Telecommun Syst
DOI 10.1007/s11235-009-9203-8

End-to-end loss differentiation for video streaming with wireless link errors

Panagiotis Papadimitriou · Vassilis Tsaoussidis · Chi Zhang

Abstract Video delivery in heterogeneous wired/wireless networks is challenging, since link errors commonly degrade throughput performance, smoothness, and eventually impair the playback quality. We present an end-to-end *Loss Differentiation Mechanism* (LDA) which effectively decouples congestion from wireless loss in order to abolish the damage of error-induced multiplicative decrease on flow throughput and smoothness. The proposed LDA relies on *Round Trip Time* measurements to estimate the queuing delay and determine the appropriate error-recovery strategy. This mechanism can be easily adapted and incorporated into existing *Additive Increase Multiplicative Decrease* (AIMD) protocols, enabling them to utilize the available resources more efficiently. In this context, we incorporate the LDA into AIMD-based *Scalable Streaming Video Protocol* (SSVP), an end-to-end TCP-friendly protocol optimized for video streaming applications. Based on simulation results, we show that the combined approach provides the desired functionality to bind operationally wired and wireless links, within the framework of bandwidth efficiency, smoothness, and fairness.

Keywords End-to-end protocols · Wireless networks · Video streaming · Quality of service

P. Papadimitriou (✉)
Lancaster University, Lancaster, UK
e-mail: p.papadimitriou@lancaster.ac.uk

V. Tsaoussidis
Democritus University of Thrace, Xanthi, Greece
e-mail: vtsaousi@ee.duth.gr

C. Zhang
Juniper Networks, Sunnyvale, CA, USA
e-mail: chizhang@juniper.net

Published online: 15 October 2009

1 Introduction

An increasing demand for multimedia data delivery coupled with reliance on best-effort networks, such as the Internet, has spurred interest in efficient transport solutions for multimedia streams. Video streaming, in particular, is comparatively intolerant to delay and variations of throughput and delay. Unlike bulk-data transfers, video delivery requires a minimum and continuous bandwidth guarantee. Video quality is also affected by reliability factors, such as packet drops due to congestion or link errors. In MPEG, for example, dropping packets from an independently encoded *I* (*intra* picture) frame causes the following dependent *P* (*predictive*), and *B* (*bidirectional*) frames not to be fully decodable. In practice, inter-frame dependencies may convert a 3% packet loss rate up to a 30% frame loss rate [4]. Generally, 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.

Today's multimedia applications are expected to run in physically heterogeneous environments composed of both wired and wireless components. Wireless links exhibit distinct characteristics, such as limited bandwidth, bit errors and potential handoff operations. Bit errors typically occur when the signal to interference and noise ratio is not high enough to decode information correctly. Furthermore, wireless channels are hard to model and predict, and designing an error-free communication link generally entails sacrificing significant capacity. QoS requirements in wireless networking essentially remain stringent and complicated, taking additionally into account the influencing mobile device characteristics and limitations. For example, a considerable number of mobile devices offer limited buffer capacities, being unable to smooth the fluctuations in the receiving rate. In

this case, the task of smooth delivery is primarily delegated to the transport protocol.

Transmission Control Protocol (TCP) is basically designed to provide a reliable service for wired Internet. The *Additive Increase Multiplicative Decrease* (AIMD) algorithm [10], incorporated into standard TCP versions, reduces the sending window by half to avoid persistent packet losses when the demand of competing flows exceeds the channel bandwidth, and increases the window additively every *Round Trip Time* (RTT) to exploit the available bandwidth. AIMD is also designed to converge to fairness. However, most existing TCP mechanisms do not satisfy the need for universal functionality in heterogeneous wired/wireless environments, since they do not flexibly adjust the rate and pattern of the transmitted multimedia streams to the characteristics of the end-to-end network path. Authors in [33] outline three major shortfalls of TCP: (i) ineffective bandwidth utilization, (ii) unnecessary congestion-oriented responses to wireless link errors (e.g. fading channels) and operations (e.g. handoffs), and (iii) wasteful window adjustments over asymmetric, low-bandwidth reverse paths. Furthermore, TCP's process of probing for bandwidth and reacting to observed congestion causes oscillations to the achievable transmission rate. TCP may also introduce arbitrary delays, since it enforces reliability and in-order delivery. In response to standard TCP limitations, several TCP protocol extensions [2, 5, 16, 20, 23, 32, 34, 37] have emerged providing more effective bandwidth utilization and sophisticated mechanisms for congestion control.

User Datagram Protocol (UDP) has been widely used instead of TCP by media-streaming applications. UDP lacks all basic mechanisms for error recovery and flow/congestion control. Thus, it allows for transmission attempts at application speed. That said, UDP cannot guarantee reliability, and certainly is not able to deal with network delays either. In [25] we showed that UDP may perform worse than TCP in several occasions.

Since TCP is not preferred by multimedia applications and UDP poses a threat to network stability, rate-based congestion control has become an attractive alternative. Rate-based mechanisms directly control the transmission rate of the connection, based on either measurements taken at the end host or feedback from the network [13, 24, 28]. Avoiding the burstiness occasionally induced by the window-based mechanisms, rate-based protocols generate a smooth data flow by spreading the data transmission across a time interval. Therefore, rate-based mechanisms compose plausible candidates when smooth delivery is a primary objective. However, the challenge does not lie in simply achieving smoothness, but rather providing adequate efficiency and resilience to the inherent characteristics of wireless links. More precisely, a suitable protocol for wired/wireless networks should be able to detect the nature of the errors that result in packet loss in order to determine the appropriate error-recovery strategy. Based on such an approach, the sender would not be obliged to reduce its transmission rate in the event of a wireless error or handoff.

In contrast to transport-layer solutions, a series of independent mechanisms have been proposed, which normally interact with the transport protocol and provide reliable transmission over wireless links [1, 11, 18, 19]. Most of them operate on link layer and generally are considered more efficient than physical-layer techniques, such as spread-spectrum and OFDM modulation or channel coding. However, link-layer approaches may degrade performance, especially in the presence of highly variable error rates. Local error recovery may alter the characteristics of the network affecting the functionality of higher layer protocols. For example, local retransmission could result in packet reordering or in considerable RTT fluctuations. In addition, concurrent responses from both local and end-to-end error control may result in undesirable interactions, causing inefficiencies and potentially instability. Considering real-time traffic where data packets bear information with a limited useful lifetime, retransmissions are often a wasted effort. In such conditions, unfruitful retransmissions deliver delayed packets which are either discarded, or at the worst they obstruct the proper reconstruction of oncoming packets.

Our objective is to combine an efficient transport protocol with a mechanism that provides robustness and resilience to link errors, achieving uninterrupted and smooth video delivery in wired/wireless environments. In this context, we present an end-to-end mechanism that effectively decouples congestion from wireless loss to abolish the damage of error-induced multiplicative decrease on flow throughput and smoothness. This work builds on [26] extending the analysis, discussion, and simulation results. Packet loss is characterized as wireless or congestive based on the measured queuing delay. The mechanism relies on RTT measurements to estimate current queue length and subsequently determine the appropriate error-recovery strategy. We note that the proposed loss differentiation scheme does not require any modifications in the network infrastructure or the underlying transport protocol. This mechanism can be easily adapted and incorporated into existing transport-layer solutions, spanning from TCP to rate-based AIMD protocols. Since we focus on streaming video delivery, we incorporate the proposed mechanism into *Scalable Streaming Video Protocol* (SSVP) [24] and evaluate its efficiency in terms of bandwidth utilization, video playback quality, and fairness. SSVP is an AIMD-oriented rate control scheme optimized for video streaming applications.

We organize the remainder of the paper as follows. Section 2 provides an overview of related work. In Sect. 3 we elaborate on the proposed loss differentiation mechanism, and we further incorporate it into SSVP in Sect. 4. Section 5

provides a detailed description of the experimental environment, followed by Sect. 6, which includes extensive performance studies based on simulations. Finally, in the last section we highlight our conclusions.

2 Related work

In the sequel, we provide a taxonomy of the most prominent approaches that tackle the limitations of media delivery in wired/wireless networks. In this context, we discuss separately end-to-end enhancements for heterogeneous environments, transport-layer solutions for efficient media delivery, and selected mechanisms operating on the link layer.

2.1 End-to-end enhancements for wireless links

Numerous proposals have been presented in order to improve transport-layer efficiency over wireless links [1, 3, 6–9, 16, 18, 23, 29, 32]. Most related research efforts focus on bulk-data transmission and are usually advertised as enhanced TCP versions. *TCP Westwood* [17, 23] is a TCP-friendly 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 [30] and congestion window after a congestion episode. The protocol incorporates a recovery mechanism which avoids the blind halving of the sending rate of TCP Reno after packet losses and enables Westwood to achieve high link-utilization in the presence of wireless errors. In [24, 25] we showed that TCP Westwood tends to underestimate the available resources, since the estimation filter is slow, needing time to converge to the available bandwidth. TCP Westwood+ is a recent extension of TCP Westwood, based on an *Additive Increase/Adaptive Decrease* 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.

In TCP-Vegas [5], for every RTT the sender calculates the throughput rate which subsequently is compared to an expected rate. Depending on the outcome of this comparison the transmission rate of the sender is adjusted accordingly. More precisely, whenever an acknowledgement is received, TCP Vegas computes the quantity:

$$diff = \left(\frac{cwnd}{baseRTT} - \frac{cwnd}{RTT} \right) \times baseRTT$$

where *cwnd* is the congestion window and *baseRTT* is the minimum RTT measured by the TCP source. If $diff > 3$, the sender infers congestion and decreases the transmission rate. Any loss experienced while $diff < 1$ is interpreted as wireless. If $1 \leq diff \leq 3$, the sending rate remains temporarily unaffected.

Selected end-to-end loss differentiation algorithms are applied to *TCP-friendly Rate Control* (TFRC) and their efficiency is analyzed in [8]. These algorithms include *ZigZag*, *Biaz* and *Spike*. *ZigZag* and *Biaz* detect the nature of loss based on packet inter-arrival times at the receiver, while *Spike* measures one-way delay to decouple wireless loss from congestion. The measured one-way delay is used to identify the *spike* state. More precisely, if the connection is in the *spike* state, any experienced packet losses are interpreted as congestive; otherwise, losses are assumed to be wireless. [8] also presents a hybrid loss differentiation algorithm, namely ZBS, which dynamically switches between *ZigZag*, *Biaz* and *Spike* according to the prevailing network conditions. However, the accuracy of ZBS discriminator is highly dependent on the number of flows sharing the bottleneck link and yields high wireless and congestion misclassifications [3]. Furthermore, inferring a specific behavior from inter-arrival times or packet pair can be inaccurate, due to the variation and complication of traffic patterns in the Internet.

