



Available online at [www.sciencedirect.com](http://www.sciencedirect.com)

SCIENCE @ DIRECT®

J. Vis. Commun. Image R. 16 (2005) 450–474

JOURNAL OF  
**VISUAL**  
Communication &  
**IMAGE**  
Representation

[www.elsevier.com/locate/jvci](http://www.elsevier.com/locate/jvci)

# Split-domain video transmission protocol for video streaming over hybrid wired–wireless connections

Rick Ha, Weihua Zhuang \*

*Department of Electrical and Computer Engineering, University of Waterloo,  
200 University Avenue West, Waterloo, Ont., Canada N2L 3G1*

Received 29 February 2004; accepted 25 November 2004  
Available online 9 April 2005

---

## Abstract

The imminent inception of third-generation mobile communication technologies and continual proliferation of wireless local area networks offer an unprecedented opportunity for the development of video streaming applications through wireless Internet access. Different design challenges exist in implementing video streaming connections spanning both wired and wireless domains. This paper proposes a split-domain video transmission protocol to allow streaming video transmission based on adaptive rate encoding over hybrid wired–wireless connections. Computer simulation results are presented to demonstrate the benefits and viability of such a video streaming method over existing transport options.

© 2005 Elsevier Inc. All rights reserved.

*Keywords:* Mobile communication systems; Multimedia applications; Transport protocols; TCP; Video streaming

---

## 1. Introduction

The advent and proliferation of the Internet in the last decade have tremendously facilitated the spread of information amongst individuals throughout the

---

\* Corresponding author. Fax: +1 519 746 3077.

*E-mail addresses:* [rwkha@bbr.uwaterloo.ca](mailto:rwkha@bbr.uwaterloo.ca) (R. Ha), [wzhuang@bbr.uwaterloo.ca](mailto:wzhuang@bbr.uwaterloo.ca) (W. Zhuang).

world. At the same time, the explosive growth of wireless communication has rendered itself as an indispensable part of the modern society. Besides supporting ordinary voice communications, cellular phones, and other wireless devices serve as connection portals to the Internet for providing a seamless Internet connection to mobile users. However, because of the intrinsic differences that exist between wired and wireless domains, the freedom of mobility does not translate into full access to Internet resources and capabilities, such as video streaming. The principal challenge in providing streaming video applications to mobile users lies on devising a single protocol that is able to harmonize the different design parameters of wired and wireless domains, guarantee the quality-of-service (QoS) requirements of the video stream, and minimize conflicts with other existing Internet connections. Both wired and wireless domains introduce various degrees of network volatility and design complexity that pose as a formidable obstacle in guaranteeing high video streaming QoS.

Currently, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are the main vehicles for video transmission over the Internet. TCP [1] was first designed with the intent of operating over terrestrial wired links that guarantees data transmission integrity while maintaining overall network stability. It is not, however, optimized for video streaming applications. First of all, the perpetual changes in the TCP congestion window, caused by the slow start algorithm, introduce undesirable fluctuations in video transmission rate, as demonstrated in Fig. 1. Also, the TCP retransmission mechanism does not exploit the loss-tolerant nature of video traffic such that unnecessary retransmissions can be minimized. Finally, in cross-domain connections, TCP cannot distinguish the source of packet loss whether due to congestion in the wired link or due to random bit errors in the wireless link, thus degrading the overall throughput. A number of well-known studies have suggested TCP enhancements (such as Indirect TCP (I-TCP) [2] and TCP Snoop [3]) to accommodate wireless domain characteristics, but they do not specifically target video streaming applications.

Unlike TCP, UDP [4] is a lightweight transmission protocol that is ideal for real-time video streaming since it is not loaded with any congestion control or

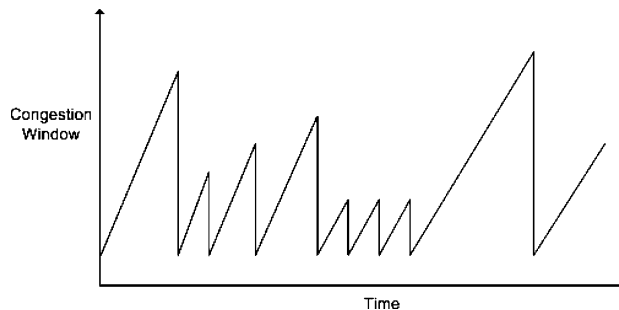


Fig. 1. Saw-tooth profile of TCP congestion window.

retransmission mechanisms. The playback rate is guaranteed to be constant and smooth at all times without the constraint of a congestion window structure. Also, UDP-based protocols have a more pronounced advantage over TCP-based protocols in competing for network bandwidth because any influx of UDP traffic into the network will invoke the congestion control mechanism of existing TCP connections, thereby surrendering the extra bandwidth to the greedy UDP streams for the sake of maintaining network stability. Despite these seeming advantages over TCP, there exist a number of drawbacks in the use of UDP in video streaming applications. First of all, without the restrictions of congestion control, multiple high-rate UDP sessions would starve other TCP connections of network bandwidth, which may eventually lead to congestion collapse [5]. Second, in transmitting over noisy wireless links, UDP connections tend to suffer high packet loss because of the lack of retransmission mechanism, which would substantially degrade video quality at the mobile host. Although the situation can be alleviated with the protection of error resiliency tools incorporated by current video encoding standards such as MPEG-4 [6,7], additional processing power and battery power must be consumed at the mobile device for recovery computations, which is undesirable.

A number of prior research papers offer different strategies in dealing with various problems associated with video streaming over the Internet [8], while others attempt to resolve similar difficulties in the wireless domain [9,10]. Besides not catering to the specific requirements for cross-domain end-to-end video streaming scenarios, these various proposals often involve extensive network parameter gathering and complex mathematical computation for the optimum video transmission rate. Given the shortcomings of the existing protocols in implementing video streaming across both domains, this paper proposes a split-domain video transmission protocol (SDVTP) to support video streaming applications that are equipped with adaptive rate adjustment controls and error resiliency tools over a hybrid wired and wireless link. This relatively simple transport-layer protocol is largely based on TCP semantics and it addresses the following issues:

- Maintaining video QoS amidst wired link congestion.
- Achieving relative fairness in wired link resource allocation.
- Coping with high bit error rate (BER) in wireless links.
- Minimizing the deployment of video error resilience tools at the mobile receiver.
- Exploiting the split wired–wireless domain architecture.

The rest of this paper is organized in the following manner. Details of the proposed SDVTP are presented in Section 2, including initialization of congestion window size, congestion control algorithm, retransmission and timeouts, proxy and mobile device processing. Section 3 describes the computer simulation model setup and presents the simulation results to demonstrate the performance of the proposed protocol in comparison with TCP, UDP, and other existing options. Concluding remarks and further plans for this research project are given in Section 4.

## 2. The proposed protocol

Although open-looped video error resiliency tools such as forward error correction (FEC) and packet interleaving provide added protection against wireless channel errors, it is achieved at the expense of extra processing and battery power at the mobile devices. Thus it may be advantageous to couple the error resilient encoding algorithms with a transmission protocol that applies retransmissions and rate control for video transmission. Therefore, the main objective of the proposed SDVTP is to implement such a closed-loop control mechanism, i.e., dynamic source rate adjustment and retransmissions, which minimizes the invocation of open loop error control at the mobile device during end-to-end video streaming. Because of the drastic disparity in operating conditions between the wired and wireless domains, a single end-to-end solution is intuitively not very effective. Therefore, a split protocol approach is suggested with the aid of a cross-domain proxy, similar to the one advocated by Wireless Application Protocol (WAP) [11], as shown in Fig. 2. Other features of the proposed protocol include:

- Close collaboration between transport protocol and video encoder.
- Unicast window-based congestion control with a large initial window.
- Adaptive source encoding with respect to the congestion window.
- Use of selective acknowledgements (SACKs) in reporting packet arrivals.
- Network probing through receiver window size changes.
- Full acknowledgement of corrupted or lost packets after unsuccessful retransmissions.

To simplify the system model, the following assumptions are made:

- Both the intermediate proxy and the mobile device have sufficient physical memory for buffering;

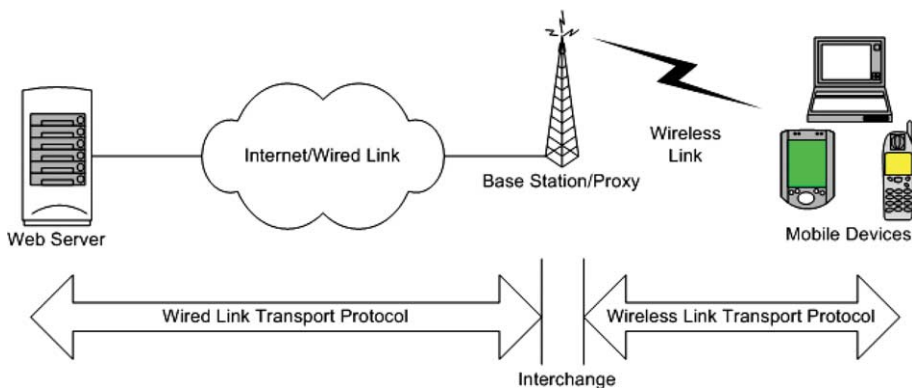


Fig. 2. SDVTP system overview.

- Each SDVTP segment has the same header structure as a TCP/IP packet, and it encapsulates only one video packet to achieve simplicity in performance comparison of simulation results;
- This study focuses only on the steady state transfer of video streams. Other aspects such as connection management and inter-cell handoffs are not considered at this time;
- Accurate measurements of round trip time (RTT) are readily available for calculating the correct dimensions of the congestion window, timeout period, and receiver window.

### 2.1. Initialization and congestion window profile

For each video streaming session, the upper and lower bounds on the source encoding rate are predetermined by the application as constants. The minimum playback rate is defined as the lowest video streaming rate at which the minimum video QoS requirements are guaranteed. For a mobile device at QCIF ( $176 \times 144$  pixels) format, 32 kbps should be adequate as the barely acceptable minimum transmission rate. The maximum playback rate is set as the highest possible video streaming rate, which is likely determined by mobile device parameters and channel conditions. The rate values are translated into congestion and receiver window sizes via multiplying the RTT for each domain, so it is imperative to obtain an accurate RTT estimate, possibly through the use of methods suggested in [12]. Note that two receiver windows are used in SDVTP: one for the wired link that is managed by the proxy, and the other for the wireless link that is regulated by the mobile device. For the sake of clarity, they are separately named throughout this paper as *proxy receiver window* and *mobile device receiver window*.

We start by first examining the behaviour of the wired link transport protocol between the web server and the proxy. After an initial connection establishment, the video encoder at the web server outputs a video stream at the maximum or user-defined playback rate. Each video packet is encapsulated in a TCP/IP structure, as shown in Fig. 3, branded with a timestamp to identify its relative position within the video stream, and then queued in the sender buffer to await transmission. Since random bit errors within the video data portion can be recovered or concealed via error resiliency tools, the checksum field of the TCP/IP header should be limited to provide cyclic redundancy check (CRC) protection to the TCP/IP header only to expedite the checksum process at the mobile device.

Fig. 4 illustrates the desired SDVTP congestion window profile. Initially, the proxy receiver window size is set by the proxy to support the default maximum play-

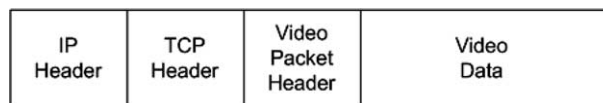


Fig. 3. Encapsulation of video packets in TCP/IP.

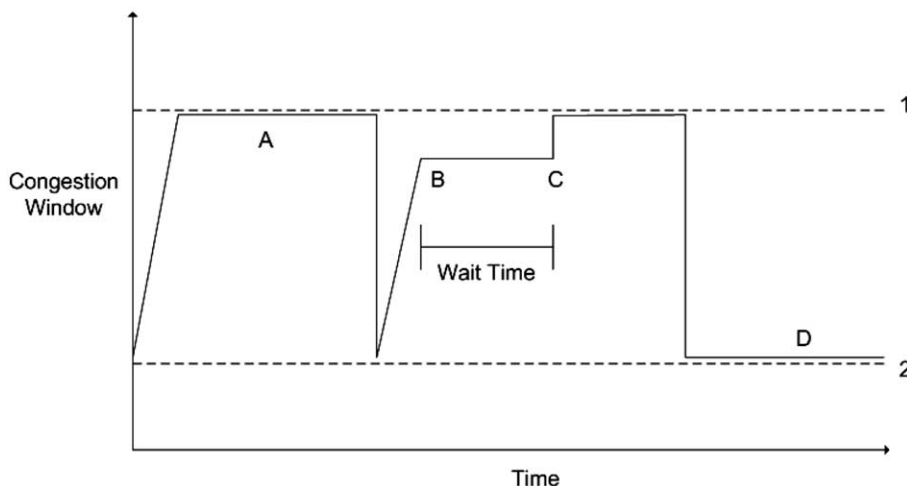


Fig. 4. Desired SDVTP congestion window profile.

back rate, for instance, 384 kbps, as indicated by line 1. The original intent of the receiver window in TCP is to indicate the amount of buffer space available at the receiver such that the sender can regulate the transmission rate to achieve flow control [1]. As mentioned previously, an abundance of physical memory is assumed to be allocated at the proxy, and the video streaming rate is kept at a relatively low level for mobile applications. Since the probability of proxy receiver buffer overflow is therefore reduced, the role of the receiver window field in the TCP header can be reconfigured to assist in maintaining a constant video streaming rate. Near the bottom of the diagram, line 2 specifies the initial congestion window size that supports the minimum playback rate. There are two advantages in setting the initial transmission window to be as large as the minimum QoS requirement for the video stream. First of all, it guarantees the quality of the video stream at worst to be barely acceptable to the user during highly congested situations. Although the proposed large initial window size may exceed the initial window limit set by TCP [13], it avoids video stoppage instances when the slow start algorithm resets the congestion window to the initial window size that leads to rebuffering delay. The second advantage is that, with a large initial window, the congestion window size will recover much faster during the exponential increase phase of the slow start algorithm, thus shortening the disconcerting glitches in video playback during the growth phase of the congestion window.

When transmission commences, the congestion window at the sender increases multiplicatively until it reaches the proxy receiver window size. Similar to TCP's slow start algorithm, successive SACK packets exponentially increase the sender's congestion window size. The video encoder at the web server dynamically references the congestion window size to compute the optimal video encoding rate that can adapt to changing network conditions. Unlike common TCP implementations, however, SDVTP does not implement the additive increase phase immediately after the expo-

nential growth period. Instead, the threshold value of the slow start algorithm is pegged against the advertised proxy receiver window size so that the congestion window can remain constant to avoid fluctuations that severely damage picture smoothness during video playback. Under favorable network conditions, the video sequence should be able to run at the maximum allowable end-to-end data rate, as indicated by Region A in Fig. 4.

When a timeout occurs due to increased network congestion, as indicated by Region B, the web server first sends a timeout notification to the proxy along with the congestion window size being set prior to the timeout event. The proxy should then decrease the proxy receiver window size accordingly to prevent further timeouts by restricting the maximum size the congestion window can grow, thereby reducing bandwidth consumption. The sender congestion window will then retract to the initial window size and go through the slow start phase again. Because of the lack of the slow start threshold, the congestion window is able to enjoy rapid exponential recovery and bypass the slow linear growth stage. The exponential growth phase stops when the congestion window size reaches the newly advertised proxy receiver window size.

Since the proxy receiver window is less than the maximum allowable proxy receiver window size at Region C, further bandwidth increases are possible. Such increases are executed after a pre-defined *wait time* since the last congestion window increase. The length of wait time and the scope of proxy receiver window increases are primarily dependent on user preferences and perceived QoS requirements. A shorter wait time and a larger proxy receiver window increase step-size imply greater aggressiveness in network bandwidth probing, and vice versa. If no timeout events occur during the next wait time, the proxy receiver window is then increased yet by another step-size. This process continues until the proxy receiver window reaches the maximum proxy receiver window size or a timeout occurs in the meantime. In the latter case, the congestion window recovery process is repeated, though the wait time in this case is increased by 2 times of its original value. Any single timeout henceforth would further increase the wait time by another 2 times. If no timeouts occur during three consecutive increases of the proxy receiver window, the wait time will be reduced by half until it reaches the preset minimum. This proxy receiver window adjustment procedure is designed to ensure relatively constant video playback rate while allowing limited periodic network probing for higher bandwidth without resulting in rate oscillations, which is a considerable improvement over TCP's additive increase mechanism.

If network conditions deteriorate substantially to a point such that excessive loss occurs even at the lowest transmission rate as indicated in Region D, then video transmission is not viable at that instant. The connection can be terminated automatically when the detected packet loss rate exceeds a certain threshold while transmitting at the minimum rate, or manually by the understandably frustrated end user. However, since the proposed protocol exhibits UDP behaviour at the lowest transmission rate, persistent transmission of video packets may wrestle extra bandwidth from the busy network through the reduction of congestion windows of other TCP connections. Although this phenomenon may seem somewhat unfair, it can

enforce the inherent higher priority of video transmission over other TCP connections and guarantee the minimal video QoS without network assistance.

## 2.2. Transmission management, retransmissions, and timeouts

After transmission, the video packets, each identified by their TCP sequence numbers as well as video stream timestamps, should remain at the source buffer until a positive acknowledgement from the corresponding SACK packet is received. The role of the timestamps will become apparent later on, and their granularity can be represented upwards in the range of seconds. If retransmissions are indeed required as indicated by SACKs or timeouts, the sender would simply resend the requested video packets again without changing the TCP/IP header timestamp. Selective acknowledgements in SDVTP are based on the TCP SACK option [14] and are used for acknowledging multiple packet reception. Given the higher overhead required by SACK packets, the proxy should return a SACK after the reception of a predetermined number of packets, thus cumulatively acknowledging these receptions. If a number of video packets were delayed in the network, gaps would appear within the reception buffer, as shown in Fig. 5. To the sender, these gaps within the SACK block sequencing indicate the possible occurrence of increased network traffic, and retransmissions of the missing packets would be necessary. Upon reception of such SACK packets, the sender should immediately retransmit the packets indicated by the gaps without waiting for a full timeout. Yet because of the real-time nature of video traffic, some packet loss can be tolerated by bypassing the missing packet sequences.

Fig. 6 provides an example to illustrate transmission management on the wired portion of the hybrid link. In this diagram, the first number denotes the packet sequence number, whereas second number indicates the timestamp in terms of seconds. From the web server, video packets 86–88 belonging to the 12th second frame are transmitted and they will be queued upon arrival at the proxy in the order according to their sequence numbers as well as their timestamps. Due to traffic congestion, packet 87 is lost within the network while the other two arrive safely and are inserted to the appropriate slots within the queue. The subsequent SACK will then signal this packet loss via the gap in sequence numbers, which will prompt the web server to retransmit packet 87. In the meantime, the steady flow of video packets is forwarded from the proxy onto the wireless link and the timestamp of the packets at the front of the proxy buffer acts as a floating threshold that serves as a cutoff point for incoming packets from the web server. When the lost packet 87 does not arrive in time before the cutoff point at the proxy buffer where they are due to be relayed to the wireless domain, it will be considered correctly received by the proxy and future SACK packets will have the corresponding gaps removed to prevent further useless retransmission of the lost video packets from the web server.

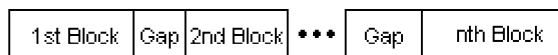


Fig. 5. Received packets at buffer.

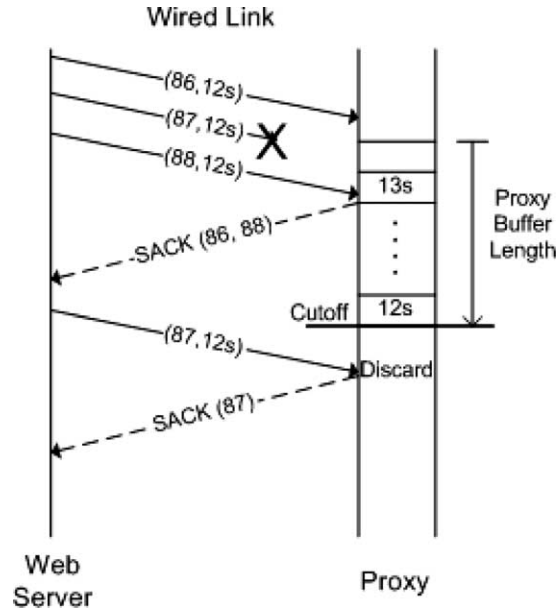


Fig. 6. Transmission management for the wired channel.

The retransmission mechanism is only suitable for odd lost packets. Persistent deterioration of network condition triggers significant packet loss and timeouts. To integrate with the congestion control algorithm, the timeout period must be set such that it permits rapid retransmission of video packets while still correctly interprets the changing dynamics of network traffic. The timeout interval should be set appropriately according to RTT estimates such that the sender can react to changing network conditions promptly. After a timeout, the sending rate is reduced accordingly, and all unacknowledged packets are continually retransmitted until a positive SACK is received by the sender.

### 2.3. Wireless link processing

Typically, the intermediate proxy simply passes every packet received from one domain onto the other domain. While this ensures simplicity in proxy implementation, it puts the burden of congestion control and retransmission mechanism squarely on the receiver, which in our case is a processing power-limited mobile device. In our system, the proxy is responsible for handling rigorous processing duties that span both wired and wireless domains. Besides managing two separate connections, the proxy should meticulously manage the interchange of packets between the domains. To minimize video streaming interruptions due to inter-cell handoff, the proxy should be located at the mobile switching center or some node higher in the network hierarchy so that adjacent base stations can share the same packet stream.

During steady state, the rate at which the packets are sent over the wireless link should mimic the sending rate of the original video server by referencing the timestamps of the received packets at the proxy. The correlation between the timestamps of consecutive video packets indicates the relative departure rates from the proxy to the mobile device. Since the packet arrival rate is presumably identical to the video playback rate, forwarding the packets at a high data rate over the wireless channel creates buffer overflow at the mobile device since the video packets are not consumed as rapidly as they are received. Also if packets are sent faster than they arrive, buffer underrun would occur at the proxy, which may leave gaps in the video packet stream arriving at the mobile device. Setting the wireless data rate lower than the video playback rate reverses the situation at the proxy and mobile device buffers, which is also undesirable. Therefore, the ideal solution is for the proxy to deliver video packets to the mobile device at a rate that is exactly the same as the server transmission rate.

The main problem with implementing TCP over wireless links is the slow start algorithm. Whenever packet loss occurs over the wireless link during high BER periods, the TCP sender misinterprets it as congestion and reduces the congestion window accordingly, whereas the correct approach is to increase frequency of retransmission to compensate for the noisy channel. To adapt to the wireless channel, the SDVTP approach is to eliminate the slow start algorithm altogether by equating the initial window to the advertised receiver window. Since the transmission window is kept constant to the receiver window, timeouts are not necessary, and retransmissions are handled solely by SACK packets.

Similar to the wired domain implementation, retransmissions are triggered by the gaps in sequence numbers indicated by SACK packets returned from the mobile device. An extremely noisy wireless channel causes frequent packet corruptions that lead to increased number of retransmissions. If channel deterioration persists, a backlog of unacknowledged video packet will appear at the proxy. The proxy keeps on retransmitting the unacknowledged packets until a positive SACK packet is received. Since the wireless channel is often bombarded with periods of high bit error loss, the frequency of SACK generation should be increased accordingly to recover corrupted video packets more expediently and to minimize messaging errors due to SACK packet loss. Such retransmission mechanism on the wireless link is very much like the link layer automatic retransmit request (ARQ) procedure, but SDVTP is able to hide any packet loss caused by radio channel errors from the video server, thereby eliminating the possibility of resetting the TCP congestion window.

Apart from managing the reception of video packets over the wireless TCP connection, the mobile terminal caches all received packets that are fed to the video decoder. Any erroneous receptions are recovered through video error resiliency tools, though the proposed transmission protocol should have minimized such algorithm invocations. Similar to the wired link approach, any missing packets at playback time is deemed as unrecoverable, and the corresponding gap within the SACK packet sequencing will be filled accordingly.

Fig. 7 puts all aspects of traffic management across the hybrid connection into perspective. Here, the left side of the diagram is exactly identical to Fig. 6, where packets received from the web server are placed in the proxy buffer to be forwarded

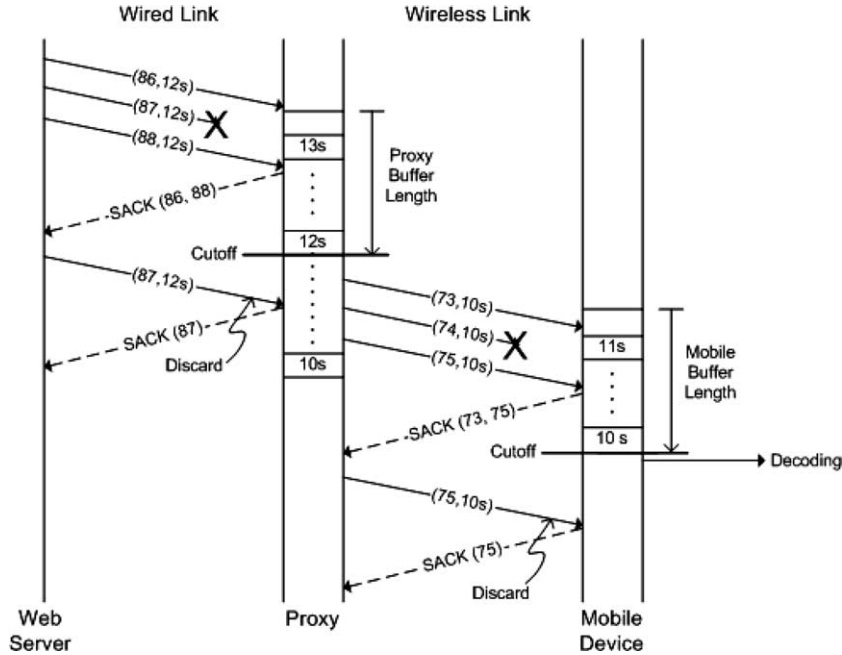


Fig. 7. Overall SDVTP transmission control.

to the mobile device. On the wireless link, packets 73–75 of the 10th second frame are being forwarded to the mobile device, but packet 74 is corrupted by channel errors. Therefore, the mobile device requests a retransmission of the lost packet as indicated by the sequence gap in SACK. However, the retransmitted packet 74 does not arrive in time for decoding perhaps due to channel latency or processing delay. As a result, it is discarded at the mobile receiver and its corresponding gap within SACKs is removed.

To summarize, the main advantage of SDVTP is its ability to isolate the cross-domain design issues with separate dedicated protocols that assures fairness in network resources sharing in the wired link and combats wireless channel errors simultaneously. It provides additional performance enhancements through the best effort delivery strategy and retransmission mechanisms. By integrating SDVTP features with the widely accepted TCP structure, the proposed protocol avoids undesirable network upgrades to the existing Internet infrastructure, and takes advantage of the vast amount of TCP research and application results currently available to assist future development of SDVTP.

### 3. Performance analysis

To evaluate the effectiveness of the proposed protocol, the performance of video transmission over the proposed protocol is compared with that of the same video

stream delivered over separate end-to-end UDP, end-to-end TCP, I-TCP [2], and TCP Snoop [3] connections via computer simulations.

### 3.1. Simulation system setup and parameters

The simulation system model for SDVTP consists of five components, shown in Fig. 8, which are constructed using the C programming language. Each of the system components (i.e., web server, wired link emulator, proxy, wireless link emulator, and mobile receiver) may run on individual terminals of a local area network (LAN), or be collectively executed in a single test bench machine. The interconnections between each system component should be reliable and stable with provisions for high traffic throughput. For simulating end-to-end SDVTP and UDP video streams, the interconnecting protocol of choice is UDP, while TCP is intuitively used for end-to-end TCP and TCP Snoop video sessions. TCP Snoop will implement a snoop agent at the proxy for suppressing duplicate SACKs and faster retransmissions. I-TCP will operate separate TCP connections over wired and wireless links, respectively. During a simulation run, test data streams originate from the sender, passing through the wired link emulator onto the proxy. Then the proxy delivers the packets via the wireless link emulator to the receiver. The implementation details about each of the system components are given in [15].

The simulation runs were conducted in the Broadband Communications Research Laboratory at the University of Waterloo. Each of the five network components ran on separate SUN Ultra 60 terminals with 512 RAM, all interconnected in a 100 Mbps LAN configuration. Typically, a simulation run involves transmitting imitated video data from the sender continuously for 400 s whilst exposed to the effects of a set of selected system parameters for that particular test case. The major system variables for the simulation model are wired link RTT, delay SACK frequency, and wireless link BER.

The wireless channel follows a simple two-state Markov chain that should be adequate to produce a good approximation for modeling the error process at the packet level in fading channels [9]. The packet loss rate in the wireless channel is related to the location of bit corruption within the video packet such that packet retransmission is required if the header portion is corrupted. In other words, a packet is not considered lost when there are no bit errors within the header portion, even though some of the video data may be corrupted. It follows that the computation for the burst error length becomes secondary as long as the probability of the first bit error occurring in the packet header is known.

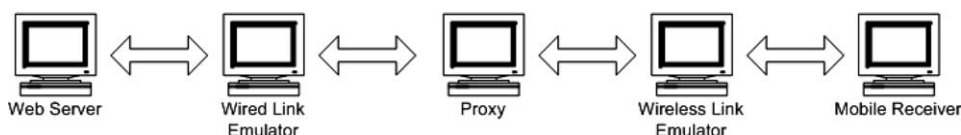


Fig. 8. Simulation system model configuration.

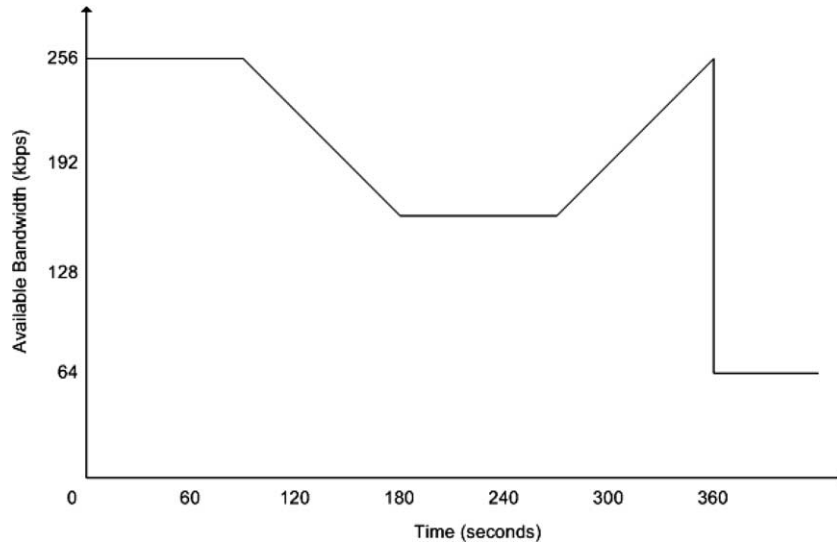


Fig. 9. Wired link congestion profile for simulations.

To demonstrate the performance of the five protocols in video streaming applications, test cases are constructed with various system parameters along with a wired link congestion profile, shown in Fig. 9, which represent different realistic network scenarios. They are:

- Maximum video streaming rate with no congestion (0–90 s).
- Gradually degrading network conditions (90–180 s).
- Intermediate video streaming rate allowed by the network (180–270 s).
- Gradually improving network conditions (270–360 s).
- Drastic reduction of video streaming rate to the minimum value due to heavy congestion (360–400 s).

By using the congestion profile to model traffic load patterns on the wired link, the simulation process can be simplified and all the test cases can be executed in a controlled and stable environment that produces more relevant results for meaningful comparisons.

Table 1 lists the testing attributes specific to each protocol, whereas Table 2 lists the important global system parameters used in simulations. In all test cases, the same imitated video source is used, and the wired and wireless domains are placed under identical sets of channel and congestion parameters.

### 3.2. UDP test results

In total, 9 UDP test cases are assembled with different combinations of data rate and wireless channel BER values as shown in Table 3. Since UDP connections con-

Table 1  
Specific protocol parameters for simulations

| Parameter name                  | SDVTP                | I-TCP                | TCP Snoop            | TCP                  | UDP             |
|---------------------------------|----------------------|----------------------|----------------------|----------------------|-----------------|
| Proxy buffer length             | 5 s                  | —                    | —                    | 0 s                  | 0 s             |
| Initial window                  | 64 kbps * RTT        | 3 * 1 kB             | 3 * 1 kB             | 3 * 1 kB             | —               |
| Loss window                     | 64 kbps * RTT        | 1 kB                 | 1 kB                 | 1 kB                 | —               |
| Retransmission timer length     | 2 * RTT              | 2 * RTT              | 2 * RTT              | 2 * RTT              | —               |
| Delay SACK frequency (wired)    | 1–4 Packets per SACK | 1–2 Packets per SACK | 1–2 Packets per SACK | 1–2 Packets per SACK | —               |
| Delay SACK frequency (wireless) | 4 Packets per SACK   | 4 Packets per SACK   | —                    | —                    | —               |
| Transmission rate               | Varies               | Varies               | Varies               | Varies               | 64,128,256 kbps |
| Probing step                    | 64 kbps              | —                    | —                    | —                    | —               |
| Minimum waiting time            | 15 s                 | —                    | —                    | —                    | —               |

Table 2  
Global system parameters for simulations

| Parameter name              | Value(s)              |
|-----------------------------|-----------------------|
| Video streaming rate range  | 64–256 kbps           |
| Wired link RTT              | 125, 250, 500 ms      |
| Wireless link RTT           | 2 ms                  |
| Video packet size           | 1024 bytes            |
| Video header ratio          | 0.25                  |
| Wireless link BER           | $10^{-4}$ – $10^{-6}$ |
| Mobile device buffer length | 5 s                   |

Table 3  
UDP test cases

| Test case | Data rate (kbps) | Wireless channel BER |
|-----------|------------------|----------------------|
| 1         | 64               | $10^{-4}$            |
| 2         | 64               | $10^{-5}$            |
| 3         | 64               | $10^{-6}$            |
| 4         | 128              | $10^{-4}$            |
| 5         | 128              | $10^{-5}$            |
| 6         | 128              | $10^{-6}$            |
| 7         | 256              | $10^{-4}$            |
| 8         | 256              | $10^{-5}$            |
| 9         | 256              | $10^{-6}$            |

tain no acknowledgement structure, simulating all test cases with a single wired link RTT value of 250 ms should be adequate to reveal meaningful results. As a graphical example, Fig. 10 shows the video packet throughput detected at the receiver and the overall packet loss (expressed as a percentage) in the end-to-end cross-domain connection of Test Case 7 (source data rate = 256 kbps, BER =  $10^{-4}$ ). The full set of numerical results for all the test cases is listed in Table 4.

When the video encoding rate is set at 64 kbps in Test Cases 1–3, which is below the minimum traffic threshold of the wired link congestion profile, the receiver throughput is generally able to maintain near 64 kbps for the majority of the playback duration, and the video QoS is therefore presumably preserved as demonstrated by the lower loss probability in Table 4. Still, the UDP connection fails to fully maximize bandwidth utilization because of the lack of a network traffic probing mechanism, whereas the absence of a packet loss recovery strategy leads to high packet loss and degradation of video QoS during episodes of high BER in the wireless channel. The most severe degradation occurs when BER equals  $10^{-4}$ , which leads to frequent instances of 100% packet loss per second that are detrimental to video quality even with the assistance of video error resilience tools.

When the video encoding rate at the sender is increased to 128 kbps in Test Cases 4–6, the wired link emulator begins to drop packets when the allowed throughput in the wired link is reduced to 64 kbps in the last 40 s of video streaming. During this period, the average packet loss, apart from those inflicted by wireless channel errors, ballooned to over 50% with the effective receiver throughput being reduced by more

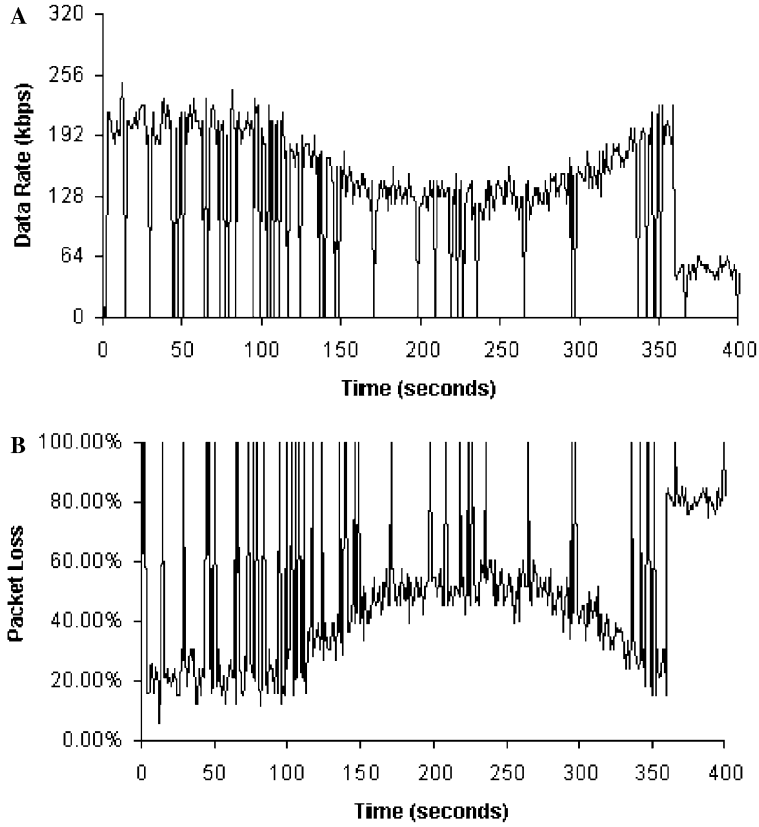


Fig. 10. UDP Test Case 7 results. (A) Throughput. (B) Overall packet loss.

Table 4  
Average throughput and loss probability statistics for UDP

| Test case       | Data rate (kbps) | Loss probability (%) |
|-----------------|------------------|----------------------|
| 1               | 47.16            | 23.59                |
| 2               | 59.90            | 2.71                 |
| 3               | 61.28            | 0.50                 |
| 4               | 92.46            | 28.02                |
| 5               | 113.38           | 7.39                 |
| 6               | 115.22           | 5.72                 |
| 7               | 135.10           | 48.65                |
| 8               | 186.68           | 27.19                |
| 9               | 191.94           | 25.07                |
| Overall average | 111.46           | 18.76                |

than half. Elsewhere, the throughput profile behaves in a similar manner as in Test Cases 1–3 with the random bit errors in the wireless channel producing the same disturbing effects and the bandwidth utilization not being maximized.

The rise of the video encoding rate to 256 kbps in Test Cases 7–9 further magnifies the problems experienced above. Additional packet losses are recorded amidst the wired link congestion profile modifications from 90 to 360 s. Although the packet loss rate may seem prominent, in reality the actual throughput may not suffer as much because the persistent UDP stream of video packets will gradually wrench extra bandwidth from other TCP-compliant connections that are sharing the same path, thereby infringing upon other Internet connections' right for a fair share of transmission bandwidth.

In summary, an end-to-end UDP connection is able to maintain video QoS only in the absence of wired link congestion, and is unable to absorb or rectify errors both during high wireless BER periods and heavily congested periods in the wired domain. Likewise, the notion of fairness in bandwidth sharing and the minimization of video error resilience tools deployment at the mobile receiver cannot be guaranteed by UDP. Overall, the UDP simulation results provided by the test cases vindicate the protocol's weaknesses pertinent to video streaming.

### 3.3. TCP/I-TCP/TCP Snoop test results

Similar to UDP simulations, a total of 9 test cases are constructed as listed in Table 5 for TCP, TCP Snoop, and I-TCP. Amongst them, the maximum number of video packets acknowledged per SACK is empirically determined to be 2. Although a value of 4 packets per SACK is theoretically feasible, it would induce frequent timeouts during simulations that eventually stalls the entire connection. A third performance criterion, namely congestion window changes, is also measured. Fig. 11 graphically presents the simulation results of Test Case 4 for TCP Snoop as an example. The average throughput and loss probability figures for the three protocols are separately listed in Tables 6 and 7, respectively.

Because of the similarities in characteristics amongst the three protocols, their simulation results generally behave in accordance with each other. As shown in Tables 6 and 7, TCP Snoop produces the best average throughput while sustaining the fewest packet loss, whereas TCP is the worst performer out of the trio. The reason is because both TCP Snoop and I-TCP shield the wireless channel errors from the web server so that frequent timeouts are prevented, while the congestion window of TCP

Table 5  
TCP/TCP Snoop/I-TCP test cases

| Test case | Packets per SACK | Wired link RTT (ms) | Wireless channel BER |
|-----------|------------------|---------------------|----------------------|
| 1         | 1                | 125                 | $10^{-4}$            |
| 2         | 1                | 125                 | $10^{-5}$            |
| 3         | 1                | 125                 | $10^{-6}$            |
| 4         | 2                | 250                 | $10^{-4}$            |
| 5         | 2                | 250                 | $10^{-5}$            |
| 6         | 2                | 250                 | $10^{-6}$            |
| 7         | 2                | 500                 | $10^{-4}$            |
| 8         | 2                | 500                 | $10^{-5}$            |
| 9         | 2                | 500                 | $10^{-6}$            |

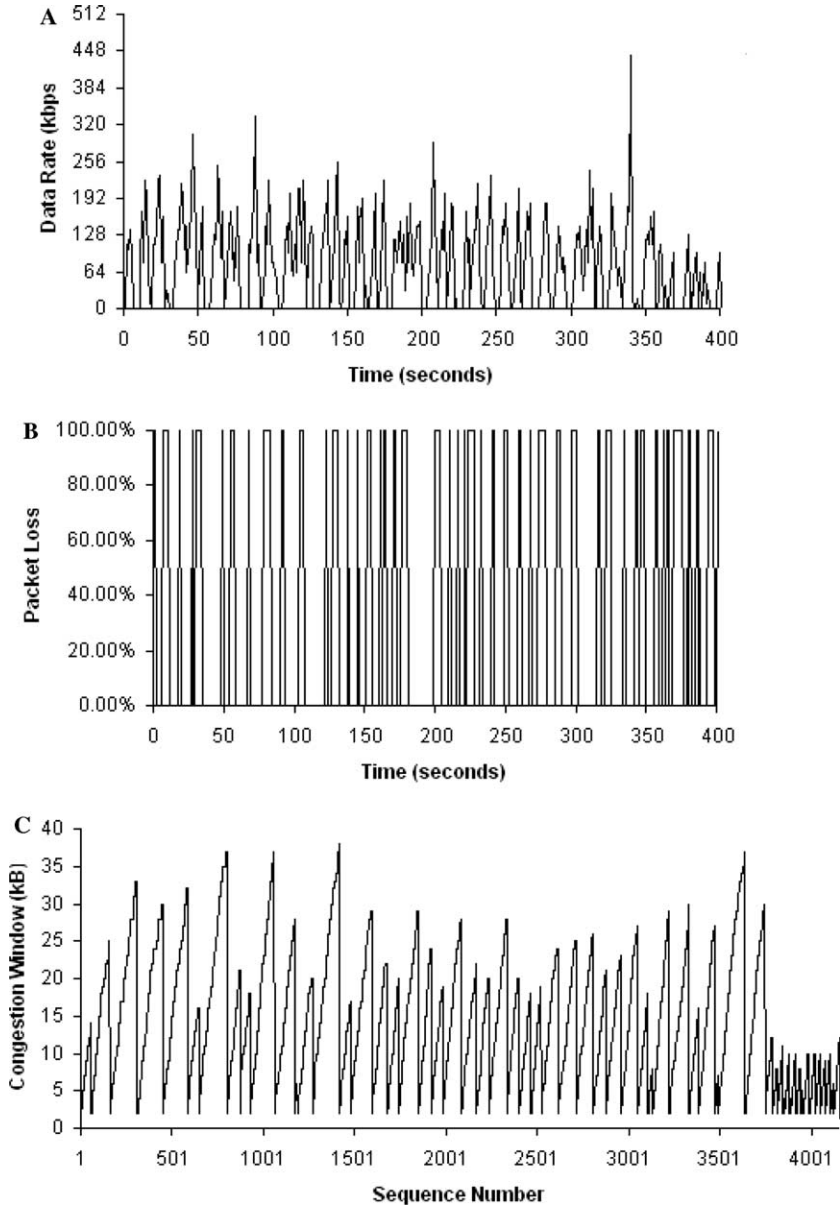


Fig. 11. TCP Snoop Test Case 4 results. (A) Throughput. (B) Overall packet loss. (C) Congestion window changes.

constantly fluctuates from packet losses stemming from both wired link congestion and wireless bit errors. For TCP Snoop, the detection agent at the proxy effectively suppresses negative SACKs by immediately forwarding any missing packets from its

Table 6  
Average throughput (in kbps) for TCP, I-TCP, and TCP Snoop

| Test case       | TCP    | I-TCP  | TCP Snoop |
|-----------------|--------|--------|-----------|
| 1               | 84.92  | 72.86  | 82.70     |
| 2               | 115.70 | 99.02  | 104.08    |
| 3               | 116.06 | 101.62 | 108.58    |
| 4               | 39.70  | 64.74  | 78.76     |
| 5               | 53.94  | 91.66  | 97.82     |
| 6               | 56.80  | 88.80  | 98.38     |
| 7               | 38.28  | 67.64  | 71.96     |
| 8               | 48.42  | 84.96  | 88.10     |
| 9               | 51.76  | 84.30  | 90.28     |
| Overall average | 67.29  | 83.96  | 91.18     |

Table 7  
Loss probability statistics for TCP, I-TCP, and TCP Snoop

| Test case       | TCP (%) | I-TCP (%) | TCP Snoop (%) |
|-----------------|---------|-----------|---------------|
| 1               | 44.13   | 39.17     | 18.51         |
| 2               | 29.54   | 22.01     | 2.81          |
| 3               | 31.51   | 19.85     | 3.03          |
| 4               | 73.30   | 41.37     | 30.56         |
| 5               | 63.03   | 29.78     | 10.56         |
| 6               | 61.52   | 29.14     | 14.28         |
| 7               | 76.43   | 45.63     | 36.26         |
| 8               | 73.87   | 33.32     | 21.31         |
| 9               | 72.88   | 30.06     | 19.39         |
| Overall average | 58.47   | 32.26     | 17.41         |

cache, thus increasing overall throughput and reducing packet loss. I-TCP does not perform as well as TCP Snoop in those categories since the wired link connection of I-TCP tends to increase its transmission rate far beyond 256 kbps more frequently than TCP Snoop because of shielding effects of a split connection, and its repercussions will be discussed later on.

Despite the disparities in input system parameters, most of the test runs are able to produce a receiver throughput that can, to a certain extent, follow the contours of the wired link congestion profile. This demonstrates TCP's revered ability to adapt to dynamic changes in network bandwidth, at least moderately. However, the rather rough throughput envelopes with intermittent spikes and valleys of varying data rates cause undesirable fluctuation in video quality and possibly frequent timeouts. The main culprit is the characteristic TCP saw-tooth additive increase multiplicative decrease (AIMD) congestion window pattern, which are central in all three protocols. Also, bursts of wireless channel errors that corrupt both video packets and SACKs that are mishandled by the sender as a flood of timeouts limit the growth of the congestion window to stifle data transmission rate.

Although TCP and its variants have often been relied upon to provide data transmission integrity in normal circumstances, unusually high packet losses (defined as

packets that do not arrive in time for playback processing at the receiver) have been detected in most of the tests. Closer examination on the simulation results exposes the fact that undeterred congestion window growth causes the instantaneous data rate to go beyond the 256 kbps threshold, thus leading to unnecessary packet loss in the wired link emulator. Packet stream integrity is also compromised during high BER periods in Test Cases 1, 4, and 7 as the sender fails to retransmit lost packets in time after long error bursts.

Compared with other scenarios, Test Cases 1–3 collectively generate the best receiver throughput envelope and the lowest packet loss rate, but they also cause the largest degree of congestion window fluctuation. More packet losses are recovered in Test Cases 1–3 because of the shorter RTT at 125 ms and larger buffer to RTT ratio that accommodates more retransmission attempts. Test Cases 7–9 produce the worst results in data throughput that are laden with isolated spikes of data intertwined with long lengths of connection stalls, even though congestion window fluctuations are kept at a smaller degree than the other test cases.

### 3.4. SDVTP test results

Table 8 shows the parameters of 12 test cases created for SDVTP simulations, which include the testing of the attribute of four packets per SACK. Similarly, Fig. 12 plots the resultant data of Test Case 4 against the three aforementioned performance criteria for visual comparison with previous examples. The results for average throughput and loss probability from all 12 test cases are collectively presented in Table 9.

Under similar testing conditions, SDVTP comprehensively outperforms the other four tested protocols in maintaining a higher and much more stable data throughput at the receiver that closely adheres to the wired link congestion profile via a highly structured and steady congestion control algorithm, while experiencing virtually no packet loss in all SDVTP test case results. In particular, the proxy excels in its role of regulating wired domain traffic in face of changing network dynamics and

Table 8  
SDVTP test cases

| Test case | Packets per SACK | Wired link RTT (ms) | Wireless channel BER |
|-----------|------------------|---------------------|----------------------|
| 1         | 1                | 125                 | $10^{-4}$            |
| 2         | 1                | 125                 | $10^{-5}$            |
| 3         | 1                | 125                 | $10^{-6}$            |
| 4         | 2                | 250                 | $10^{-4}$            |
| 5         | 2                | 250                 | $10^{-5}$            |
| 6         | 2                | 250                 | $10^{-6}$            |
| 7         | 2                | 500                 | $10^{-4}$            |
| 8         | 4                | 500                 | $10^{-4}$            |
| 9         | 2                | 500                 | $10^{-5}$            |
| 10        | 4                | 500                 | $10^{-5}$            |
| 11        | 2                | 500                 | $10^{-6}$            |
| 12        | 4                | 500                 | $10^{-6}$            |

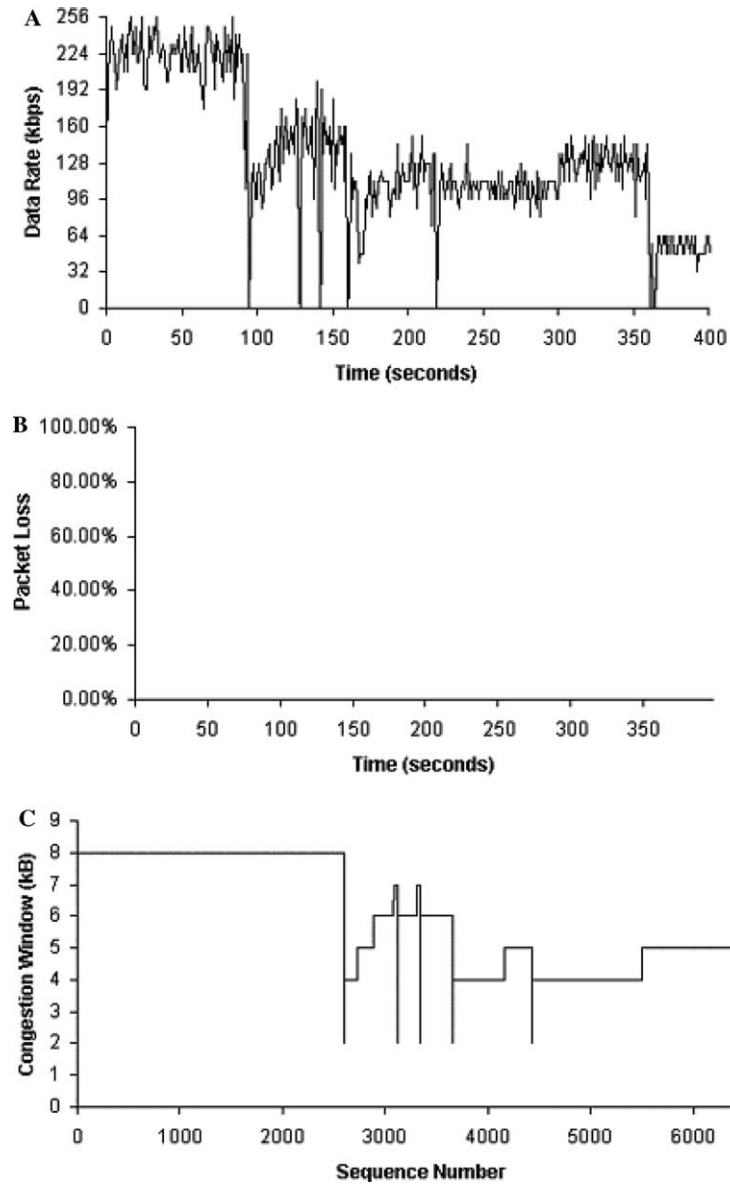


Fig. 12. SDVTP Test Case 4 results. (A) Throughput. (B) Overall packet loss rate. (C) Wired link congestion window changes.

overcoming bursts of wireless channel errors simultaneously. Again, the simulation results demonstrate the advantages of SDVTP as summarized in Section 2.

Still, the simulation results expose a few areas of minor concern. In Test Cases 1–3 with RTT at 125 ms, SDVTP falls short of achieving the intended maximum stream-

Table 9  
Average throughput (in kbps) and loss probability statistics for SDVTP

| Test case       | Throughput (kbps) | Loss probability (%) |
|-----------------|-------------------|----------------------|
| 1               | 117.84            | 0.02                 |
| 2               | 115.66            | 0.13                 |
| 3               | 114.22            | 0.02                 |
| 4               | 134.24            | 0.00                 |
| 5               | 136.08            | 0.00                 |
| 6               | 129.94            | 0.00                 |
| 7               | 101.00            | 0.00                 |
| 8               | 127.52            | 0.84                 |
| 9               | 115.90            | 0.00                 |
| 10              | 126.50            | 1.25                 |
| 11              | 106.14            | 0.36                 |
| 12              | 129.26            | 0.52                 |
| Overall average | 121.19            | 0.26                 |

ing rate in the first 90 s because of the higher ratio of processing time to RTT. The increase in processing time is due to the heavier computation load instigated by the short delay SACK frequency of 1. While this situation can be remedied by dynamic adjustment of RTT estimation in relation to the processing delay, it shows the vulnerability of SDVTP against an inaccurate RTT value.

Even with the luxury of a predetermined constant RTT value, there exist minor local fluctuations in data throughput due to subtle volatility in attaining a stable encoding rate. Though the degree of fluctuation is much smaller than those of TCP and alike, such slight changes would still be enough to inflict picture smoothness inconsistencies. In practical applications, SDVTP should more closely cooperate with the adaptive rate video encoder to ensure a smoother encoding rate in response to dynamically changing RTT values and SDVTP congestion window dimensions.

During the first 90 s of video streaming with the wired link congestion profile at its maximum value of 256 kbps, some test cases experience unexpected timeouts that render the congestion window to undergo the network probing phase. The cause of the initial timeouts is a combination of high video traffic and a low delay SACK frequency to RTT ratio. Because of high traffic volume at 256 kbps, packets often arrive at the proxy delayed and out of order, thus creating premature gaps in the proxy buffer. If the delay SACK frequency is small compared with the wired link RTT, then multiple SACKs containing the same gaps in their sequencing space are sent back to the sender. This leads to redundant and often unnecessary retransmissions that overwhelm the wired channel with useless data. The wired link emulator reacts by discarded all the excessive traffic that unfortunately creates more superfluous retransmissions at the proxy. Eventually a timeout occurs that reduces the sender output to stop the downward-spiralling cycle.

In SDVTP test cases with RTT equal to 500 ms, as expected, a smaller delay SACK frequency of 2 causes the sender to commit unnecessary retransmits, thus inundating the wired link with extra packets that lead to timeouts and degradation of overall performance. Interestingly though, this high level of sensitivity to packet

loss is of advantage when real packet loss occurs such that the corresponding packet recovery is much quicker. In comparison, a delay SACK frequency of 4 creates a less transmission overhead that is able to maximize congestion window dimensions and overall traffic throughput, but sustains a higher packet loss rate than its counterpart as a high delay SACK frequency value is less sensitive to packet loss. Therefore, in practical situations, the SDVTP should be able to dynamically determine the optimal delay SACK frequency to balance the trade-off between data throughput maximization and packet loss rate reduction.

In summary, based on the simulation results, SDVTP comprehensively outperforms the other four tested protocols under similar testing conditions in maintaining a higher and much more stable data throughput at the receiver. The data throughput closely adheres to the wired link congestion profile via the highly structured and steady congestion control algorithm, and there is virtually no packet loss in all the SDVTP test case results. In particular, the proxy excels in its role of regulating wired domain traffic in face of changing network dynamics and overcoming bursts of wireless channel errors simultaneously. However, there exist some minor concerns within the SDVTP protocol design that slightly hinder the overall performance, and further protocol enhancements should address these subtle issues.

#### **4. Conclusions and future work**

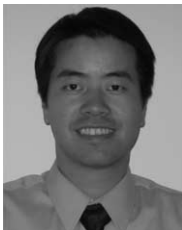
This paper proposes the split-domain video transmission protocol for adaptive rate video streaming over hybrid wired and wireless connections. The protocol aims to maintain video QoS amidst wired link congestion, achieve reasonable fairness in wired link resource allocation, cope with high bit error rate in wireless links, minimize the deployment of video error resilience tools at the mobile receiver, and exploit split wired–wireless domain architecture. The principal features of SDVTP include a split-domain approach, window-based congestion control with proxy assistance, large initial and loss windows in congestion control, adaptive rate video source encoding with respect to congestion window changes, and best effort loss recovery with selective acknowledgements. Evaluation of the proposed protocol is performed through rigorous simulations under various plausible source, network, channel, and client usage conditions. Simulation results indicate the viability of the suggested method for practical video streaming applications.

However, the current stage of performance analysis only generated numerical results with respect to data throughput stability and packet loss rates, not in terms of actual video quality at the mobile device decoder output. For the same BER and packet loss rate, the received video streams may produce different picture quality depending on where the bit errors occur within the stream. Therefore, the next step in research includes simulating with real video traffic on a more advanced testbed to evaluate the SDVTP protocol using video quality as the main performance metric. In addition, the scope of performance comparisons will be expanded to involve recently developed cross-domain video streaming protocols such as the ones described in [16,17], using a more sophisticated Internet traffic model that incorporates multiple

competing traffic flows as well as a more realistic wireless channel profile that accounts the effects of multipath fading and burst errors. Specific considerations such as connection management procedures, inter-cell handoffs, interactive user features (video seeking, security), and multicast capabilities would also be explored in further research projects to refine the rudimentary SDVTP specifications into a truly integrated cross-domain video streaming protocol.

## References

- [1] J. Postel, Transmission Control Protocol, RFC 0793, IETF, September 1981.
- [2] A. Bakre, B. Badrinath, Implementation of performance evaluation of indirect TCP, *IEEE Trans. Comput.* 46 (3) (1997) 260–278.
- [3] H. Balakrishnan et al., A comparison of mechanisms for improving TCP performance over wireless links, *IEEE/ACM Trans. Network.* 5 (6) (1997) 756–769.
- [4] J. Postel, User Datagram Protocol, RFC 0789, IETF, August 1980.
- [5] S. Floyd, K. Fall, Promoting the use of end-to-end congestion control in the internet, *IEEE/ACM Trans. Network.* 7 (4) (1999) 458–472.
- [6] R. Koenen, Overview of the MPEG-4 Standard, ISO/IEC JTC1/SC29/WG11, March 2001.
- [7] R. Talluri, Error-resilient video coding in the ISO MPEG-4 standard, *IEEE Commun. Mag.* 36 (6) (1998) 112–119.
- [8] J. Widmer, R. Denda, M. Mauve, A survey on TCP-friendly congestion control, *IEEE Network Mag.* 15 (3) (2001) 28–37.
- [9] C. Hsu, A. Ortega, M. Khansari, Rate control for robust video transmission over burst-error wireless channels, *IEEE J. Sel. Area. Commun.* 17 (5) (1999) 756–773.
- [10] K. Yu, D. Onishi, R. Van Dyck, Modification of ETFTP for MPEG-4 wireless video transport, *Proceedings of IEEE MILCOM 2* (1999) 1221–1225.
- [11] WAP Architecture, WAP Forum, July 2001.
- [12] V. Jacobson, R. Braden, D. Borman, TCP extensions for high performance, RFC 1323, IETF, May 1992.
- [13] M. Allman, V. Paxson, W. Stevens, TCP congestion control, RFC 2581, IETF, April 1999.
- [14] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, TCP selective acknowledgement options, RFC 2018, IETF, October 1996.
- [15] R. Ha, Split-domain TCP-friendly protocol for MPEG-4 adaptive rate video streaming over 3G networks, Master Thesis, Department of Electrical and Computer Engineering, University of Waterloo, September 2002.
- [16] L. Huang, F. Hartung, U. Horn, M. Kampmann, Proxy-based TCP-friendly streaming over mobile networks, *Proceedings of the 5th International Workshop on Wireless Mobile Multimedia* (2002) 17–24.
- [17] F. Yang, Q. Zhang, W. Zhu, Y.-Q. Zhang, End-to-end TCP-friendly streaming protocol and bit allocation for scalable video over wireless Internet, *IEEE J. Sel. Area. Commun.* 22 (4) (2004) 777–790.



BASc and MASc in computer engineering in 2000 and 2002, respectively, from University of Waterloo, where he is presently pursuing his Ph.D. degree. His current research interests include cross-layer design of wireless ad hoc and sensor networks.



Weihua Zhuang received the B.Sc. and M.Sc. degrees from Dalian Maritime University (China) and the Ph.D. degree from the University of New Brunswick (Canada), all in electrical engineering.

Since October 1993, she has been with the Department of Electrical and Computer Engineering, University of Waterloo, Canada, where she is a Full Professor. She was a Visiting Professor at the Department of Information Engineering, the Chinese University of Hong Kong, from August 2003 to June 2004. She is a co-author of the textbook *Wireless Communications and Networking* (Prentice Hall, 2003). