Optimization of AIMD Congestion Control for Media-Streaming Applications

Panagiotis Papadimitriou and Chi Zhang

***Abstract*—Media-streaming applications require smooth patterns of data transmission and also benefit from efficient resource utilization. System-wise, the underlying congestion control mechanism should achieve fairness and maintain TCP-friendliness. In this context, we optimize *Additive Increase Multiplicative Decrease* (AIMD) congestion control for multimedia applications, within the framework of bandwidth efficiency, smoothness, and inter-protocol fairness. We assume *Scalable Streaming Video Protocol* (SSVP) as the underlying congestion control mechanism. Departing from SSVP's throughput model and based on the concepts of the *knee* and the *cliff* as defined in [4], we provide an analysis of AIMD congestion control, taking into account the role of the bottleneck queue. We observe that although multiplicative decrease is necessary to achieve fairness, it does not necessarily sacrifice the system throughput, as long as the system operates between the *knee* and the *cliff*. The proposed AIMD mechanism introduces congestion control parameters adaptable to current network conditions, preventing the system from operating below the *knee*, where a fraction of the available bandwidth is not utilized and smoothness is compromised as throughput fluctuates.**

***Index Terms*—*Congestion Control*, *Quality of Service*, *Multimedia Streaming*.**

I. Introduction

Recently Internet has been experiencing an increasing demand for multimedia services, such as audio and video delivery. Media-streaming applications yield satisfactory performance only under certain *Quality of Service* (QoS) provisions, which may vary depending on the application task and the type of media involved. Unlike bulk-data transfers, multimedia flows require a minimum bandwidth guarantee, and they are also sensitive to variations of throughput and delay. Generally, streaming applications seek to achieve smooth playback quality rather than simply transmit at the highest attainable bandwidth.

Additive Increase Multiplicative Decrease (AIMD) [4] flows compose the Internet's main bandwidth consumer. The goal of AIMD algorithms is to prevent applications from either overloading or under-utilizing the available network resources. Although AIMD-based *Transmission Control Protocol* (TCP) provides reliable and efficient services for bulk-data transfers, several design issues render the protocol a less attractive solution for multimedia applications. More precisely, the process of probing for bandwidth and reacting to observed congestion causes oscillations to the achievable transmission rate. Furthermore, TCP occasionally introduces arbitrary delays, since it enforces reliability and in-order delivery. In response to standard TCP's limitations, several *TCP-friendly* protocols (e.g. [6, 18, 19]) achieve smoother sending rate adjustments, while they manage to compete fairly with TCP flows. In order to achieve smoothness, they use gentle backward adjustments upon congestion. However, they compromise responsiveness through moderated upward adjustments [17].

Essentially, the choice of additive increase rate α and multiplicative decrease rate 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 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 data delivery, with an adverse effect on the playback quality of media-streaming applications. The variations in sending rate can be theoretically smoothed out with application-level buffering, but this could result in huge client buffers and unacceptable end-to-end delays depending on the extent of fluctuations.

Most AIMD protocols tune these parameters to favor either smoothness or responsiveness, depending on the underlying network and the supporting application requirements. From the perspective of delay-sensitive traffic, a static and smoothness-oriented modulation of AIMD parameters may impose considerable limitations in terms of resource utilization, especially in dynamically changing environments, such as the Internet which operates in the transient than in the stationary regime. Therefore, the challenge does not lie in simply reducing AIMD oscillation, but rather providing smoothness along with bandwidth efficiency and fairness.

Following these observations, we analyze and further extend the dynamics of AIMD congestion control to optimize bandwidth utilization and concurrently achieve high levels of smoothness, inline with the requirements of streaming applications. We assume *Scalable Streaming Video Protocol* (SSVP) [11] as the underlying congestion control mechanism; however, the proposed model can be easily incorporated into any congestion control scheme that is based on an AIMD algorithm. SSVP is an AIMD-oriented rate-based protocol that operates on top of UDP. Generally, rate-based congestion control [6, 11, 13] composes a plausible candidate for media-streaming applications, considering TCP's limitations and the impending threat of unresponsive *User Datagram Protocol* (UDP). SSVP generates a smoothed data flow by spreading the data transmission across a time interval, avoiding the burstiness occasionally induced by the window-based mechanisms.

Manuscript received May 12, 2007; revised August 2, 2007.

Panagiotis Papadimitriou is with Democritus University of Thrace, Xanthi, Greece (e-mail: ppapadim@ee.duth.gr). Chi Zhang is with Juniper Networks, Sunnyvale, CA 94086, USA (e-mail: chizhang@juniper.net).

However, the protocol employs static AIMD parameters (i.e. $\alpha = 0.31$, $\beta = 0.875$) and may result in inferior bandwidth utilization.

In this context, we optimize SSVP congestion control with the adaptive modulation of additive increase and multiplicative decrease factors in order to provide the underlying AIMD algorithm with a sufficient operating scope, where the bottleneck bandwidth is fully exploited and system throughput is stable. Departing from SSVP's throughput model, we derive an analytical expression for the points *knee* and *cliff*, as defined in [4]. *Knee* is considered as the point where the whole bottleneck capacity has been utilized and the queue is empty. Likewise, the point where the queue starts overflowing is called *cliff*. We observe that although multiplicative decrease is necessary to achieve fairness, it does not necessarily sacrifice the system throughput, as long as the system operates between the *knee* and the *cliff*. Consequently, we derive an equation for the adaptive adjustment of an effective decrease ratio that (i) prevents the system from operating below the *knee*, where the available bandwidth remains underutilized and (ii) maintains adequate AIMD oscillation which is required by the system in order to converge to fairness. The proposed mechanism introduces new congestion control parameters adaptable to current network conditions to avoid the damage of static multiplicative decrease on throughput performance and smoothness. We note that the specific adjustment strategy relies only on *Round Trip Time* (RTT) estimation and does not require the collaboration of *Active Queue Management* mechanisms, such as *Random Early Detection* (RED) [7] or *Explicit Congestion Notification* (ECN) [12].

The remainder of the paper is organized as follows. Section II reviews related work. In Section III we provide an overview of SSVP and we derive a throughput model for the SSVP flows. In Section IV we analyze the dynamics of AIMD congestion control and we elaborate on the proposed AIMD mechanism, which is further incorporated into SSVP. Section V includes our evaluation methodology followed by Section VI, where we provide extensive performance studies based on simulations. Finally, in Section VII we highlight our conclusions and refer to future work.

II. Related Work

The literature includes numerous studies and proposals towards efficient congestion control for media-streaming applications in the Internet. Authors in [10] provide a comparative overview of congestion control schemes for the Internet. Furthermore, [8] studies analytically smoothness, responsiveness, and the fairing speed of linear congestion control algorithms, including AIMD.

Rate Adaptation Protocol (RAP) [13] 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. RAP attempts to resemble TCP's functionality, leaving out only the undesired reliability. The RAP source receives acknowledgments (ACK) infrequently and exploits the redundant information on a single incoming ACK to detect packet loss, inline with TCP's *Fast Recovery* algorithm [15]. However, some aspects of TCP design that do not favor smooth delivery are incorporated into RAP. For example, the multiplicative decrease by a factor of ½ invokes abrupt rate reductions upon congestion, compromising smoothness.

Since TCP is rarely chosen to transport delay-sensitive 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 [5]. TCP-friendly congestion control maintains network stability by promptly responding to congestion and is also cooperative with other flows, while it commonly provides more efficient QoS, i.e. smoothed sending rate and reduced delays for media-streaming applications. *TCP-friendly Rate Control* (TFRC) [6] 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 by TFRC and hence, the protocol follows a more gentle congestion control strategy. More precisely, the TFRC sender uses the following TCP response function:

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

where p is the steady-state loss event rate and *RTO* is the retransmission timeout value. Equation (1) enforces an upper bound on the sending rate T. However, the throughput model is quite sensitive to parameters (i.e. p, RTT), which are often difficult to measure efficiently and predict accurately. Also, the long-term TCP throughput equation does not capture the transit and short-lived TCP behaviors, and it is less responsive to short-term network and session dynamics [17]. Authors in [14] uncover a long-term throughput imbalance between competing TFRC and TCP connections, reporting that the throughput difference can be further amplified, as long as the TCP and TFRC flows experience different packet loss rates.

GAIMD is a TCP-friendly protocol that generalizes AIMD congestion control by parameterizing the additive increase rate α and multiplicative decrease ratio β. For the family of AIMD protocols, authors in [19] derive a simple relationship between α and β in order to be friendly to standard TCP:

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

Based on experiments, authors in [19] propose an adjustment of $\beta = 0.875$ as an appropriate smooth decrease ratio, and a moderated increase value $\alpha = 0.31$ to achieve TCP friendliness.

TCP-Real [16] is a high-throughput transport protocol that incorporates a congestion avoidance mechanism in order to minimize transmission-rate gaps. The protocol employs a receiver-oriented and measurement based congestion control mechanism that significantly improves real-time performance over heterogeneous networks and asymmetric paths.

Datagram Congestion Control Protocol (DCCP) [9] is a

new transport protocol that provides a congestion-controlled flow of unreliable datagrams. DCCP is intended for delay-sensitive applications with 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.

Binomial congestion control [1], such as IIAD or SQRT, is also attractive to multimedia applications for its smooth rate variations. For example, SQRT responds to packet drops by reducing the congestion window size proportional to the square root of its value instead of halving it. However, binomial schemes are not able to achieve TCP-friendliness independent of link capacity [3]. Apart from link capacity, the selection of increase rate and decrease ratio composes another influencing parameter.

III. SSVP Overview and Throughput Model

Scalable Streaming Video Protocol (SSVP) [11] is an end-to-end TCP-friendly protocol optimized for unicast video streaming applications. SSVP operates on top of the light-weight UDP which is already preferred by the majority of streaming applications and Internet telephony. The protocol employs AIMD-oriented congestion control and adapts the sending rate by adjusting the inter-packet gap (IPG). SSVP applies modifications only in the sending and receiving hosts. The recipient uses control packets in order to send feedback of reception statistics to the sender. 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 to determine bandwidth and RTT estimates, and properly adjust the rate of the outgoing video streams.

SSVP enables a smoothness-oriented modulation of AIMD parameters in order to reduce the magnitude of AIMD oscillation and allow for smooth transmission patterns, without compromising TCP-friendliness. More precisely, SSVP's congestion control employs an additive increase rate $\alpha = 0.31$ and a multiplicative decrease ratio $\beta = 0.875$. The sender adjusts the transmission rate once per RTT in order to maintain a smoothed flow. Let S denote the packet length, the instantaneous transmission rate R_i for an SSVP flow is given by:

$$R_i = \frac{S}{t_i + IPG_i} \approx \frac{S}{IPG_i} \quad (3)$$

if we consider the transmission time t_i of the i^{th} packet negligible (compared to IPG). Each RTT the SSVP source calculates the ratio of the number of control packets (received within current RTT) that indicate congestion 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:

$$IPG_{i+1} = \frac{IPG_i}{\beta} \quad (4)$$

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

$$IPG_{i+1} = \frac{1}{1+\alpha} IPG_i \quad (5)$$

The *congestion level* threshold has been set experimentally to 0.005. Further details of SSVP can be found in [11].

In the sequel, we derive a throughput model for SSVP flows. Consider an i^{th} SSVP source transmitting m packets with packet lengths S_{i1}, S_{i2}, …, S_{im} during a time period T, where S_{ij} represents the j^{th} packet of the i^{th} flow. The average throughput of a single SSVP flow i is given by:

$$throughput_i = \frac{1}{T}\sum_{j=1}^{m} S_{ij} = \frac{m\overline{S}_i}{T} \quad (6)$$

where $\overline{S}_i$ denotes the average packet length for the i^{th} flow. We further model flow throughput assuming a fixed packet length S_i and a measurement period of one RTT, as SSVP maintains a constant IPG within a certain RTT. We define $k_i(t)$ as a function of IPG_i:

$$k_i(t) = \frac{RTT(t)}{IPG_i(t)} \quad (7)$$

representing the number of packets transmitted by connection i within an RTT. Based on equations (6) and (7), the throughput rate at time t is given by:

$$throughput_i(t) = \frac{k_i(t)\, S_i}{RTT(t)} \quad (8)$$

Combining equations (7) and (8), we obtain the instantaneous transmission rate for the i^{th} flow:

$$R_i(t) = \frac{S_i}{IPG_i(t)} \quad (9)$$

inline with equation (3). Consider n SSVP flows in the system. System throughput at time t is defined as:

$$throughput(t) = \sum_{i=1}^{n} \frac{k_i(t)\, S_i}{RTT(t)} = \frac{\overline{S}}{RTT(t)} \sum_{i=1}^{n} k_i(t) = \frac{K(t)\,\overline{S}}{RTT(t)} \quad (10)$$

where $K(t)$ and $\overline{S}$ represent the aggregated number of packets and the average packet length for all system flows, respectively. We exploit this system throughput model in the following section, where we explore the dynamics of AIMD congestion control.

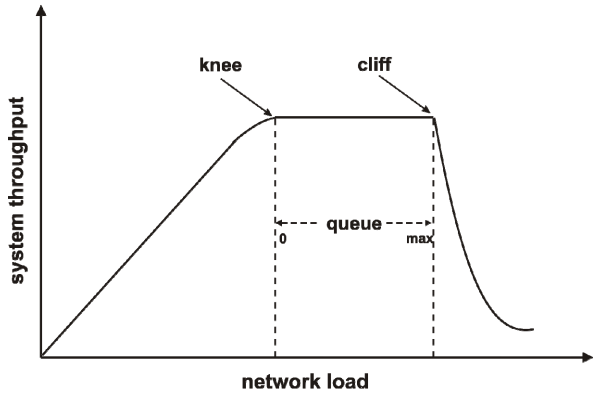


Fig. 1. System Throughput vs. Load.

IV. Optimization of AIMD Congestion Control

A. Dynamics of AIMD Congestion Control

The general patterns of system throughput as the network load increases are given in [4]. Throughput is generally increased in proportion to the load, until the load has reached the bottleneck capacity. At this point, called *knee*, throughput rate is maximum, and if the load continues to increase, a queue is being built up at the bottleneck buffer. At the time the buffer has overflowed, the network has reached the point *cliff*, where congestion may cause packet loss and subsequently diminish the throughput rate. Fig. 1 illustrates this throughput-load relation.

Based on the concepts of *knee* and *cliff*, we provide an analysis of throughput dynamics, by taking into account the role of the bottleneck queue. We assume n flows sharing a single bottleneck link with capacity B. Let RTT_0 denote the round-trip propagation delay and *qdelay(t)* the queuing delay at the bottleneck router. We rewrite equation (10) as:

$$\text{throughput}(t) = \frac{K(t)\,\overline{S}}{RTT_0 + \text{qdelay}(t)} \tag{11}$$

We consider the case where all flows probe for bandwidth and eventually the *knee* has been reached:

$$K_{knee} = RTT_0 * \frac{B}{\overline{S}} \tag{12}$$

where K_{knee} denotes the number of packets with length $\overline{S}$ that can be accommodated in the bottleneck link. Until network load has reached the *knee*, there is no *steady* queue built-up in the bottleneck buffer (i.e. RTT = RTT_0). If K(t) increases further beyond K_{knee}, the bottleneck capacity is saturated and a queue in the bottleneck buffer is gradually built up. K(t) can be now expressed as:

$$K(t) = K_{knee} + \Delta K(t) \tag{13}$$

where $\Delta K(t)$ denotes the number of packets that linger in the queue, with $\Delta K(t) > 0$. Therefore, the steady queuing delay at the bottleneck is given by:

$$\text{qdelay}(t) = \frac{\Delta K(t)\,\overline{S}}{B} \tag{14}$$

Note that increasing K(t) beyond the *knee* is not followed by throughput gains, since throughput is bounded by the physical capacity B. Indeed, with respect to equations (11)-(14) we have:

$$\text{throughput}(t) = \frac{(K_{knee} + \Delta K(t))\,\overline{S}}{RTT_0 + \text{qdelay}(t)} = \tag{15}$$

$$= \frac{(RTT_0 * \frac{B}{\overline{S}} + \frac{\text{qdelay}(t) * B}{\overline{S}})\,\overline{S}}{RTT_0 + \text{qdelay}(t)} = B$$

System throughput remains stable despite the increase of K(t), since *qdelay(t)* in the denominator of equation (15) grows as well. K(t) may continue increasing until reaching the *cliff*, where the queue occupies the whole buffer size:

$$K_{cliff} = (RTT_0 + \text{qdelay}_{max}) * \frac{B}{\overline{S}} \tag{16}$$

Potential packet loss due to buffer overflow triggers a multiplicative decrease by the AIMD source, which adjusts the transmission rate accordingly.

We therefore showed that increasing the network load beyond the *knee* does not increase throughput; only queuing delay is increased. Furthermore, our analysis indicates that some queue built-up is inevitable in order to provide the fairness-oriented AIMD algorithm with a sufficient operating scope, where the bottleneck bandwidth is fully exploited and system throughput is stable. More precisely, although multiplicative decrease is necessary to converge to fairness [4], throughput performance and smoothness are not necessarily compromised, as long as the system operates between the *knee* and the *cliff*.

Ideally, the configuration of β should prevent the system from operating below the *knee*, where a fraction of the available bandwidth is not utilized, and smoothness is degraded as throughput fluctuates. Meanwhile, downward adjustments to the *knee* maintain adequate AIMD oscillation allowing fast convergence to the fairness point [17]. Therefore, the decrease ratio β should be optimally set to K_{knee} / K_{cliff}, so that upon packet loss (with the assumption it occurs at the *cliff*) the system will be adjusted downward to the *knee*. Combining equations (12) and (16), we derive the optimal adjustment of multiplicative decrease ratio:

$$\beta = \frac{K_{knee}}{K_{cliff}} = \frac{RTT_0 * \frac{B}{\overline{S}}}{(RTT_0 + \text{qdelay}_{max}) * \frac{B}{\overline{S}}} = \tag{17}$$

$$= \frac{RTT_0 * \frac{B}{\overline{S}}}{RTT_0 * \frac{B}{\overline{S}} + bufSize} = \frac{1}{1 + \frac{bufSize * \overline{S}}{RTT_0 * B}}$$

where *bufSize* denotes the bottleneck buffer size in packets. With equation (17) bandwidth efficiency can be achieved along with smoothness and fairness. We observe that $\beta = 0.5$ when the buffer size is equal to the *bandwidth-delay* product (BDP). Equation (17) corroborates that the adjustment of the decrease ratio depends on the network settings. In order to maintain TCP-friendliness, the additive increase rate should

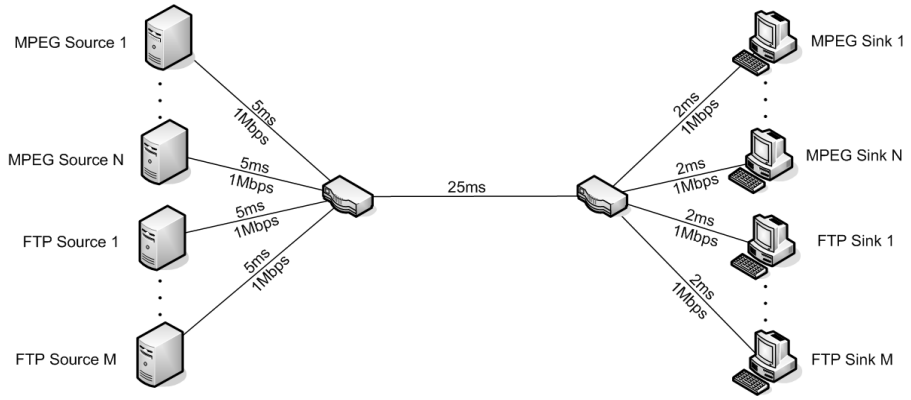


Fig. 2. Simulation Topology.

be adjusted according to equation (2), after β has been determined by equation (17).

Assume the system operates between the *knee* and the *cliff* in equilibrium, where the overall system throughput is kept at maximum, and hence is stable and smooth. Does this mean that each end user will observe a smooth throughput? Authors in [18] show that the smoothness of individual flows is directly related to the short-term fairness. With sufficient AIMD adjustments, the system can quickly converge to fairness. Given the condition of a smooth system throughput, the throughput achieved by each flow will be also smooth throughout the connection. From the perspective of an individual flow, a multiplicative decrease of $k_i(t)$ does not necessarily affect flow throughput:

$$\text{throughput}_i(t) = \frac{k_i(t)\, S_i}{RTT_0 + \text{qdelay}(t)} \tag{18}$$

if *qdelay(t)* decreases correspondingly, because of the downward adjustment of the system.

B. SSVP with Adaptive AIMD Parameters

Based on the observations in subsection IV.A, we optimize SSVP's congestion control, within the framework of bandwidth efficiency, smoothness and fairness. SSVP continuously monitors RTT and RTT_{min}, which represent the current and minimum RTT, respectively. RTT_{min} corresponds to the round-trip propagation delay RTT_0. If the system operates above the *knee*, the total number of packets sent in each RTT is expressed as:

$$K(t) = (RTT_0 + \text{qdelay}(t)) * \frac{B}{S} = RTT(t) * \frac{B}{S} \tag{19}$$

SSVP's congestion control is complemented with the adaptive adjustment of decrease ratio in order to prevent the system from operating below the *knee*. Upon the detection of packet loss, β is derived by:

$$\beta = \frac{K_{knee}}{K(t)} = \frac{RTT_0 * \frac{B}{S}}{(RTT_0 + \text{qdelay}(t)) * \frac{B}{S}} = \frac{RTT_{min}}{RTT(t)} \tag{20}$$

Combining equations (4) and (20), the sending rate of each SSVP flow is decreased via the multiplicative increase of IPG:

$$IPG_{i+1} = \frac{IPG_i\, RTT(t)}{RTT_{min}} \tag{21}$$

allowing a downward adjustment to the *knee*, where *qdelay(t)* = 0 and $K(t) = K_{knee}$. The specific value of RTT(t) that affects the decrease ratio in equations (20) and (21) is normally observed before the buffer overflows and congestion control is being triggered (i.e. $K(t) = K_{cliff}$ and $qdelay(t) = qdelay_{max}$). Essentially, RTT(t) is an estimation of the maximum RTT, i.e. $RTT_0 + \text{qdelay}_{max}$. In response to the adaptive decrease ratio, SSVP enforces a dynamic adjustment of the increase rate α by applying the current value of β to the TCP-friendly equation (2). Consequently, the protocol determines the proper increase rate to maintain friendliness with competing TCP flows.

One concern with congestion control based on RTT measurements is the accuracy of RTT_{min}, since it affects the computation of β in equation (20). While RTT_{min} should reflect the round-trip propagation delay, it can be potentially overestimated, for example if a new flow joins a network with a persistent queue. An overestimated RTT_{min} may result in a higher β, and subsequently cause unfairness to other flows. RTT_{min} overestimation (due to the advent of new flows) is a well-known problem for *TCP Vegas* [2], as its congestion control target is to stabilize at a non-empty queue. The proposed adaptive control, however, does not suffer from this problem, since the system periodically adjusts downwards to the *knee* (i.e. empty queue), where round-trip propagation delay (RTT_{min}) can be accurately measured.

V. Evaluation Methodology

A. Experimental Settings

The evaluation plan was implemented on the *NS-2* network simulator. Simulations were conducted on a single-bottleneck *dumbbell* topology (Fig. 2) with drop-tail routers and a round-trip link delay of 64 ms. The bottleneck link is shared by competing MPEG and FTP connections and its capacity is configured depending on the experiment performed. All the simulated FTP flows run over TCP Reno. We set the packet size to 1000 bytes for all system flows and the maximum congestion window to 64 KB for all TCP connections. We used drop-tail routers with the buffer size set roughly to half of the BDP. The duration of each

simulation is 200 sec. All the results are collected after 2 sec in order to avoid the skew introduced by the startup effect.

In order to simulate MPEG traffic, we developed an *MPEG-4 Traffic Generator*. The traffic generated closely matches the statistical characteristics of an original MPEG-4 video trace. The compression initiates by encoding a single I (*intra* picture) frame, followed by a group of P (*predictive*) and B (*bidirectional*) frames. P frames carry the signal difference between the previous frame and motion vectors, while B frames are interpolated; the encoding is based on the previous and the next frame. We used 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.

B. Performance Metrics

Since the simulation includes competing MPEG and FTP connections, some of our performance metrics may be applied separately to the MPEG and FTP traffic. *Throughput* is used to measure the efficiency in link utilization. Following the metric in [18], we use *Coefficient of Variation* (CoV) in order to gauge the throughput smoothness experienced by flow *i*:

$$\mathrm{CoV}_i = \frac{\sqrt{E_t\{\mathrm{throughput}_i^2(t)\} - E_t\{\mathrm{throughput}_i(t)\}^2}}{E_t\{\mathrm{throughput}_i(t)\}}$$

where $E_t\{\}$ denotes the computation of the mean along time. For the i^{th} flow, its instantaneous throughput around time t, $throughput_i(t)$, is sampled at a time scale of a few RTTs throughout the entire connection. In our simulations, the sampling period is set to 150 ms. For a system with multiple flows, we demonstrate the average of CoVs of all flows.

Long-term fairness is measured by the *Fairness Index*, derived from the formula given in [4], and defined as:

$$\text{Fairness Index} = \frac{(\sum_{i=1}^{n} \mathrm{Throughput}_i)^2}{n\sum_{i=1}^{n} \mathrm{Throughput}_i^2}$$

where $Throughput_i$ is the throughput of the i^{th} flow and n is the total number of flows. As a supplementary fairness metric, we use *Worst-Case Fairness*, in order to conduct a *worst-case* analysis and provide a tight bound on fairness:

$$\text{Worst - Case Fairness} = \frac{\min_{1\le i\le n}(\mathrm{throughput}_i)}{\max_{1\le i\le n}(\mathrm{throughput}_i)}$$

Worst-Case Fairness is more sensitive to the system unfairness to a small fraction of flows and ranges in [0, 1] (with 1 representing the absolute fairness).

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. In this context, we define a metric for the performance evaluation on video delivery, called *Video Delivery Index*, which captures the joint effect of jitter and packet loss on perceptual 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:

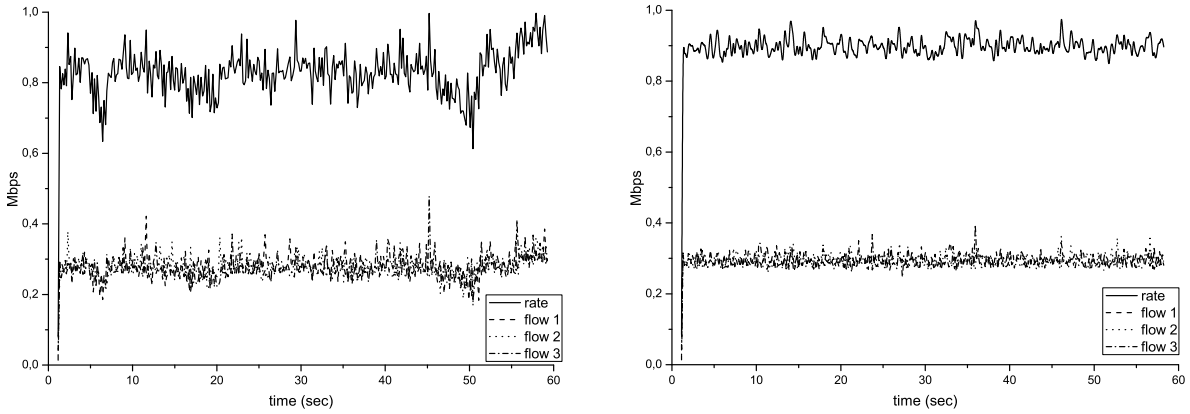
$$\text{Video Delivery Index} = \frac{\#\,\text{jitter\_free packets}}{\#\,\text{sent\_packets}} \le 1$$

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 Delivery Index* of each MPEG flow.

VI. Performance Evaluation

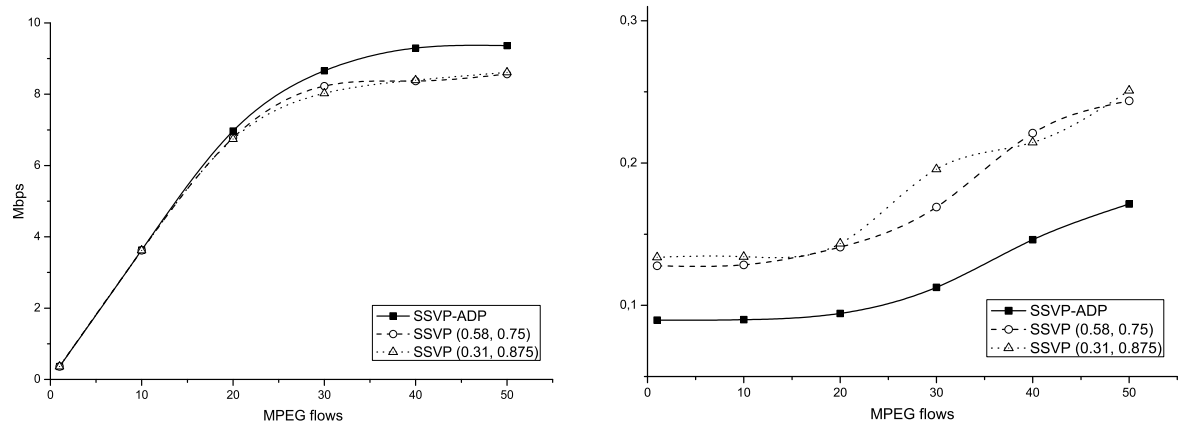
In this section, we demonstrate performance studies based on selected simulation results. First, we simulated three MPEG flows of (i) SSVP (0.31, 0.875) (i.e. SSVP with static parameters $\alpha = 0.31$, $\beta = 0.75$) and (ii) SSVP with adaptive congestion control parameters (SSVP-ADP). We set the bottleneck capacity to 1 Mbps in our simulation topology. Fig. 3 illustrates the variation of sending rates for the 3 individual flows and for the aggregated rate, during the first 60 sec of the experiment. The transmission rate is averaged over 150 ms intervals, each one including a few RTTs for the simulated topology. According to Fig. 3a, SSVP flows experience perceptible variations in their transmission rate, despite the selection of AIMD parameters (i.e. 0.31, 0.875) which favor smoothness [19]. Consequently, the aggregated sending rate fluctuates between 800 Kbps and 1 Mbps, since occasionally the available bandwidth remains underutilized. In contrast, SSVP-ADP's adaptive congestion control (Fig. 3b) enables the system to operate between the *knee* and the *cliff*, utilizing the available network resources efficiently and alleviating the undesirable effects of AIMD's rapid backward and graduated upward adjustments. Essentially, SSVP-ADP confines the fluctuations in the sending rate, maintaining a smoothed flow that optimizes media delivery and playback on the receiver. In order to quantify the gains attained in terms of smoothness, we measured CoV in the protocol sending rate in both cases: $\mathrm{CoV}_{\mathrm{SSVP}} = 0.0868$ and $\mathrm{CoV}_{\mathrm{SSVP\text{-}ADP}} = 0.0581$. We note that a lower CoV indicates a lower variation in sending rates, and consequently higher smoothness.

We conducted additional simulations to assess the efficiency of the proposed adaptive control in terms of bandwidth utilization, smoothness, and fairness. In this context, we simulated a diverse range of MPEG flows (10-50 flows) of (i) SSVP (0.31, 0.875), (ii) SSVP (0.58, 0.75), and (iii) SSVP-ADP, competing with 5 FTP connections of TCP Reno, successively. The bottleneck capacity was set to 10 Mbps. We demonstrate the corresponding *Throughput*, *CoV*, *Fairness Index* and *Worst-Case Fairness* results for the



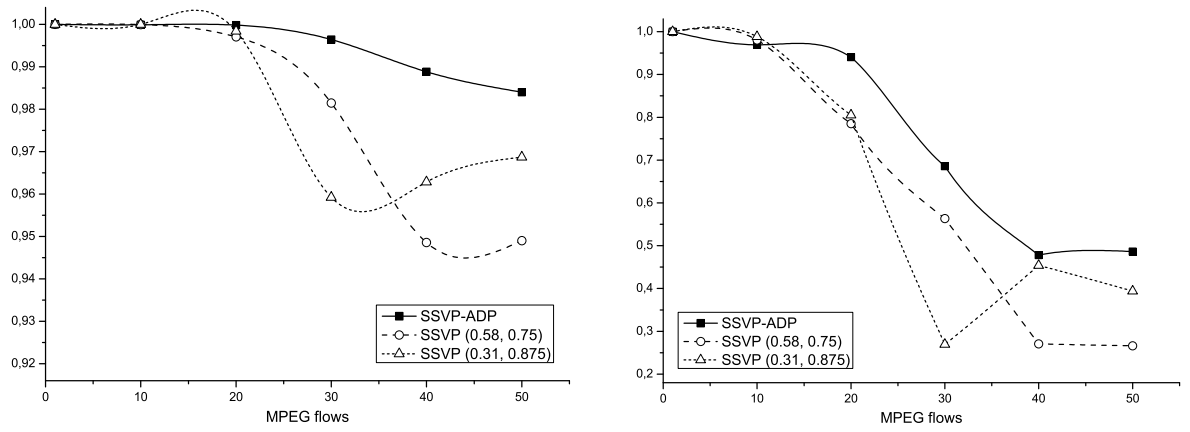
(a) SSVP (0.31, 0.875) (b) SSVP-ADP

Fig. 3. Instantaneous Transmission Rate.



(a) Throughput of MPEG flows (b) CoV of MPEG flows

Fig. 4. Protocol Performance.

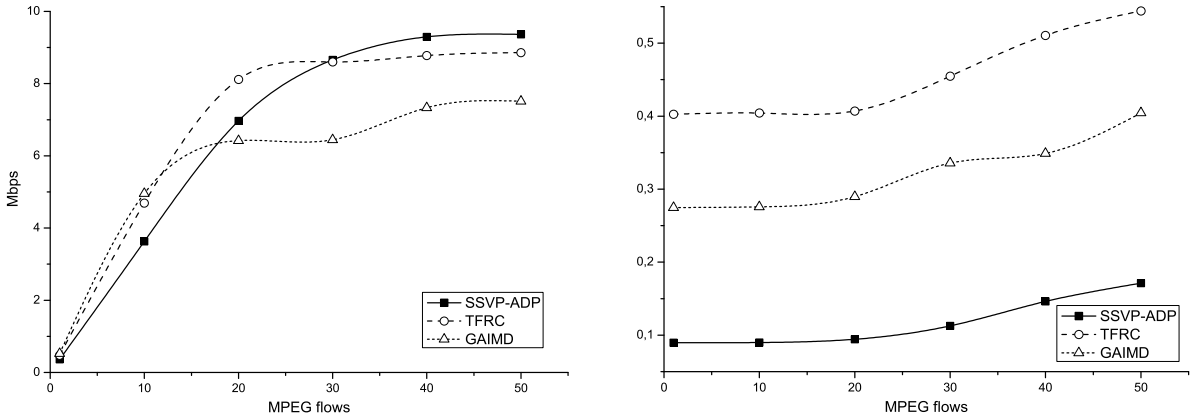


(a) Fairness Index (b) Worst-Case Fairness

Fig. 5. Fairness.

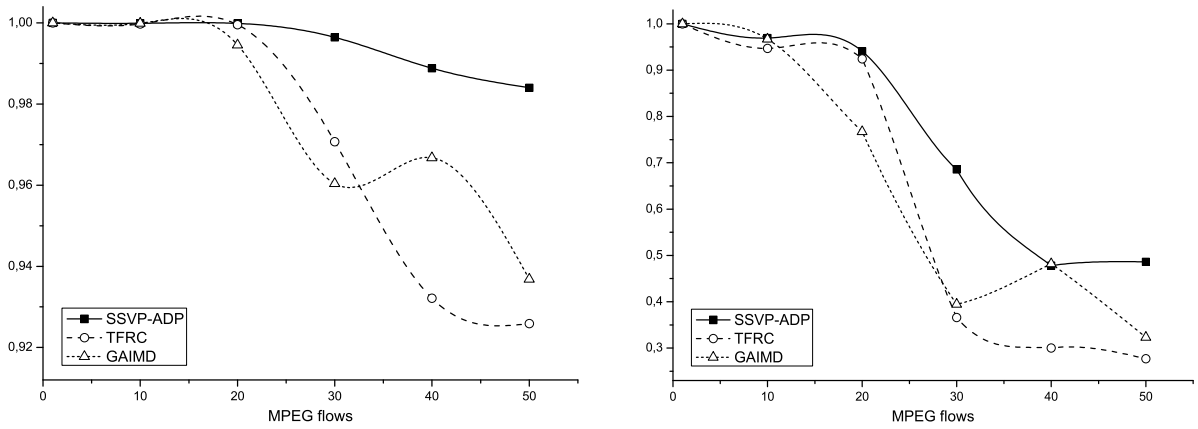
MPEG flows (Figs. 4-5). We selected the TCP-friendly pairs (0.31, 0.875) and (0.58, 0.75) for SSVP, since an AIMD flow with a small α and large β is less sensitive to bandwidth variation and subsequently exhibits higher smoothness. On the contrary, standard TCP's (1, 0.5) pair is commonly responsible for abrupt rate reductions, which may cause perceptible interruptions on media playback. According to [18], a GAIMD (0.31, 0.875) flow achieves approximately half the CoV of a TCP flow at low loss rate.

SSVP-ADP (Fig. 4a) enables the system to consistently operate above the *knee*, as depicted by the nearly optimum bandwidth utilization in the case of 30-50 flows (also considering the bandwidth allocated to the interfering TCP traffic). This is primarily the effect of the adaptive decrease ratio adopted by SSVP-ADP. Therefore, an AIMD protocol



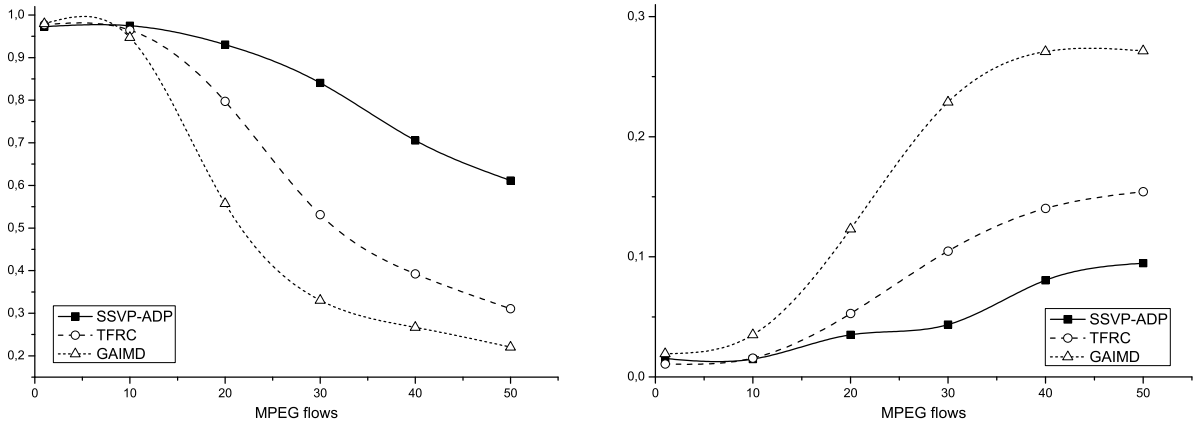
(a) Throughput of MPEG flows (b) CoV of MPEG flows

Fig. 6. Protocol Performance.



(a) Fairness Index (b) Worst-Case Fairness

Figure 7. Fairness.



(a) Video Delivery Index (b) Delayed Packets Rate

Figure 8. Performance on Video Delivery.

may rely on multiplicative decrease without sacrificing system throughput, as long as the system operates between the *knee* and the *cliff*. On the other hand, both SSVP variants with static congestion control parameters yield notably inferior bandwidth utilization for high link-multiplexing. Fig. 4b reveals that SSVP-ADP's transmission rates exhibit oscillations of much lower magnitude in average, achieving the desired smoothness required by most media-streaming applications. The adaptive congestion control reduces transmission rate fluctuations, abolishing the damage of static multiplicative decrease on flow smoothness.

Along with the reported throughput and smoothness gains, Fig. 5 demonstrates that fairness with SSVP-ADP is not compromised, since the additive increase rate α is adaptively

adjusted as well. Hence, the (α, β) pair always satisfies the TCP-friendly equation, and subsequently intra-protocol fairness is attained. Further, we identify perceptible gains for SSVP-ADP in comparison with static SSVP, either in terms of *Fairness Index* or *Worst-Case Fairness*. Depending on network conditions, SSVP-ADP's increase parameter α may be higher than 0.31 or even 0.58, allowing the system to converge faster to the fairness point.

We conclude our performance studies with the assessment of SSVP-ADP versus TFRC and GAIMD. Both TFRC and GAIMD are designed to achieve efficiency and smoothness in media delivery over a wide range of network and session dynamics. Along these lines, we simulated a wide range of MPEG flows (1-50) over (i) SSVP-ADP, (ii) TFRC, and (iii) GAIMD, competing with 5 FTP connections over TCP Reno, successively. The parameters of the simulation topology (i.e. bottleneck capacity and buffer size) are as described in the previous scenario. We measured *Throughput*, *CoV*, *Fairness Index*, *Worst-Case Fairness* and *Video Delivery Index* for the MPEG flows, and we additionally demonstrate statistics of delayed packets which is an influencing factor for perceptual video quality (Figs. 6-8).

According to Fig. 6a, SSVP-ADP exhibits a dual behavior depending on the available bandwidth. In lightly loaded environments, the protocol allocates network resources similar to SSVP's conservative additive increase rate α = 0.31 (assuming that in the absence of packet loss, α has not been adjusted significantly). When the bottleneck capacity is saturated, the adaptive congestion control enables the protocol to achieve nearly maximum throughput performance, outperforming both TFRC and GAIMD (Fig. 6a). In particular, TFRC occasionally does not obtain accurate estimates of the loss event rate, causing an inappropriate equation-based recovery, since TFRC's throughput model is sensitive to packet loss. GAIMD also fails to adapt to the network dynamics, since it relies on pre-determined congestion control parameters. Queue dynamics for GAIMD reveals that bottleneck queue remains occasionally idle, causing bandwidth underutilization.

Fig. 6b illustrates the corresponding CoV measurements for these protocols. While GAIMD achieves adequate levels of smoothness by simply employing a high multiplicative decrease ratio (i.e. β = 0.875), SSVP-ADP's adaptive decrease ratio results in a remarkable degree of smoothness for the protocol. On the contrary, TFRC's static downward adjustments induce variations in the throughput rates and compromise smoothness.

As shown in Fig. 7a, SSVP-ADP excels in bandwidth sharing, regardless of the level of link-multiplexing. *TCP-friendliness* is achieved by SSVP-ADP, since the adaptable pair of parameters always follows the TCP-friendly equation. The adaptive congestion control also maintains adequate AIMD oscillation allowing all the AIMD-based system flows (including both the MPEG flows and the competing TCP flows) to converge to the fairness point. The competing TCP flows can consume their fair share of the bandwidth. In addition, the remarkable levels of long-term fairness for SSVP-ADP validate the accuracy of RTT_{min} measurements obtained by the protocol (which enable the system to adjust downwards to the *knee*). On the contrary, authors in [13] show analytically that TFRC sources tend to achieve higher throughput rates than their bandwidth fair share with a consequent starvation of coexisting TCP flows. As showed in Fig. 7a, TFRC and GAIMD achieve relatively adequate levels of fairness (but still lower than that of SSVP-ADP), if assessed by the traditional *Fairness Index*. However, static multiplicative decrease can undermine even long-term fairness, if measured by *Worst-Case Fairness* (Fig. 7b). We observe that the *Worst-Case Fairness* for TFRC and GAIMD degrades abruptly, representing the significant difference between the minimum and maximum throughput rates achieved by both protocols. Therefore, the system can be notably unfair to a small fraction of flows.

In terms of video delivery, transmission rate fluctuations, increased delays, jitter and packet loss commonly deteriorate the perceived quality or fidelity of the received video content. *Video Delivery Index*, which provides means for the perceptual QoS assessment of video streams, validates the smooth and uninterrupted sending rate maintained by SSVP-ADP, which is slightly affected by contention (Fig. 8a). Essentially, the integrated adaptive control alleviates most of the impairments caused by the oscillatory behavior of AIMD, optimizing the experience of multimedia-users. On the other hand, TFRC's and GAIMD's static downward adjustments induce considerable variations in the sending rate, with the effect of jitter becoming evident to the end-user. We observe that the rate of delayed packets (i.e. packets that cause interruptions of user perceptions) increases in proportion to the level of contention for GAIMD, and in part for TFRC (Fig. 8b). On the contrary, SSVP-ADP effectively enforces an upper bound to the magnitude of delay variation, providing a possible guarantee for streaming applications that can efficiently operate within this QoS provision.

VII. Conclusions and Future Work

We have presented an adaptive AIMD congestion control in order to effectively combine smooth transmission patterns with bandwidth efficiency and fairness. Through the adaptive adjustment of the decrease ratio, the system effectively operates between the *knee* and the *cliff*, achieving smoothness and nearly optimum link utilization throughout the connection. Exploiting the extended AIMD dynamics, we refined SSVP's congestion control by introducing AIMD parameters adaptable to current network conditions. The additive increase step is adjusted accordingly to achieve TCP-friendliness. Our performance studies have validated the efficiency of the proposed mechanism. Eventually, media-streaming applications have more incentives to rely on AIMD congestion control, within the context of best-effort networks. Future work will be focused on the detailed analysis of the impact of unsynchronized and random multiplicative decrease on throughput smoothness and queue length.

References

[1] D. Bansal and H. Balakrishnan, "Binomial Congestion Control Algorithms", in *Proc. IEEE INFOCOM 2001*, Anchorage, Alaska, USA, April 2001.

[2] L. Brakmo and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet", *IEEE Journal on Selected Areas in Communications* (JSAC), vol. 13, no. 8, October 1995, pp. 1465–1480.

[3] L. Cai, X. Shen, J. Pan, and J. Mark, "Performance analysis of TCP-friendly AIMD algorithms for multimedia applications", *IEEE Transactions on Multimedia*, vol. 7, no. 2, April 2005, pp. 339–355.
[4] D. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks", *Journal of Computer Networks and ISDN*, vol. 17, no. 1, June 1989, pp. 1–14.
[5] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet", *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, August 1999, pp. 458–472.
[6] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-Based Congestion Control for Unicast Applications", in *Proc. ACM SIGCOMM 2000*, Stockholm, Sweden, August 2000.
[7] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, August 1993, pp. 397–413.
[8] S. Gorinsky, M. Georg, M. Podlesny, and C. Jechlitschek, "A Theory of Load Adjustments and its Implications for Congestion Control", *Journal of Internet Engineering*, Klidarithmos Press, vol. 1, no. 2, October 2007, pp. 82–93.
[9] E. Kohler, M. Handley, and S. Floyd, "Designing DCCP: Congestion control without reliability", in *Proc. ACM SIGCOMM 2006*, Piza, Italy, September 2006.
[10] L. Mamatas, T. Harks, and V. Tsaoussidis, "Approaches to Congestion Control in Packet Networks", *Journal of Internet Engineering*, Klidarithmos Press, vol. 1, no. 1, February 2007, pp. 22–33.
[11] P. Papadimitriou and V. Tsaoussidis, "SSVP: A Congestion Control Scheme for Real-Time Video Streaming", *Computer Networks*, Elsevier, vol. 51, no. 15, October 2007, pp. 4377–4395.
[12] K. Ramakrishnan and S. Floyd, "A proposal to add explicit congestion notification (ECN) to IP", *RFC 2481*, January 1999.
[13] R. Rejaie, M. Handley, and D. Estrin, "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet", in *Proc. IEEE INFOCOM 1999*, New York, USA, March 1999.
[14] I. Rhee and L. Xu, "Limitations of Equation-based Congestion Control", in *Proc. ACM SIGCOMM 2005*, Philadelphia, USA, August 2005.
[15] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", *RFC 2001*, January 1997.
[16] V. Tsaoussidis and C. Zhang, "TCP Real: Receiver-oriented congestion control, *Computer Networks*, Elsevier, vol. 40, no. 4, November 2002, pp. 477–497.
[17] V. Tsaoussidis and C. Zhang, "The Dynamics of Responsiveness and Smoothness in Heterogeneous Networks", *IEEE Journal on Selected Areas in Communications* (JSAC), vol. 23, no. 6, June 2005, pp. 1178–1189.
[18] Y. R. Yang, M. S. Kim, and S. S. Lam, "Transient Behaviors of TCP-friendly Congestion Control Protocols", in *Proc. IEEE INFOCOM 2001*, Anchorage, Alaska, USA, April 2001.
[19] Y. R. Yang and S. S. Lam, "General AIMD Congestion Control", in *Proc. 8th IEEE Int/nal Conference on Network Protocols* (ICNP), Osaka, Japan, November 2000.

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, UK. He is currently a Ph.D. Candidate in Electrical and Computer Engineering in Democritus University of Thrace, Greece. Panagiotis 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 satellite/space communications.

Chi Zhang is a Senior Kernel Engineer at Juniper Networks. He was an Assistant Professor of Computer Science at Florida International University from 2003 to 2006. He received the B.E. degree in Electronic Engineering from Shanghai Jiao Tong University, China, in 1996, and the Ph.D. degree in Computer Science from Northeastern University, Boston, MA, in 2003.

Dr. Zhang's research interests lie in the areas of network protocols, congestion control, mobile computing and QoS. He has published 28 papers and issued 1 US patent. He served on a number of program committees of networking conferences, and is on the editorial board of the Journal of Internet Engineering. Dr. Zhang received the runner-up award in the 7th IEEE Symposium on Computers and Communications (ISCC 2002), and is a member of the Phi Kappa Phi Honor Society.