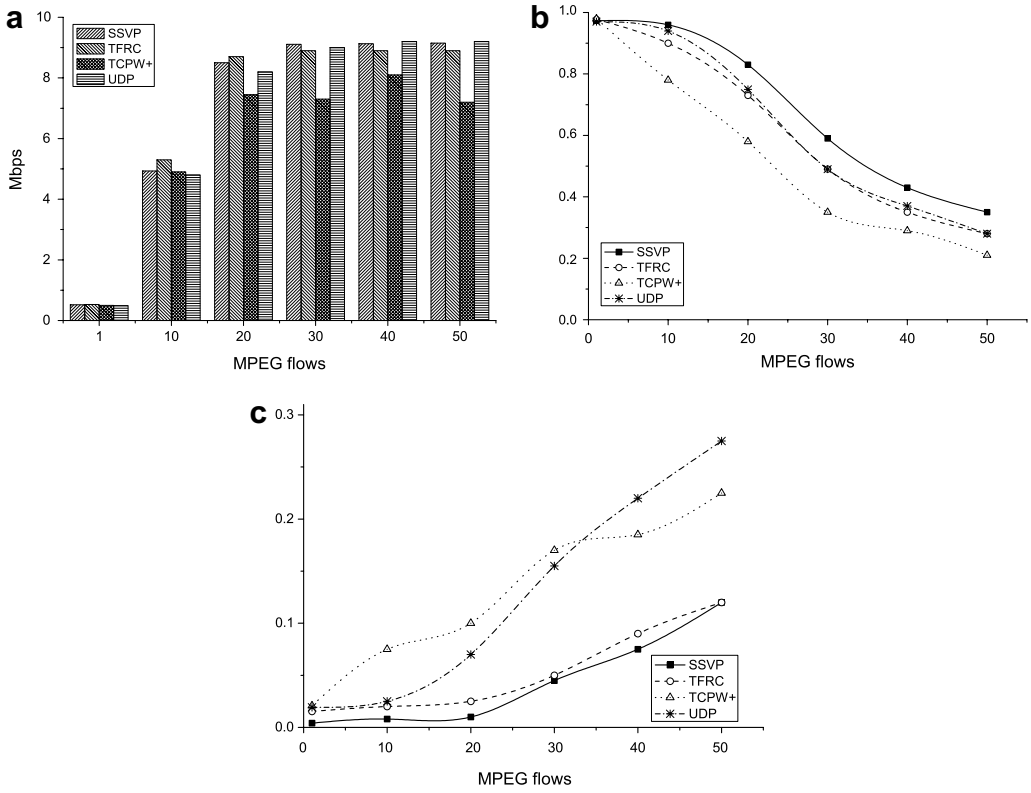

Fig. 12. Performance with heterogeneous networks: (a) goodput of MPEG flows, (b) average Video Delivery Index and (c) Delayed Packets Rate.

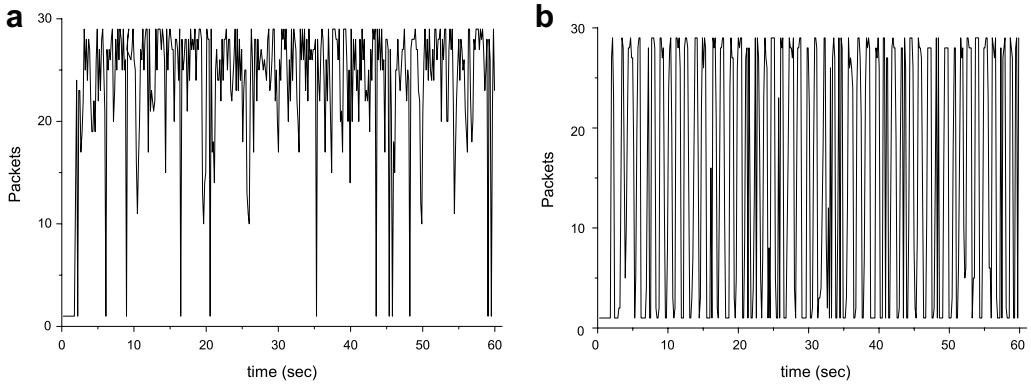

Fig. 13. Queue length of router R2 (40 MPEG flows): (a) UDP and (b) SSVP.

Table 2
Average queue length of router R2 (40 MPEG flows)

Protocol	Average queue length
TFRC	27.8
TCP Westwood+	12.2
UDP	23.6
SSVP	15.3

average *Video Delivery Index* of the MPEG flows, as well as statistics from delayed packets in order to quantify protocol efficiency. According to Fig. 12b, SSVP yields the most prominent performance on video delivery. The smoothness-oriented modulation of the AIMD parameters and the low frequency of rate adaptation enhance application

performance and maximize the user experience. Furthermore, Fig. 12c illustrates that both SSVP and TFRC achieve the timely delivery of most packets, inducing minimal impairments on video quality. Unlike the unresponsive UDP that results in rapidly growing queues and buffer overflows (Fig. 13a), SSVP enforces a more persistent buffer draining phase by promptly responding to congestion (Fig. 13b). Despite the choice of a large β, SSVP performs sharper backward adjustments than TFRC, according to the average queue-length measurements of Table 2. Since SSVP does not overload network buffers, it maintains network stability and is cooperative with interfering flows.

Despite TFRC's respectable performance in terms of bandwidth utilization, the increased packet drops degrade its efficiency, as shown by *Video Delivery Index* in Fig. 12b, which reflects the joint effect of jitter and packet loss. In addition, when competing long-lived TCP and TFRC flows on the same bottleneck have different sending rates, their observed loss event rates can be significantly different [17]. Since TFRC's throughput model, as expressed in Eq. (1), is sensitive to the packet loss rate, the transmission rate can be adjusted inaccurately with direct implications on real-time delivery. On the other hand, TCPW+'s low goodput rates inevitably impact the performance on video delivery (Fig. 12b). Besides the tendency of TCPW+ to underestimate the available bandwidth, the protocol slows down the transmission in response to the transient errors. Consequently, the resulting transmission gaps cause interruptions in the receiving rate and playback of the video stream (Fig. 12c).

6.3. Performance with receiver-buffered layered adaptation

Departing from a comparative overview of protocol performance in terms of video delivery, we quantify the performance of the receiver-buffered adaptation scheme. In this context, we enabled additional simulations of diverse SSVP connections augmented by the layered adaptation mechanism. In contrast to immediate adaptation (i.e. based on instantaneous available bandwidth), our implementation sends an additional layer only when both conditions (11) and (17) hold.

Fig. 14 illustrates an excerpt from an MPEG transfer over SSVP with receiver-buffered layered adaptation. The simulation was conducted on the dumbbell topology, where a single SSVP flow competes with two FTP flows (over TCP Reno). The receiver-buffered scheme results in a minimal number of layer changes, adapting video quality along with long-term variations in the available bandwidth. We specifically observe several occasions during the MPEG transfer, where increasing rates are not immediately followed by the allocation of a new layer. Furthermore, the sender manages to deliver higher quality video, since it is periodically able to transmit at the highest layer (i.e. layer 4).

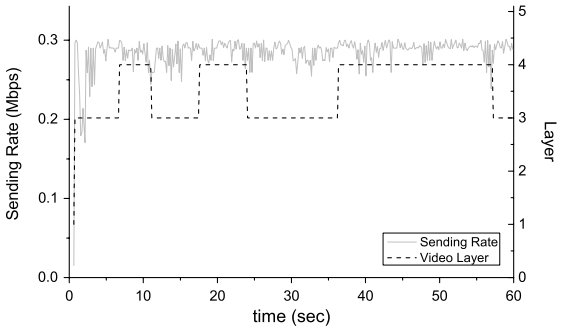

Fig. 14. SSVP sending rate and layer variation.

In the sequel, we evaluate the efficiency of the quality adaptation scheme in highly multiplexed dynamic networks. Hence, we conducted additional experiments on the cross-traffic topology simulating SSVP flows and interfering TCP traffic (10 FTP connections). We compare our findings with the SSVP measurements from the previous scenario, where we did not exploit the layered adaptation scheme. The corresponding results are shown in Fig. 15.

The SSVP flows with the layered adaptation (i.e. SSVP-LA) deliver video of lower bitrate (the difference in rate/quality is subject to the prevailing network conditions), and consequently achieve lower goodput rates (Fig. 15a). On the other hand, in the absence of a scalable coder, streaming video is transmitted at optimal quality, achieving higher link utilization. Fig. 15b illustrates the perceptual QoS assessment of video quality, which is sensitive to impairments, such as transmission rate fluctuations, jitter and packet loss. The beneficial role of the adaptation scheme is more evident, as the level of link multiplexing increases. SSVP flows exhibit a perceptible sensitivity to limited bandwidth due to the increased contention, and eventually deliver sub-optimal performance on video delivery (Fig. 15b). On the contrary, the combined approach of SSVP-LA is notably more efficient, since it alleviates most of the impairments induced by the