Authors in [7] present and analyze the bandwidth estimation schemes implemented at the sender side of a TCP connection, including the estimation algorithms of TCP Vegas [5] and TCP Westwood. In addition, they propose *TIBET* (*Time Intervals based Bandwidth Estimation Technique*), a new bandwidth estimation scheme implemented within the TCP congestion control procedure, which enhances TCP performance over wireless links. *TCP Probing* [32] grafts a probing cycle and an *Immediate Recovery Strategy* into standard TCP in order to control effectively the throughput/overhead trade-off. *Freeze-TCP* [16] distinguishes handoffs from congestion through the use of the advertised window. *WTCP* [29] implements a rate-based congestion control replacing entirely the ACK-clocking mechanism. MULTFRC [9] 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.

2.2 Transport-layer approaches for multimedia traffic

Since TCP is rarely chosen to transport multimedia traffic over the Internet, numerous *TCP-friendly* protocols [13, 36, 37] 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 [12]. TCP-friendly congestion control maintains network stability by promptly responding to congestion and is also cooperative with other flows, while it commonly provides a smoother sending rate for multimedia applications. TFRC [13] 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:

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

where p is the steady-state loss event rate and RTO is the retransmission timeout value. 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 to predict accurately. Furthermore, 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 [35].

GAIMD [37] 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 [37] derive a simple relationship between α and β in order to be friendly to standard TCP:

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

Based on experiments, they 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 [34] is a high-throughput transport protocol that incorporates congestion avoidance mechanism in order to minimize transmission-rate gaps. The protocol approximates a receiver-oriented approach beyond the balancing trade of the parameters of additive increase and multiplicative decrease. TCP-Real introduces another parameter, namely γ, which determines the window adjustments during congestion avoidance. 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.

Rate Adaptation Protocol (RAP) [28] 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 [30]. 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 1/2 invokes abrupt rate reductions upon congestion, degrading smoothness.

Datagram Congestion Control Protocol (DCCP) [21] 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. Our proposed loss differentiation algorithm can be incorporated into DCCP to improve the existing congestion control mechanisms in wireless Internet environments.

2.3 Link-layer enhancements

There are several techniques operating on the link layer, which attempt to ameliorate the impact of wireless errors [1, 11, 18, 19]. The most remarkable implementations, which provide error correction, are *Forward Error Correction* (FEC) and *Automatic Repeat Request* (ARQ) [11]. FEC introduces added overhead to data bits in order to cope with data corruption. Corrupted packets are directly corrected, without retransmission, which is critical for lossy links with long end-to-end delays. However, the redundant information is not exploited in the absence of link errors resulting in a waste of bandwidth. Furthermore, FEC requires additional resources in CPU processing time, memory and power consumption.

On the other hand, ARQ mechanisms are invoked when packets containing bit errors cannot be corrected. In such case, the erroneous packets are discarded and a retransmission is directly triggered. Unlike FEC, ARQ allocates additional network resources only when a packet is retransmitted. The mechanism generally operates more efficiently for low bit rates. An undesirable effect of ARQ is that it may interfere with the transport protocol [1]. Furthermore, retransmission-based recovery of corrupted packets is not usually a viable solution for certain types of multimedia applications (e.g., multicast video). Generally, in multimedia applications, instead of having a packet delivered late interpolating the lost data with previously delivered data is usually preferred.

Authors in [19] recently designed a link-layer retransmission protocol, namely PP-ARQ, enabling a more sophisticated error-recovery strategy than conventional ARQ. More precisely, the receiver compactly requests the retransmission of selected portions of a packet where there are possibly bits with corruption. In response, the sender retransmits the bits and checksums for those ranges, so that the receiver can

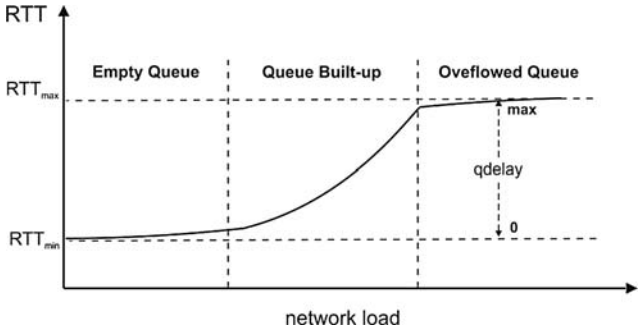


Fig. 1 RTT vs. network load

eventually be certain that all the bits in the packet are correct. The receiver's request encoding uses a dynamic programming algorithm that minimizes the expected bit overhead of communicating the feedback.

Our work is different from the existing approaches in that the proposed loss differentiation algorithm incorporates the combined dynamics of sending rates, queuing delay and link losses. As it can be seen from (1), queuing dynamics are not incorporated into existing TCP-friendly protocols. On the other hand, most of the link- layer enhancement strategies above are not discussed in the context of TCP-friendly video streaming. Without sacrificing fairness and friendliness, our algorithm is applied and evaluated in the context of video streaming over lossy links.

3 Loss differentiation algorithm

Most transport-layer mechanisms are not able to detect the cause of packet loss, invoking congestion-oriented responses to all types of losses. Apart from a wasteful rate decrease, further undesirable implications may take place since the flows that reduce their rates can be *suppressed* by competing flows that do not experience wireless loss. From the perspective of media delivery, a wireless error-prone link can induce large variations in the sending rate with an adverse effect on playback quality. In this context, we present a *Loss Differentiation Algorithm* (LDA), which allows the underlying transport protocol to distinguish congestion from random wireless loss based on queuing delay. The proposed LDA is a pure end-to-end mechanism and does not require any modifications in the network infrastructure or the underlying network protocol. It relies on RTT estimation to observe current network dynamics and detect the nature of the error.

3.1 Queue dynamics

Figure 1 illustrates the relation between network load and RTT, as the load increases. RTT_{min} denotes the round-trip propagation delay and *qdelay* the queuing delay in the bottleneck buffer of the network path, which can be expressed as:

$$qdelay = \frac{k \times S}{B} \tag{3}$$

where k represents the number of packets that linger in the queue, S is the packet length and B is the capacity of the bottleneck link. While the link remains underutilized, there is no *steady* queue built-up in the bottleneck buffer (i.e., $RTT = RTT_{min}$). As soon as the bottleneck channel has been fully utilized, a queue is being built up and RTT is currently:

$$RTT = RTT_{min} + qdelay. \tag{4}$$

If the load increases further, the queue grows as well, until it occupies the whole bottleneck buffer where RTT is eventually maximized and expressed as:

$$RTT_{max} = RTT_{min} + qdelay_{max}. \tag{5}$$

At this point, queuing delay is maximum and expressed as:

$$qdelay_{max} = \frac{bufSize \times S}{B} \tag{6}$$

where *bufSize* denotes the bottleneck buffer size in packets. Buffer overflow inevitably leads to congestion and the experienced packet loss triggers a decrease in the sending rate, enforcing the draining of the buffer.

We observe that congestion is the outcome of buffer overflow (i.e., when queuing delay is maximum), while wireless loss and delay have a weak correlation since wireless errors may occur independently of the bottleneck buffer state. Hence, queuing delay composes a plausible loss discriminator, allowing the sender to follow the appropriate error-recovery strategy. Without explicit feedback (e.g., ECN [27]), queuing delay should be properly measured at the end hosts. Note that throughput cannot be used to estimate the bottleneck buffer occupancy, since it stops increasing after the capacity of the link has been fully utilized. Indeed, with respect to (3) and (4), and considering that the number of packets with length S that can be accommodated in the bottleneck link within each RTT can be expressed as:

$$K = RTT_{min} \times \frac{B}{S}$$

throughput is given by:

$$throughput = \frac{(K+k) \times S}{RTT} = \frac{(K+k) \times S}{RTT_{min} + qdelay}$$
$$= \frac{(RTT_{min} \times \frac{B}{S} + \frac{qdelay \times B}{S}) \times S}{RTT_{min} + qdelay} = B$$

where $(K+k)$ is the total number of packets transmitted per RTT. Since throughput is bounded by link capacity while queue increases, queuing delay can be simply estimated by RTT measurements based on (4).

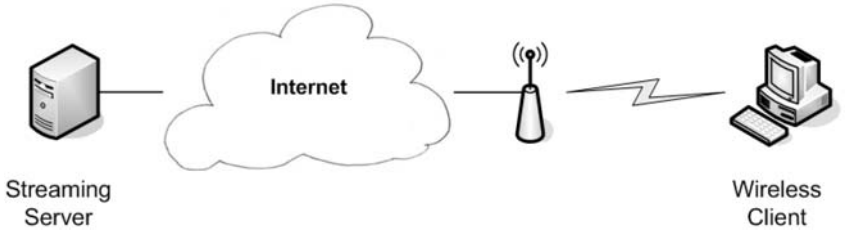


Fig. 2 Typical wireless scenario

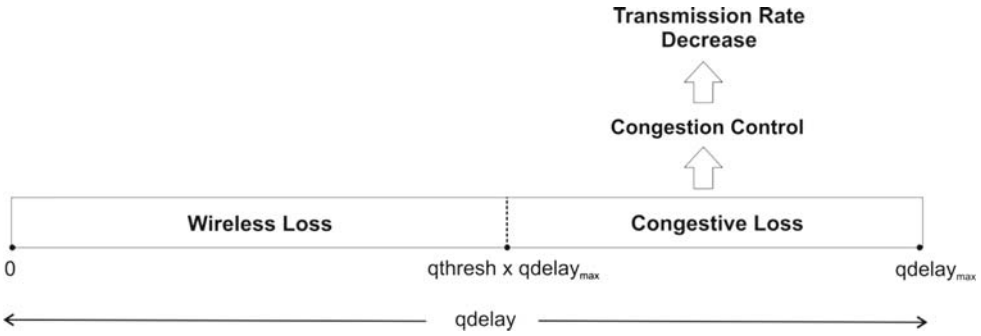


Fig. 3 LDA behavior

3.2 Proposed algorithm

We consider a typical scenario for streaming video delivery across an Internet path that includes a wireless link. As shown in Fig. 2, a streaming server transmits data to a receiver located in the wireless network. In the event of packet loss, the proposed LDA virtually suffices to observe current RTT and subsequently determine the nature of the loss. The LDA interacts with the protocol monitoring RTT_{min} and RTT_{max}, while the queuing delay can be derived by deducting RTT_{min} from the last RTT measured. In the absence of wireless loss, RTT_{max} is normally observed before congestion control is triggered. Practically, upon packet loss if the last RTT is close to RTT_{min}, the bottleneck is not congested and the loss is due to a link error. On the other hand, a measured RTT substantially larger than RTT_{min} and close to RTT_{max} indicates a congestive loss. The protocol's congestion control is therefore complemented with the following loss differentiation algorithm. Upon the detection of packet loss, the transmission rate is decreased only when the following condition is satisfied:

$$\frac{qdelay}{qdelay_{max}} = \frac{RTT - RTT_{min}}{RTT_{max} - RTT_{min}} \geq qthresh. \quad (7)$$

Threshold *qthresh* in (7) specifies the point in queuing delay where packet loss is considered to be congestion-induced. If condition (7) does not hold, the experienced loss is classified as wireless.