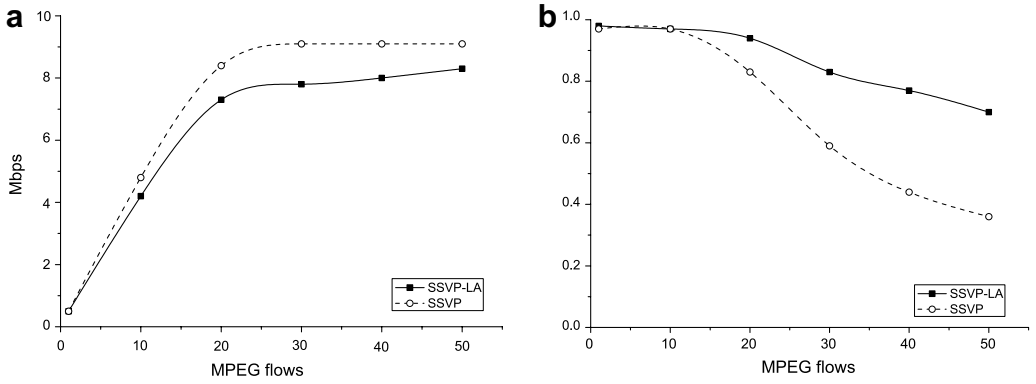

Fig. 15. The effect of receiver-buffered layered adaptation: (a) goodput of MPEG flows and (b) average Video Delivery Index.

disturbances of corporate flows. Besides smooth video delivery, SSVP-LA effectively limits the quality degradation induced by dropped frames, since the adaptation mechanism renders the underlying protocol (i.e. SSVP) more robust to congestion, as well as to transient loss. Therefore, the flexibility of the receiver-buffered layered approach allows for the delivery of smooth video in a wide range of network and session dynamics.

7. Conclusions and future work

We have presented an alternative solution for the delivery of congestion-controlled video over the Internet. SSVP enables AIMD-oriented congestion control on top of the light-weight UDP and incorporates attractive properties that enhance real-time delivery. The selection of protocol parameters and the IPG adjustments reduce the magnitude of AIMD oscillation, while the established goal of TCP-friendliness is not compromised. Based on simulations, we validated the efficiency of SSVP and we also demonstrated its feasibility in terms of wide range deployment. We also showed that SSVP compares very favorably with congestion control mechanisms that explicitly address time-sensitive traffic, such as TFRC. Beyond the protocol's behavior, we quantified the interactions of SSVP with a receiver-buffered quality adaptation mechanism. In this situation, we identified further gains on the performance on video delivery that can be achieved on a long-term timescale, especially in the presence of scarce bandwidth. Essentially, SSVP can effectively interact with scalable video coders, achieving improved performance in highly multiplexed dynamic networks.

Although SSVP is primarily designed for video streaming applications, it can also provide efficient transport services for delay-sensitive applications with relaxed packet loss requirements. Retransmissions can be easily layered above SSVP, enabling the protocol to support a broader application domain (e.g. semi-reliable applications). We plan to integrate a mechanism to adjust reliability according to the prevailing network conditions, triggering retransmissions for the most important data. We also investigate potential gains from the adaptive modulation of AIMD parameters, based on current network dynamics.

References

[1] D. Bansal, H. Balakrishnan, Binomial congestion control algorithms, in: Proceedings of the IEEE INFOCOM 2001, Anchorage, Alaska, USA, April 2001.
[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, An Architecture for Differentiated Services, RFC 2474, 1998.
[3] L. Brakmo, L. Peterson, TCP Vegas: end to end congestion avoidance on a global internet, IEEE Journal on Selected Areas of Communications 13 (8) (1995) 1465–1480.
[4] M. Chen, A. Zakhor, Rate control for streaming over wireless, in: Proceedings of the IEEE INFOCOM 2004, Hong Kong, China, March 2004.
[5] D. Chiu, R. Jain, Analysis of the increase/decrease algorithms for congestion avoidance in computer networks, Journal of Computer Networks and ISDN 17 (1) (1989) 1–14.
[6] N. Feamster, H. Balakrishnan, Packet Loss Recovery for Streaming Video, in: Proceedings of the 12th IEEE Int/nal Packet Video Workshop, Pittsburgh, USA, April 2002.
[7] N. Feamster, D. Bansal, H. Balakrishnan, On the interactions between layered quality adaptation and congestion control for streaming video, in: Proceedings of the 11th IEEE Int/nal Packet Video Workshop, Korea, 2001.
[8] S. Floyd, K. Fall, Promoting the use of end-to-end congestion control in the Internet, IEEE/ACM Transactions on Networking 7 (4) (1999) 458–472.
[9] S. Floyd, M. Handley, J. Padhye, J. Widmer, Equation-Based Congestion Control for Unicast Applications, in:

Proceedings of the ACM SIGCOMM 2000, Stockholm, Sweden, August 2000.
[10] L. Grieco, S. Mascolo, Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control, ACM Computer Communication Review 34 (2) (2004) 25–38.
[11] J. Gross, J. Klaue, H. Karl, A. Wolisz, Cross-layer optimization of OFDM transmission systems for MPEG-4 video streaming, Computer Communications 27 (11) (2004) 1044–1055.
[12] E. Kohler, M. Handley, S. Floyd, Designing DCCP: congestion control without reliability, in: Proceedings of the ACM SIGCOMM 2006, Piza, Italy, September 2006.
[13] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, R. Wang, TCP Westwood: bandwidth estimation for enhanced transport over wireless links, in: Proceedings of the ACM MOBICOM 2001, Rome, Italy, July 2001.
[14] P. Papadimitriou, V. Tsaoussidis, S. Tsekeridou, The impact of network and protocol heterogeneity on application QoS, in: Proceedings of the 10th IEEE Int/nal Symposium on Computers and Communications (ISCC), Cartagena, Spain, June 2005.
[15] R. Rejaie, M. Handley, D. Estrin, Layered quality adaptation for internet video streaming, IEEE Journal on Selected Areas in Communications (JSAC) 18 (12) (2000) 2530–2544.
[16] R. Rejaie, M. Handley, D. Estrin, RAP: an end-to-end rate-based congestion control mechanism for realtime streams in the Internet, in: Proceedings of the IEEE INFOCOM, New York, USA, March 1999.
[17] I. Rhee, L. Xu, Limitations of equation-based congestion control, in: Proceedings of the ACM SIGCOMM 2005, Philadelphia, USA, August 2005.
[18] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: a transport protocol for real-time applications, RFC 3550, IETF 2003.
[19] S. Shenker, C. Partridge, R. Guerin, Specification of guaranteed Quality of Service, RFC 2212, 1997.
[20] V. Tsaoussidis, C. Zhang, TCP Real: Receiver-oriented congestion control, Computer Networks 40 (4) (2002) 477–497.
[21] V. Tsaoussidis, C. Zhang, The dynamics of responsiveness and smoothness in heterogeneous networks, IEEE Journal on Selected Areas in Communications (JSAC) 23 (6) (2005) 1178–1189.
[22] D.X. Wei, C. Jin, S.H. Low, S. Hegde, TCP FAST: motivation, architecture, algorithms, performance, IEEE/ACM Transactions on Networking 14 (6) (2006) 1246–1259.
[23] Y.R. Yang, M.S. Kim, S.S. Lam, Transient behaviors of TCP-friendly congestion control protocols, in: Proceedings of the IEEE INFOCOM 2001, Anchorage, Alaska, USA, April 2001.
[24] Y.R. Yang, S.S. Lam, General AIMD congestion control, in: Proceedings of the 8th IEEE ICNP, Osaka, Japan, November 2000.
[25] C. Zhang, V. Tsaoussidis, TCP real: improving real-time capabilities of TCP over heterogeneous networks, in: Proceedings of the 11th IEEE/ACM NOSSDAV, Port Jefferson, New York, USA, June 2001.
[26] C. Zhang, V. Tsaoussidis, TCP Smoothness and window adjustment strategy, IEEE Transactions in Multimedia 8 (3) (2006) 600–609.

Panagiotis Papadimitriou obtained a B.Sc. in Computer Science from University of Crete, Greece; and an M.Sc. in Information Technology from University of Nottingham. He is currently a Ph.D. Candidate in Electrical and Computer Engineering in Democritus University of Thrace, Greece. He is a Visiting Lecturer in the Information Management Department of Technological Educational Institute (TEI) of Kavala, Greece. His research interests include transport protocols for multimedia applications, video and voice over IP, congestion control, and space communications.

Vassilis Tsaoussidis received a B.Sc. in Applied Mathematics from Aristotle University, Greece; a Diploma in Statistics – and Computer Science from the Hellenic Institute of Statistics; and a Ph.D. in Computer Networks from Humboldt University, Berlin, Germany (1995). He held faculty positions in Rutgers University, New Brunswick, SUNY Stony Brook and Northeastern University, Boston. In May 2003, he joined the Department of Electrical and Computer Engineering of Democritus University of Thrace, Greece. His research interests lie in the area of transport/network protocols, i.e. their design aspects and performance evaluation. He is editor-in-chief in the Journal of Internet Engineering and editor for IEEE Transactions in Mobile Computing, the Journal of Computer Networks (Elsevier), the Journal of Wireless Communications and Mobile Computing and the Journal of Mobile Multimedia. He participates in several Technical Program Committees in his area of expertise, such as INFOCOM, GLOBECOM, ICCCN, ISCC, EWCN, WLN, and several others.