Although recovery in response to congestion appears to be straightforward (i.e., multiplicative decrease for AIMD protocols), how should the sender respond to a wireless error detected by the LDA? One approach would allow the sender to ignore the loss and readily increase the transmission rate. In contrast, following a conservative recovery strategy, the source could trigger a gentle rate decrease (i.e., with a higher β), interpreting the wireless error as an indication of a fading wireless channel. We adopt an intermediate approach keeping the sending rate unaffected. Such recovery does not endanger packet loss by increasing the sending rate and at the same time it does not enforce a rate decrease that might be unnecessary and harmful (degrading flow throughput and smoothness). Figure 3 illustrates the behavior of the proposed LDA.

Note that both congestion and wireless loss may take place, rendering differentiation limited. In this case, queuing delay is considerable:

$$qdelay \geq qthresh \times qdelay_{max} \quad (8)$$

which is sensed by:

$$RTT \geq qthresh \times RTT_{max} + (1 - qthresh) \times RTT_{min}. \quad (9)$$

However, in the presence of large queuing delay differentiation is not necessary: packet loss should trigger a rate reduction anyway. Therefore, the LDA classifies the loss as congestive, allowing the flow to recover from congestion, even if wireless loss has been also experienced.

4 SSVP with loss differentiation

4.1 SSVP overview

SSVP [24] is an end-to-end TCP-friendly protocol which is optimized for video streaming applications in the Internet. The protocol does not rely on functionality in routers, such

as *Random Early Detection* (RED) [14], *Explicit Congestion Notification* (ECN) [27] or other *Active Queue Management* (AQM) mechanisms. 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 in order to 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 $\beta = 0.875$. On the occurrence of packet loss, the protocol infers congestion and the sender immediately reduces the transmission rate via the multiplicative increase of IPG:

$$IPG_{i+1} = \frac{IPG_i}{\beta}. \tag{10}$$

If congestion has not been detected, the SSVP source increases the transmission rate by decreasing IPG, as follows:

$$IPG_{i+1} = \frac{1}{1+\alpha} IPG_i \tag{11}$$

The sender adjusts the transmission rate once per RTT in order to maintain a smoothed flow, especially at sudden changes of bandwidth availability. Further details on SSVP can be found in [24].

4.2 Interaction of SSVP with LDA

Consider an SSVP source that transmits n packets with packet lengths $S_1, S_2, \ldots, S_n$ during a time period t, where S_i represents the length of the ith packet. The average throughput achieved by the connection is given by:

$$throughput_i = \frac{1}{t} \sum_{i=1}^{n} S_i = \frac{n \times \bar{S}}{t} \tag{12}$$

where $\bar{S}$ denotes the average packet length. Since SSVP performs rate adjustments once every RTT, we define K as a function of IPG:

$$K_j = \frac{RTT_j}{IPG_j} \tag{13}$$

representing the number of packets transmitted within jth RTT. Assuming a measurement period of one RTT, flow throughput is given by:

$$throughput = \frac{K_j \times \bar{S}}{RTT_j}. \tag{14}$$

Combining (13) and (14), we obtain the instantaneous transmission rate R_i for the SSVP flow:

$$R_i = \frac{\bar{S}}{IPG_i} \tag{15}$$

which is derived from:

$$R_i = \frac{\bar{S}}{t_i + IPG_i} \tag{16}$$

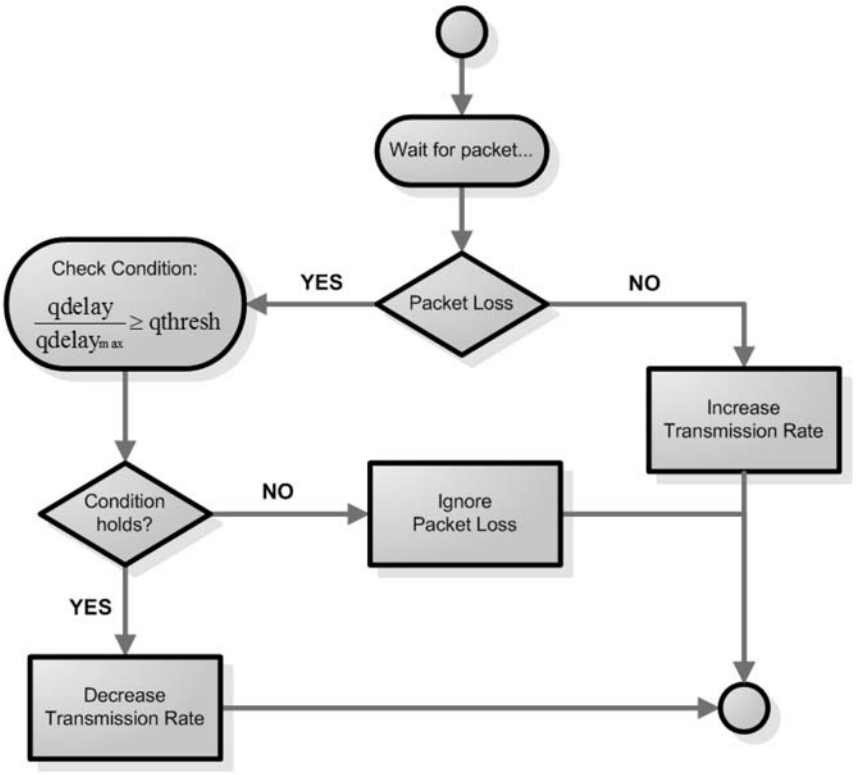
if we consider the transmission time t_i of each packet negligible (compared to IPG).

SSVP does not inherently incorporate any loss differentiation mechanism, and therefore the protocol invokes congestion-oriented responses to all wireless errors. Therefore, upon detecting packet loss, IPG is increased multiplicatively based on (10). Apparently, an increase in the IPG directly affects the transmission rate, as well as flow throughput. Since the protocol does not decouple wireless loss from congestion, wireless errors result in considerable throughput degradation.

The proposed LDA can effectively interact with SSVP, exploiting the RTT estimates obtained by the protocol. The combined approach, denoted as SSVP-LD, decouples congestion from wireless loss based on measured queuing delay. Packet loss is followed by a reduction in the transmission rate with the protocol's standard decrease ratio (i.e., $\beta = 0.875$) only when (7) holds. Alternatively, the sending rate remains temporarily unaffected. Figure 4 provides an overview of SSVP-LD's responses to loss events, depending on current queuing delay. Threshold *qthresh* is adjusted experimentally at 0.5 for SSVP. Hence, when the queue occupies less than half of the bottleneck buffer size, packet drops do not trigger congestion control actions. Certainly, *qthresh* can be adjusted differently in order to modify the protocol's error-recovery strategy.

SSVP-LD is one of the few available end-to-end schemes that exploit both packet loss and delay to decouple wireless loss from congestion. Most mechanisms rely solely on packet loss (e.g., TCP, TFRC [13]) or delay (e.g., Vegas [5], TCP FAST [20]) in order to adjust the transmission rate. While packet loss alone is arguably not sufficient to classify the loss, recent studies (e.g., [22]) uncovered that using delay as a measure of congestion may cause undesirable effects in terms of bandwidth utilization, if it is not augmented with loss information. This implication usually takes place when packet loss occurs due to other reasons than buffer overflow (i.e., wireless errors).

Fig. 4 SSVP-LD flowchart



5 Experimental environment

5.1 Experimental settings

The evaluation was performed on the *NS-2* network simulator. Initially, we conducted simulations on a single-bottleneck *dumbbell* topology (Fig. 5a) with a round-trip link delay of 64 ms. The bottleneck link is shared by competing MPEG and TCP Reno connections and its capacity is 1 or 10 Mbps, depending on each experiment. The capacity of all access links to the sink nodes is set to 1 Mbps. Furthermore, we used a network topology (Fig. 5b) which includes multiple bottlenecks, reverse traffic and varying RTTs. TCP Reno flows are simulated both in the forward and backward direction, while random UDP traffic with the Pareto distribution has been also introduced to the forward direction. The propagation delays of the access links from all the source nodes, as well as the links to the TCP sink nodes range from 5 ms to 15 ms, while the corresponding bandwidth capacities range from 2 Mbps to 10 Mbps. By randomizing RTTs, we avoid synchronization effects.

In both topologies, we used a wireless packet loss model in the access links to the MPEG sink nodes. The loss model was configured on both directions of the link traffic. We specifically used a two-state Markov model, also known as *Gilbert-Elliot* model [15], where the channel switches between a *Good* (G) and a *Bad* (B) state according to given transition probabilities (Fig. 6). A transmission is successful only if the channel is in the *Good* state, otherwise the transmitted packets are lost. The system starts from the *Good* state. The state transitions are shown in the transition probability matrix:

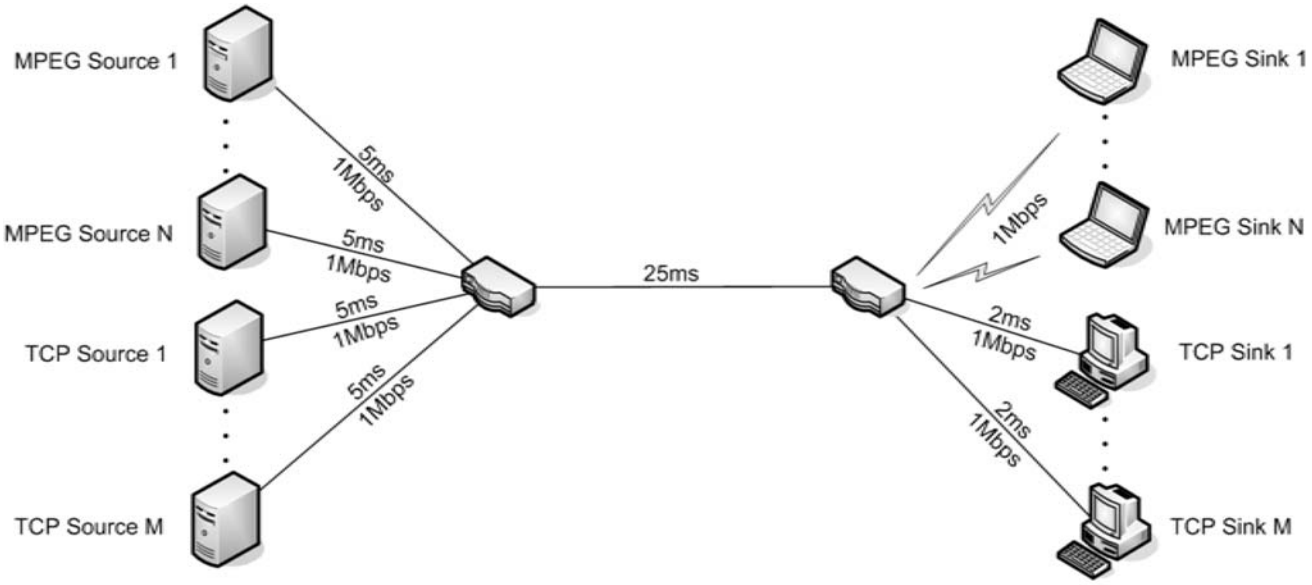
$$P = \begin{bmatrix} p & 1-q \\ 1-p & q \end{bmatrix}$$

where p denotes the probability of successfully transmitting a packet if the previous one was successfully transmitted, and $1-q$ is the probability of successfully transmitting a packet given the previous packet was not received. The steady-state packet error rate is defined as:

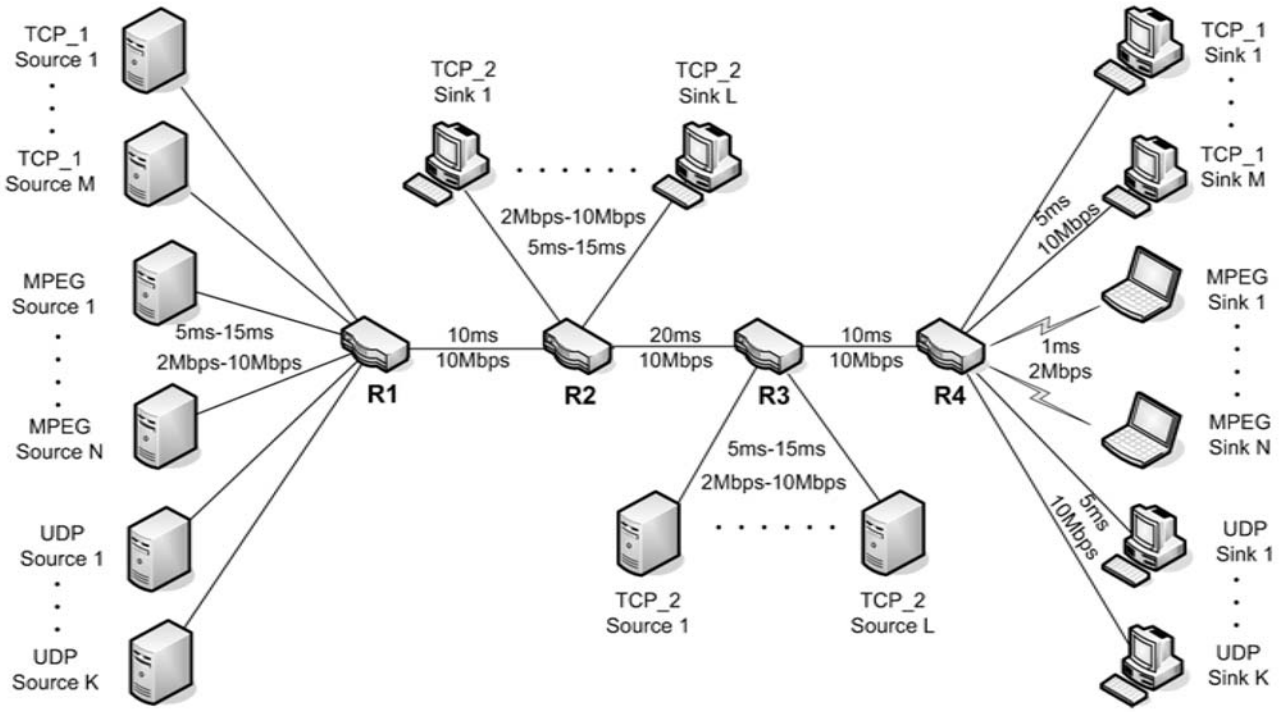
$$\varepsilon = \frac{1-p}{1-p+1-q}.$$

In our simulations, we adjusted $p = 0.994$ and $q = 0.78$ which roughly give $\varepsilon = 0.02$, unless otherwise explicitly stated.

All routers are drop-tail with buffer size adjusted in accordance with the *bandwidth-delay* product. We set the packet size to 1000 bytes for all system flows (with the exception of UDP flows in the topology in Fig. 5b, which have 250 bytes packet size) and the maximum congestion window to 64 KB for all TCP connections. Each simulation lasts for 200 sec allowing all protocols to unfold their potential. Successive runs of each experiment resulted in insignificant confidence intervals. All the results are collected after 2 sec in order to avoid the skew introduced during flows start-up.



(a) Dumbbell Topology



(b) Complex Topology

Fig. 5 Simulation topologies

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. MPEG-4 coding standard is based on I, P and B frames. The compression initiates by encoding a single frame, followed by a group of P and B 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. The model developed is based on *Transform Expand Sample* (TES). We used three separate TES models for I, P and B frames, respectively. The resulting MPEG-4 stream is generated by interleaving data obtained by the three models.

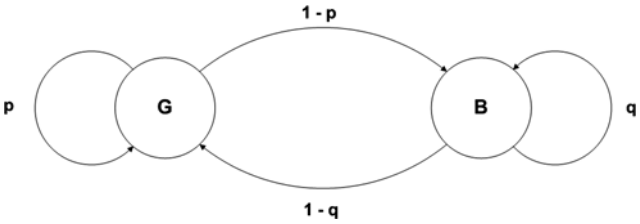


Fig. 6 Two-state Markov model

5.2 Measuring performance

We hereby refer to the performance metrics supported by our simulation model. Since the simulation topology includes competing MPEG and FTP connections, our performance metrics are applied separately to the MPEG and FTP traffic. Throughput is used to measure the efficiency in link utilization:

$$Throughput = \frac{Transmitted\_Data}{Connection\_Time}$$

where *Connection_Time* is the time required for data delivery. Long-term fairness is measured by the *Fairness Index* [10], which is defined as:

$$Fairness\ Index = \left(\sum_{i=1}^{n} throughput_i\right)^2 \Big/ \left(n \sum_{i=1}^{2} throughput_i^2\right)$$

where $Throughput_i$ is the throughput of the ith flow and n is the total number of flows.

In order to quantify the performance on video delivery, we monitor packet inter-arrival times and eventually distinguish the packets that can be effectively used by the client application from delayed packets (according to a configurable inter-arrival threshold). The proportion of delayed packets is reflected in *Delayed Packets Ratio*. Each recipient, receiving packets from the MPEG streaming application, calculates the number of delayed packets based on Algorithm 1.

Several notations used in the pseudocode algorithms are as follows:

1. *threshold*: packet inter-arrival time threshold
2. *delayedPackets*: number of packets with inter-arrival times exceeding the threshold
3. *packetTime*: packet arrival time.

According to streaming video guidelines, playback quality is notably degraded when delay variation exceeds 75 ms. Buffering can eliminate the effects of delay variation by smoothing out jitter; however, additional delays are incurred to the video playback. Furthermore, buffering exhibits certain limitations, such as application delay tolerance and buffer memory constraints. Along these lines, we adjusted the packet inter-arrival threshold at 75 ms, which specifies the point where delay variation becomes perceptible and possibly disturbing. Since MPEG traffic is sensitive to packet drops, we additionally define *Packet Loss Ratio*, as the ratio of the number of lost packets over the number of packets sent by the application.

Algorithm 1 Delayed packets

```
# For each packet received with sequence number i,
determine
# whether it is delayed
if threshold > 0 then
  set packetTime[i] = currentTime
  increase packetsReceived
  if i > 0 and
    packetTime[i] − packetTime[i − 1] > threshold then
    increase delayedPackets
  end if
end if
```

In order to gauge the accuracy of an LDA, we define the *accuracy of loss classification* as the proportion of the number of accurately detected packet losses over the total number of packets losses:

$$Accuracy = \frac{\#accurately\ detected\ packet\ losses}{\#total\ packet\ losses}$$

For a system with multiple flows, we present the average *accuracy of loss classification* of all flows.

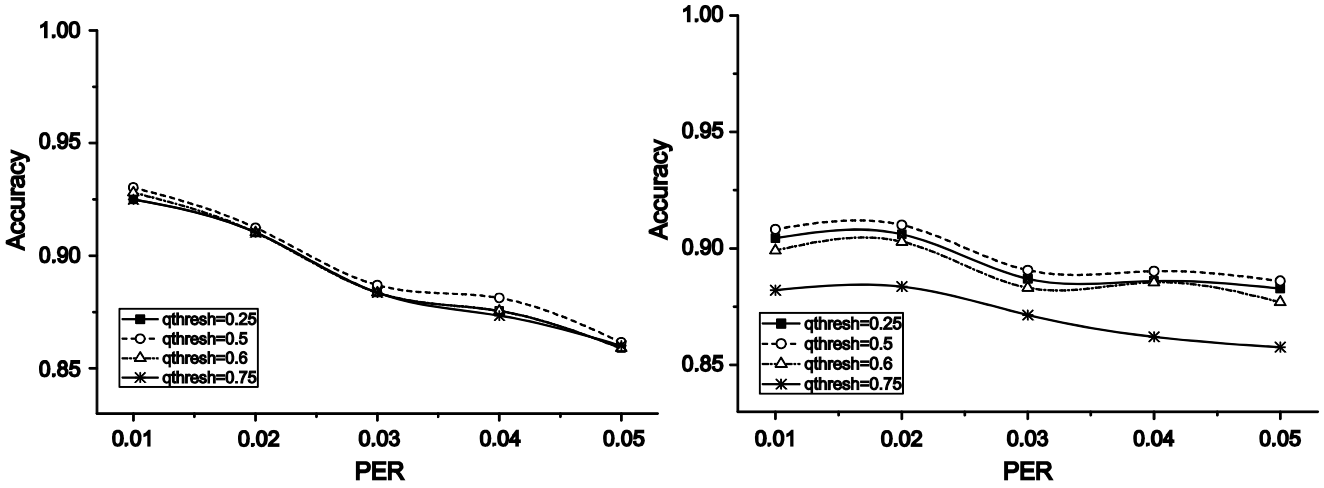
6 Performance evaluation

In this section, we demonstrate conclusive performance studies based on extensive simulation results. More precisely, we evaluate the loss classification accuracy of the proposed LDA and the efficiency of SSVP when it is augmented with the proposed LDA in terms of bandwidth utilization, video delivery, and intra-protocol fairness.

6.1 Accuracy of loss classification

First, we explore the efficiency of the proposed LDA with diverse ***qthresh*** adjustments, which allows the fine tuning of this mechanism. We study *accuracy of loss classification* results for (i) 1 MPEG flow competing with 1 TCP flow on the dumbbell topology with the bottleneck capacity set to 1 Mbps (Fig. 7a), and (ii) 30 MPEG flows on the same topology with bottleneck capacity set to 10 Mbps (Fig. 7b). The corresponding results are presented with diverse packet error rate (PER) adjustments (0.01–0.05).

In both scenarios, our simulation results indicate a slightly increased accuracy for *qthresh* adjusted to 0.5. In



(a) 1 MPEG flow **(b)** 30 MPEG flows

Fig. 7 Accuracy of loss classification with diverse *qthresh* adjustments vs. PER

Fig. 7a, the single MPEG flow experiences contention from the corporate TCP flow, which eventually leads to congestive drops. However, as PER increases, wireless losses become dominant. In this case, the presence of congestion and wireless loss makes differentiation difficult and inevitably the accuracy of the LDA slightly decreases.

The proper adjustment of *qthresh* can provide perceptible benefits. A *qthresh* that is set close to 0 tends to misclassify wireless losses as congestion. In contrast, a *qthresh* adjustment near 1 prevents the LDA from detecting congestion, interpreting the experienced loss mostly as wireless. This behavior does not allow rapid recovery from congestion. Our results show that setting *qthresh* to 0.5 achieves the desired loss differentiation, while misclassifications can only occur for increased queuing delays where the LDA infers congestion. Despite the possible misclassification, the protocol is instructed (by the LDA) to decrease its rate (typically multiplicatively), which is the correct action due to the large queue in the bottleneck buffer.

With the multiple MPEG flows (Fig. 7b), congestion is experienced more frequently, due to the strong contention between the flows. Adjusting *qthresh* to 0.75 evidently decreases the level of accuracy, since, as already explained, an amount of congestive loss is misclassified as wireless. Setting *qthresh* to 0.5 provides the highest accuracy among the *qthresh* adjustments. Hence, *qthresh* is always set to 0.5 for the simulation results in the remainder of Sect. 6.

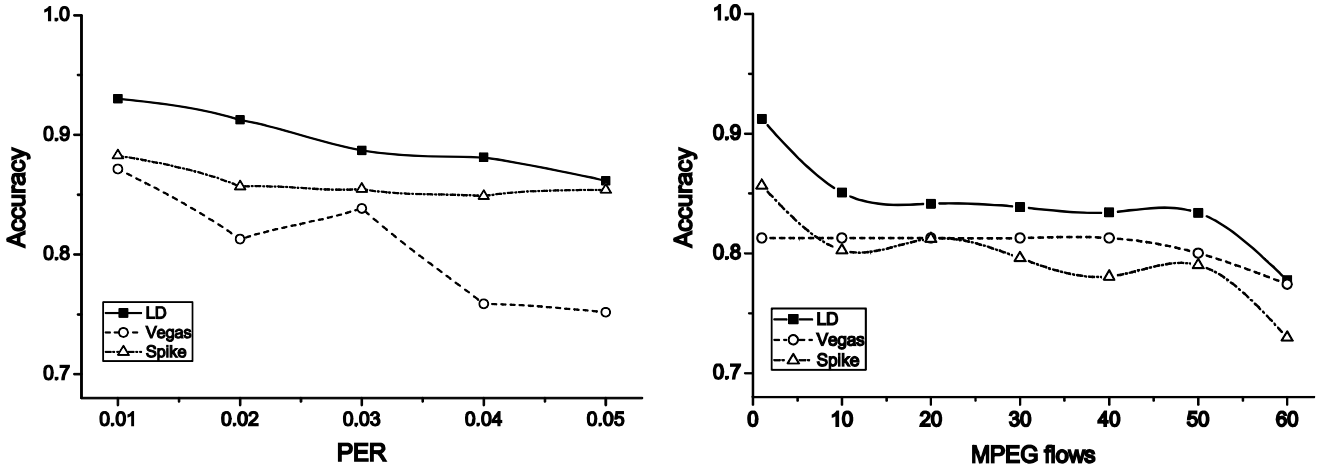
Furthermore, we conducted simulations to evaluate the loss classification accuracy of our proposed LDA (noted as LD in Fig. 8) versus two well-known LDAs: the Vegas predictor and Spike. More precisely, we simulated (i) 1 MPEG flow competing with 1 TCP flow on the dumbbell topology with the bottleneck capacity set to 1 Mbps and varying PER adjustments, and (ii) a diverse number of MPEG flows in the same topology with 10 Mbps bottleneck and PER set to 0.02.

Although misclassifications occur in all LDAs, our mechanism achieves high accuracy, even with high PER or increased contention. The Vegas predictor tends to misclassify wireless losses and its accuracy decreases for high PER (Fig. 8a). On the contrary, its accuracy is not significantly affected by the level of contention (Fig. 8b). Although Spike does not achieve a remarkable degree of accuracy, it exhibits less sensitivity to increased PER or contention. Our simulation results show that the proposed LDA compares favorably with the two representative mechanisms for loss classification. The gains from using our LDA in terms of throughput and video performance are discussed in the following subsections.

6.2 LDA efficiency with SSVP

Departing from the loss classification accuracy, we assess the efficiency of the proposed LDA on the SSVP protocol. The corresponding experiments were conducted on the dumbbell topology with the bottleneck capacity set to 10 Mbps. We simulated a wide range of MPEG flows (10–60) of (i) SSVP, and (ii) SSVP with LDA (SSVP-LD), successively. We measured *System Throughput* and *Fairness Index,* and we additionally demonstrate statistics from delayed packets which compose an influencing factor for perceptual video quality (Fig. 9).

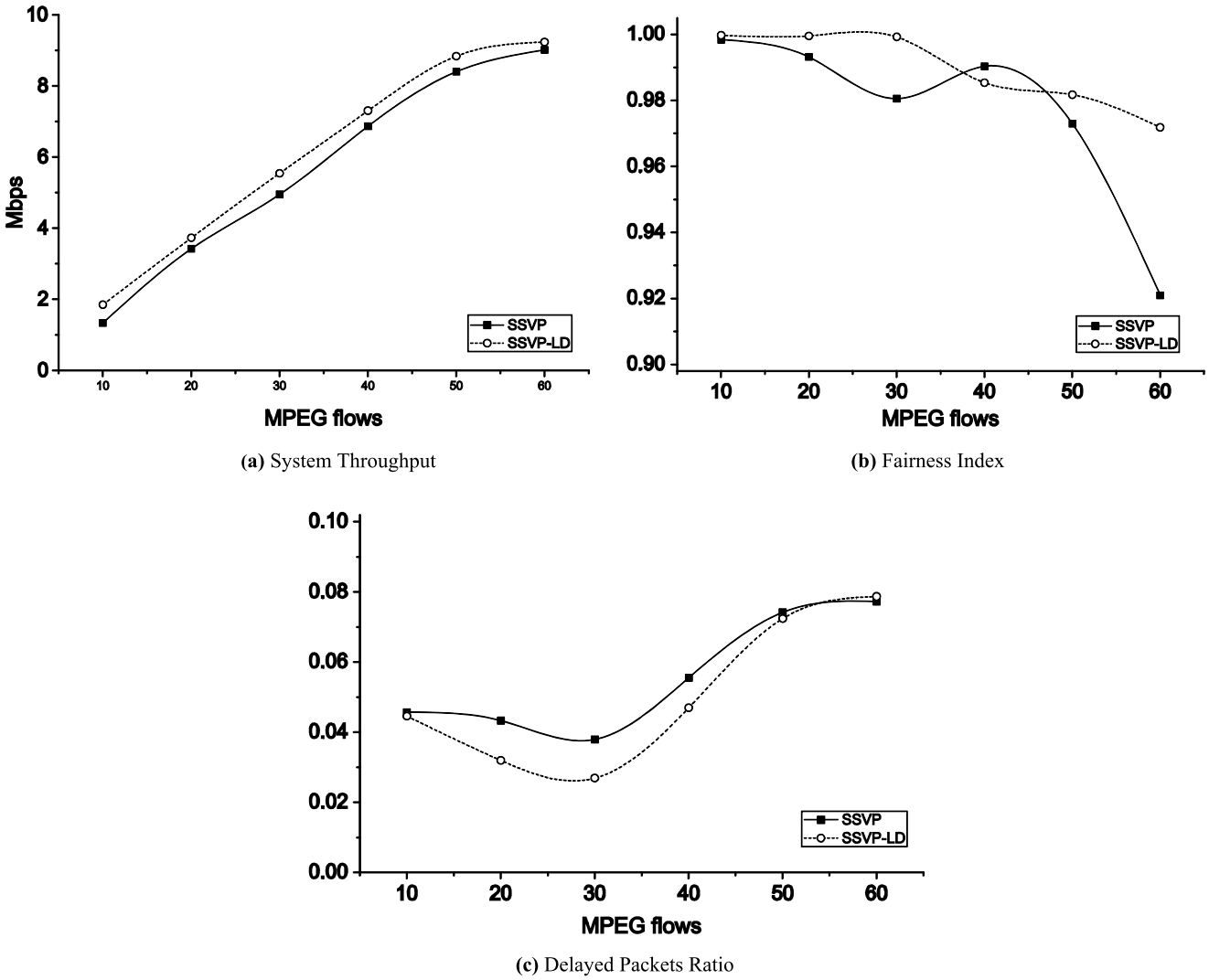
Throughput performance (Fig. 9a) reflects the beneficial role of the integrated LDA. SSVP-LD is relatively immunized by the link errors across the wireless channel and utilizes a higher fraction of the available bandwidth, regardless of link multiplexing. On the other hand, standard SSVP invokes congestion-oriented responses to the wireless errors,



(a) 1 MPEG flow and diverse PER

(b) Diverse MPEG flows (PER: 0.02)

Fig. 8 Accuracy loss of classification for the proposed LD vs. Vegas and Spike

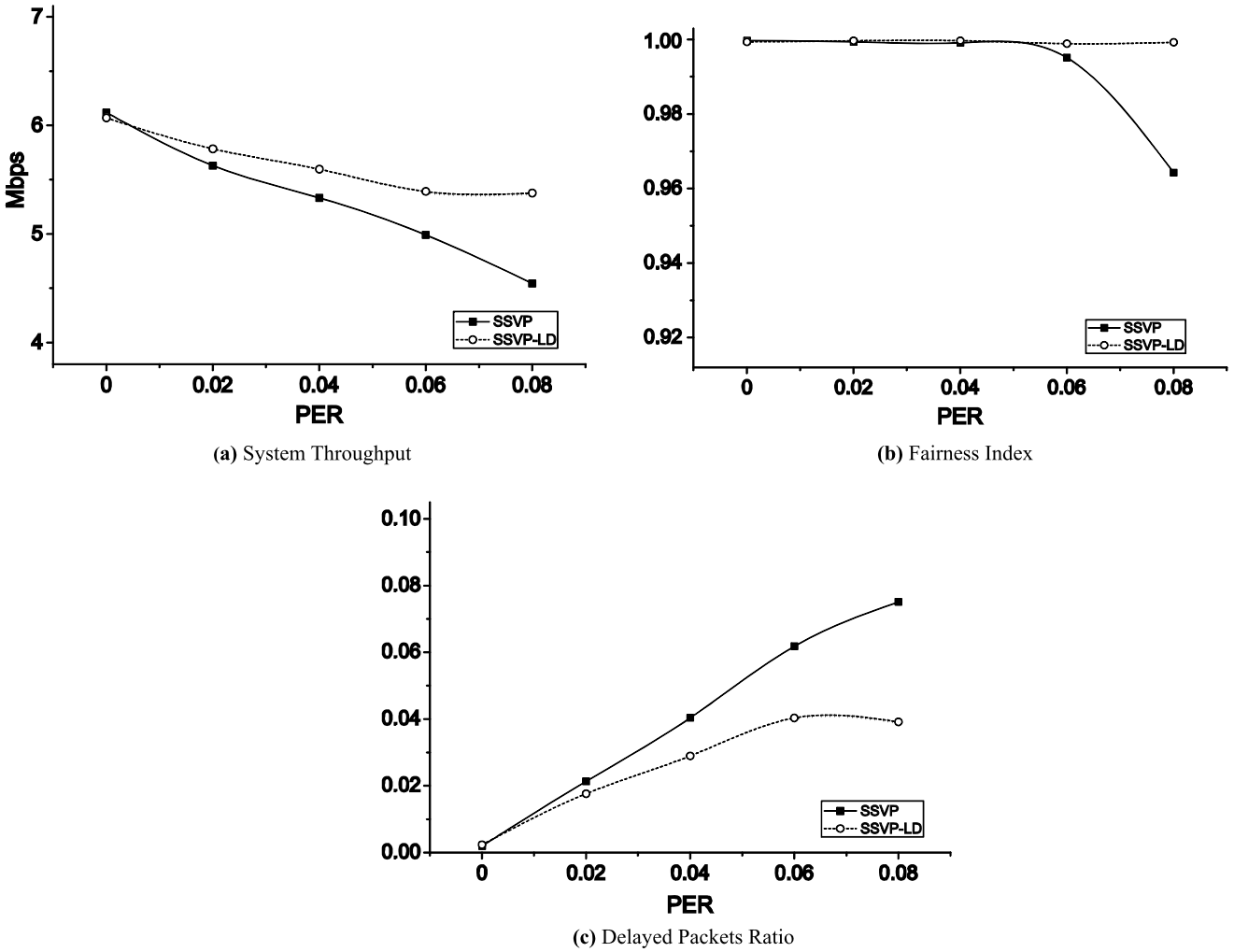


(a) System Throughput

(b) Fairness Index

(c) Delayed Packets Ratio

Fig. 9 Performance with wireless errors



(a) System Throughput

(b) Fairness Index

(c) Delayed Packets Ratio

Fig. 10 Performance with diverse error rates

diminishing the throughput rate. We note that the protocol incorporates a gentle decrease ratio (i.e., $\beta = 0.875$), and therefore the impact of an unnecessary multiplicative decrease is not destructive on throughput performance. Nevertheless, a protocol with conventional congestion control parameters (i.e., $\alpha = 1$, $\beta = 0.5$), such as TCP, would experience significant throughput degradation. Therefore, incorporating the proposed LDA into other protocols may result in more gains, depending on the selection of the AIMD parameters.

As shown in Fig. 9b, SSVP-LD excels in bandwidths sharing, while SSVP also achieves satisfactory levels of fairness. Although SSVP's congestion control maintains adequate AIMD oscillation allowing all the system flows to converge to the fairness point, congestion along with wireless loss can undermine long-term fairness (i.e. 60 SSVP flows). In this case, the presence of the LDA notably improves fairness, as well as system stability.

In order to quantify the smoothness observed by the end-user, we trace the proportion of delayed packets, effectively capturing the effect of jitter. In [24] we showed that SSVP maintains a smooth sending rate in accordance with the QoS provisions of video streaming applications. SSVP's smoothness is also demonstrated in Fig. 9c, where the protocol achieves the timely delivery of most packets. However, the incorporated LDA further refines transmission rate fluctuations, abolishing the damage of error-induced multiplicative decrease on flow throughput and smoothness. As a result, SSVP-LD delivers a smoother video flow, especially when link errors are the primary cause for the observed packet drops (i.e., 10–40 flows).

We also carried out experiments using diverse PER adjustments (0.02–0.08) in order to investigate the efficiency of the LDA over highly erroneous wireless links. We hereby demonstrate results from 30 MPEG flows on the dumbbell topology (Fig. 10). Although packet error rates as high as 0.08 are not common across wireless channels, such simu-

lations provide useful insights into protocol sensitivity over lossy links.

Figure 10a illustrates that throughput performance notably degrades, especially in the case of SSVP. SSVP-LD is less sensitive to the increased error rates, since the integrated LDA alleviates most of the undesirable effects induced by the link errors. We note that throughput degradation is sometimes inevitable, since wireless errors may occur while the queue length is close to the buffer size.

Besides the throughput gains achieved by the LDA, Fig. 10b reveals that fairness for SSVP-LD is not degraded, even for error rates as high as 0.08. Errors are not concurrently experienced by all flows; hence, some flows may back off while the others may keep growing. This *partial* downward adjustment upon packet loss results in varying flow throughput rates and has a direct impact on fairness. This observation is primarily applicable to SSVP, as well as to other protocols that cannot detect the nature of the error. SSVP-LD nearly avoids this implication, since multiplicative decrease is invoked only when the queue occupies a considerable proportion of the buffer size.

In terms of video delivery, delay variation becomes more evident, as the error rate increases (Fig. 10c). Packet errors occasionally induce interruptions in the sending rate and the perceptual video quality inevitably deteriorates. The proposed LDA manages to alleviate most of these impairments and sustain a relatively smooth video flow. On the contrary, SSVP's downward adjustments in response to the wireless errors cause perceptible oscillations in the sending rate, with the effect of jitter becoming evident to the end-user. We observe that the *Delayed Packets Ratio* increases almost in proportion to PER for SSVP. In contrast, SSVP-LD 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.

6.3 SSVP-LD performance

The remainder of Sect. 6 includes evaluations of SSVP-LD with TFRC [31] and GAIMD (0.31, 0.875). TFRC, in particular, is designed for efficiency on media delivery over a wide range of network and session dynamics. We simulated a diverse range of MPEG flows (10–60) of (i) SSVP-LD, (ii) TFRC, and (iii) GAIMD, competing with 5 FTP connections of TCP Reno, successively. PER was adjusted at 0.02. Similarly to the previous scenarios, we obtained the corresponding *Throughput*, *Fairness Index* and *Delayed Packets Ratio* measurements, along with *Packet Loss Ratio* (Fig. 11). These experiments were conducted on the dumbbell topology with 10 Mbps bottleneck capacity.

According to Fig. 11a, both SSVP-LD and TFRC utilize a high fraction of the available bandwidth. SSVP-LD remains relatively immunized by the wireless errors, as well as by the interfering TCP flows. The protocol apparently operates more efficiently during high link-multiplexing, since it performs downward adjustments primarily in response to congestion. On the contrary, GAIMD fails to adapt to the network dynamics, since it cannot detect the nature of the error. The presence of both congestion and error-induced loss eventually diminishes the protocol's throughput rate.

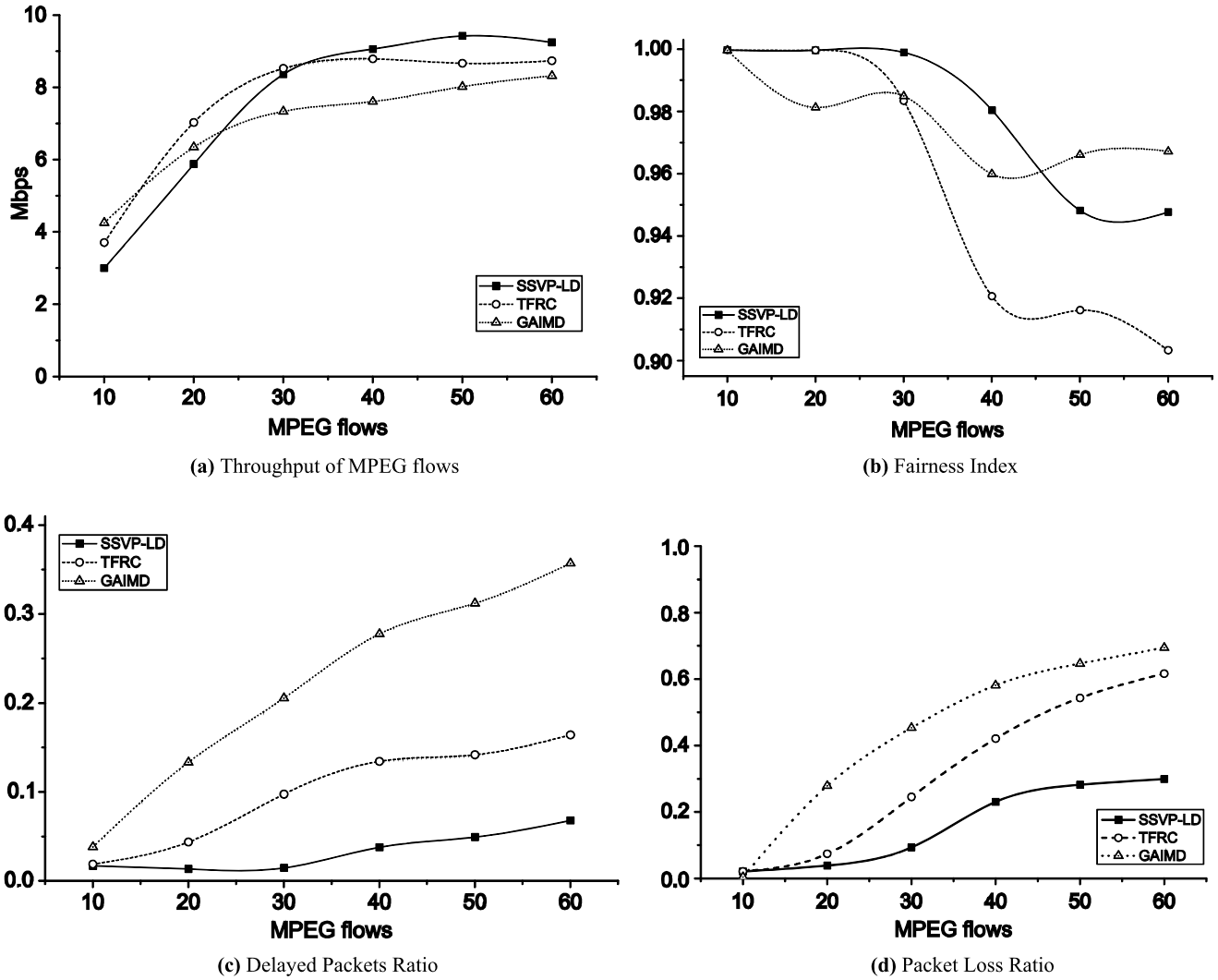
Figure 11b illustrates that SSVP-LD and GAIMD achieve high levels of fairness. The AIMD-based responses during congestion enforce competing flows to converge to the fairness point for both protocols. On the other hand, we observe that the *Fairness Index* for TFRC degrades abruptly, reflecting a throughput imbalance between the connections. TFRC's equation-based responses to packet loss undermine long-term fairness, along with contention increase.

According to Fig. 11c, SSVP-LD achieves the timely delivery of video packets maintaining an uninterrupted and smooth sending rate that is slightly affected by wireless errors and contention. TFRC's error-induced downward adjustments cause oscillations in the sending rate, and subsequently delay variation of perceptible magnitude. Furthermore, Fig. 11d illustrates increased packet losses for TFRC, which inevitably deteriorate the perceptual video quality. In dynamic environments with wireless errors, TFRC occasionally fails to obtain accurate estimates of the loss event rate, since TFRC's throughput model, as expressed in (1), is sensitive to packet loss. GAIMD's performance on video delivery may as well frustrate the end-user. The protocol's congestion-oriented responses to all types of errors counterbalance the potential gains from a gentle decrease ratio that could favor smoothness in a static and error-free network.

6.4 SSVP-LD performance with heterogeneous networks and reverse traffic

We conclude our performance studies with simulations on the reverse-traffic topology, which allowed us to draw further conclusions on SSVP-LD efficiency at conditions of increased contention with various types of traffic, including TCP reverse flows and UDP burst traffic. We specifically simulated 10–60 MPEG flows competing with 15 TCP (5 in the forward and 10 in the reverse direction) and 10 UDP flows. The peak sending rates for UDP flows range from 64 to 256 kbps, each one occupying up to 2.5% of the bottleneck bandwidth. The MPEG flows were simulated with SSVP-LD, TFRC, and GAIMD.

Figure 12a shows that SSVP-LD achieves effective bandwidth utilization, despite the relatively limited available resources due to the presence of interfering TCP and UDP flows. Furthermore, SSVP-LD is not perceptibly affected by reverse traffic, achieving high throughput rates even for high link-multiplexing. TFRC's error-induced downward adjust-



(a) Throughput of MPEG flows

(b) Fairness Index

(c) Delayed Packets Ratio

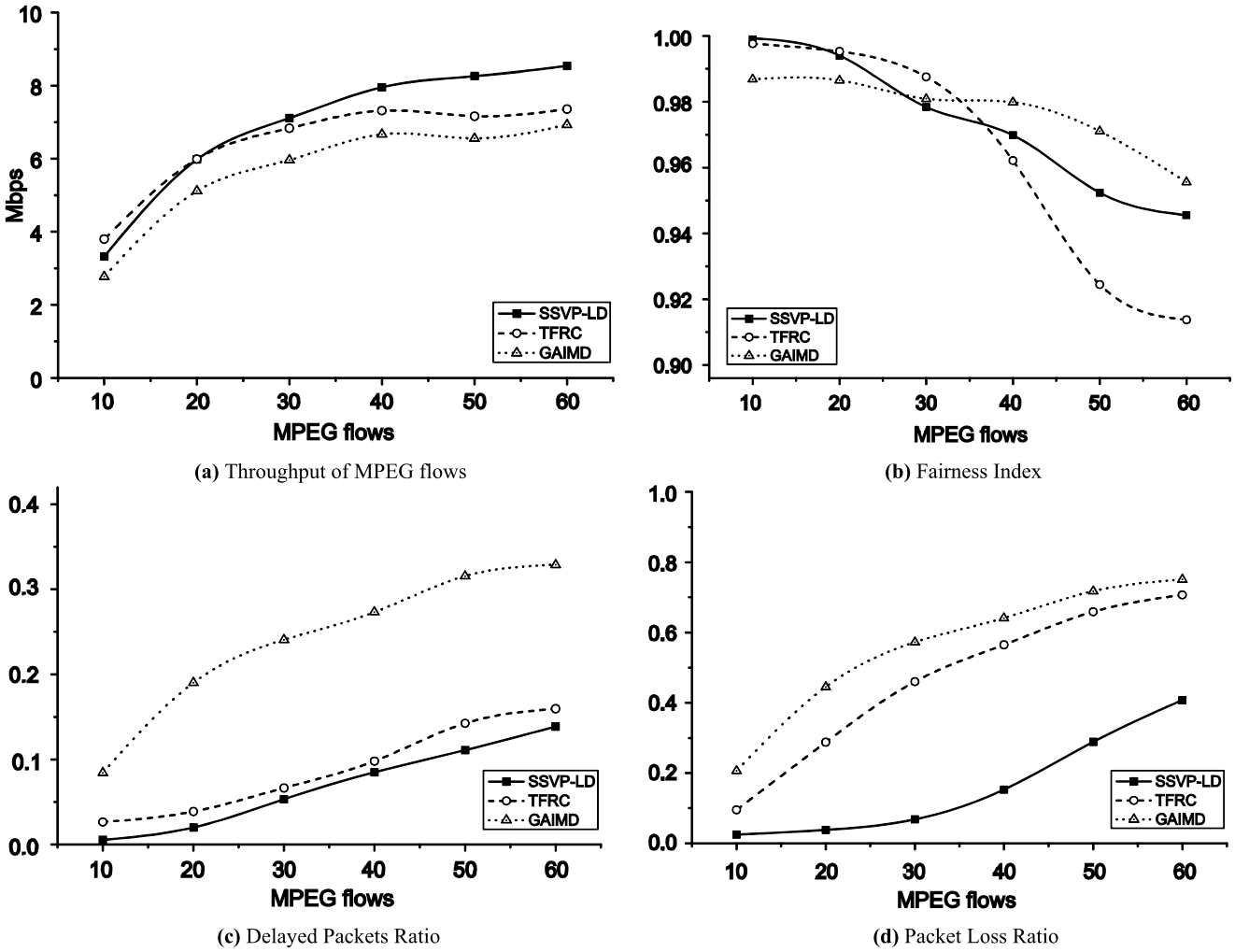
(d) Packet Loss Ratio

Fig. 11 Performance with wireless errors and competing traffic

ments do not allow the protocol to utilize wireless resources efficiently, sacrificing an amount of available bandwidth. GAIMD exhibits a notable inefficiency, since reverse traffic throttles the rate of incoming ACKs and subsequently the growth of the congestion window. An infrequent ACK rate may also result in timeout events for TCP protocols, since *Fast Retransmit* and *Fast Recovery* [30] are not triggered (i.e., the TCP sender does not receive 3 duplicate ACK) when the reverse path becomes congested.

In terms of intra-protocol fairness, the AIMD-based SSVP-LD and GAIMD achieve a fair behavior among their flows (Fig. 12b). Inline with our observations in Sect. 6.3, TFRC's *Fairness Index* exhibits a noticeable degradation, as the number of MPEG flows increases. Hence, multiple TFRC flows may have varying throughput rates resulting in different levels of service quality. This difference can be further amplified in the presence of many background flows or higher loss rates.

Figures 12c, 12d illustrate the remarkable efficiency of SSVL-LD from the perspective of video delivery. Inline with the corresponding results of the previous section (Fig. 11), the protocol delivers video of acceptable quality regardless of link-multiplexing. The interaction of the LDA with SSVP sustains low packet loss, delivering a large amount of the video data sent by the application without any retransmission effort (Fig. 12d). Apart from packet loss statistics, the proportion of delayed packets, as shown in Fig. 12c, reflects SSVP-LD's smooth video delivery for a wide range of network dynamics. An overview of all results in Fig. 12 reveals that competing flows in the forward and backward direction do not cause notable implications on SSVP-LD efficiency and the perceptual video quality achieved by the protocol. The rest of the protocols exhibit a notable degree of performance degradation (Figs. 12c, 12d), which becomes more evident as contention increases. Packet loss composes a limiting factor both for TFRC and GAIMD, while the latter



(a) Throughput of MPEG flows

(b) Fairness Index

(c) Delayed Packets Ratio

(d) Packet Loss Ratio

Fig. 12 Performance with heterogeneous networks and reverse traffic

additionally introduces considerable delay variation with an adverse effect on stream playback quality.

7 Conclusions

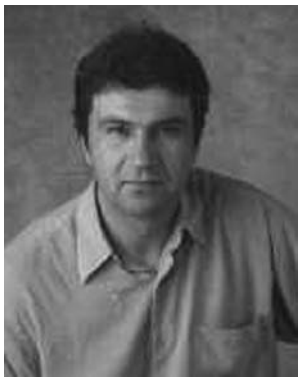
We have presented an efficient and viable end-to-end mechanism for loss differentiation that renders the underlying transport protocol less susceptible to wireless loss. Consequently, a wasteful rate reduction in response to a wireless error is usually prevented with notable gains in terms of flow throughput. Furthermore, the proposed LDA reduces the magnitude of AIMD oscillation inline with the requirements of media-streaming applications for smooth patterns of data transmission. We incorporated the LDA into AIMD-based SSVP, concentrating on the interactions between the two mechanisms. Our simulation results validated the feasibility and efficiency of this combined approach, in the context of bandwidth utilization, video delivery, and intra-protocol fairness. According to our knowledge, SSVP-LD composes one of the few available end-to-end schemes that achieve efficient performance on video delivery in wired/wireless networks, without requiring the support from lower-layer feedback or AQM mechanisms.

References

1. Balakrishnan, H., Padmanabhan, V., Seshan, S., & Katz, R. (1997). A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, *5*(6), 756–769.
2. Bansal, D., & Balakrishnan, H. (2001). Binomial congestion control algorithms. In *Proc. IEEE INFOCOM 2001*, Anchorage, Alaska, USA, April 2001.
3. Biaz, S., & Vaidya, N. H. (2005). De-randomizing congestion losses to improve TCP performance over wired-wireless networks. *IEEE/ACM Transactions on Networking*, *13*(3), 596–608.
4. Boyce, J., & Gaglianello, R. (1998). Packet loss effects on MPEG video sent over the public Internet. In *Proc. ACM Multimedia*, Bristol, UK, September 1998.

5. Brakmo, L., & Peterson, L. (1995). TCP Vegas: end to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications*, *13*(8), 1465–1480.
6. Bregni, S., Caratti, D., & Martignon, F. (2003). Enhanced loss differentiation algorithms for use in TCP sources over heterogeneous wireless networks. In *Proc. IEEE GLOBECOM*, San. Francisco, CA, USA, December 2003.
7. Capone, A., Fratta, L., & Martignon, F. (2004). Bandwidth estimation schemes for TCP over wireless networks. *IEEE Transactions on Mobile Computing*, *3*(2), 129–143.
8. Cen, S., Cosman, P. C., & Voelker, G. M. (2003). End-to-end differentiation of congestion and wireless losses. *IEEE/ACM Transactions on Networking*, *11*(5), 703–717.
9. Chen, M., & Zakhor, A. (2004). Rate control for streaming over wireless. In *Proc. IEEE INFOCOM 2004*, Hong Kong, China, March 2004.
10. Chiu, D., & Jain, R. (1989). Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *Journal of Computer Networks and ISDN*, *17*(1), 1–14.
11. Chockalingam, A., Zorzi, M., & Tralli, V. (1999). Wireless TCP performance with link layer FEC/ARQ. In *Proc. IEEE ICC*, Vancouver, Canada, June 1999.
12. Floyd, S., & Fall, K. (1999). Promoting the ese of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, *7*(4), 458–472.
13. Floyd, S., Handley, M., Padhye, J., & Widmer, J. (2000). Equation-based congestion control for unicast applications. In *Proc. ACM SIGCOMM* 2000, Stockholm, Sweden, August 2000.
14. Floyd, S., & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, *1*(4), 397–413.
15. Gilbert, E. (1960). Capacity of a burst-noise channel. *Bell System Technical Journal*, *39*(5), 1253–1266.
16. Goff, T., Moronski, J., Phatak, D., & Gupta, V. (2000). Freeze-TCP: A true end-to-end enhancement mechanism for mobile environments. In *Proc. IEEE INFOCOM 2000*, Tel-Aviv, Israel, March 2000.
17. Grieco, L., & Mascolo, S. (2004). Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control. *ACM Computer Communication Review*, *34*(2), 25–38.
18. Hu, F., & Sharma, N. K. (2002). Enhancing wireless Internet performance. *IEEE Communications Surveys and Tutorials*, *4*(1), 2–15.
19. Jamieson, K., & Balakrishnan, H. (2007). PPR: Partial packet recovery for wireless networks. In *Proc. ACM SIGCOMM 2007*, Kyoto, Japan, August 2007.
20. Jin, C., Wei, D. X., & Low, S. H. (2004). TCP FAST: motivation, architecture, algorithms, performance. In *Proc. IEEE INFOCOM 2004*, Hong Kong, China, March 2004.
21. Kohler, E., Handley, M., & Floyd, S. (2006). Designing DCCP: Congestion control without reliability. In *Proc. ACM SIGCOMM 2006*, Piza, Italy, September 2006.
22. Martin, J., Nilsson, A., & Rhee, I. (2003). Delay-based congestion avoidance for TCP. *IEEE/ACM Transactions on Networking*, *11*(3), 356–369.
23. Mascolo, S., Casetti, C., Gerla, M., Sanadidi, M., & Wang, R. (2001). TCP Westwood: Bandwidth estimation for enhanced transport over wireless links. In *Proc. ACM MOBICOM 2001*, Rome, Italy, July 2001.
24. Papadimitriou, P., & Tsaoussidis, V. (2007). SSVP: A congestion control scheme for real-time video streaming. *Computer Networks*, *51*(15), 4377–4395.
25. Papadimitriou, P., Tsaoussidis, V., & Tsekeridou, S. (2005). The impact of network and protocol heterogeneity on application QoS. In *Proc. 10th IEEE int/nal symposium on computers and communications (ISCC)*, Cartagena, Spain, June 2005.
26. Papadimitriou, P., Tsaoussidis, V., & Zhang, C. (2007). Enhancing video streaming delivery over wired/wireless networks. In *Proc. 13th European wireless conference (EW 2007)*, Paris, France, April 2007.
27. Ramakrishnan, K., & Floyd, S. (1999). *A proposal to add explicit congestion notification (ECN) to IP*, RFC 2481, January 1999.
28. Rejaie, R., Handley, M., & Estrin, D. (1999). 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.
29. Sinha, P., Venkitaraman, N., Sivakumar, R., & Bharghavan, V. (1999). WTCP: a reliable transport protocol for wireless wide-area networks. In *Proc. ACM MOBICOM '99*, Seattle, USA, August 1999.
30. Stevens, W. (1997). *TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms*, RFC 2001, January 1997.
31. TCP-friendly rate control NS-2 implementation. Available online at: http://www.isi.edu/nsnam/dist/.
32. Tsaoussidis, V., & Badr, H. (2000). TCP-probing: towards an error control schema with energy and throughput performance gains. In *Proc. 8th IEEE int/nal conference on network protocols (ICNP)*, Osaka, Japan, November 2000.
33. Tsaoussidis, V., & Matta, I. (2002). Open issues on TCP for mobile computing. *Journal of Wireless Communications and Mobile Computing*, *2*(1), 3–20.
34. Tsaoussidis, V., & Zhang, C. (2002). TCP real: Receiver-oriented congestion control. *Computer Networks*, *40*(4), 477–497.
35. Tsaoussidis, V., & Zhang, C. (2005). The dynamics of responsiveness and smoothness in heterogeneous networks. *IEEE Journal on Selected Areas in Communications*, *23*(6), 1178–1189.
36. Yang, Y. R., Kim, M. S., & Lam, S. S. (2001). Transient behaviors of TCP-friendly congestion control protocols. In *Proc. IEEE INFOCOM*, Anchorage, Alaska, USA, April 2001.
37. Yang, Y. R., & Lam, S. S. (2000). General AIMD congestion control. In *Proc. 8th IEEE ICNP*, Osaka, Japan, November 2000.

Panagiotis Papadimitriou is currently a Research Associate in the Computing Department of Lancaster University, UK. He received a Ph.D. in Electrical and Computer Engineering from Democritus University of Thrace, Greece, in 2008. He also obtained a B.Sc. in Computer Science from University of Crete, Greece, in 2000, and a M.Sc. in Information Technology from University of Nottingham, UK, in 2001. From 2003 to 2008, Panagiotis was a visiting Lecturer in the Information Management Department of Technological Institute of Kavala, Greece. During the summer of 2006, he was a visiting Researcher in the Florida International University, USA. Panagiotis has numerous publications in high-quality international conferences and scientific journals. His research interests include network and router virtualization, congestion control, Internet QoS, and video and voice over IP. Panagiotis serves on the technical program committee of numerous networking conferences.



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). Vassilis held faculty positions in Rutgers University, New Brunswick, SUNY Stony Brook and Northeastern University, Boston. In May 2003, Vassilis 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. Vassilis 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.

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.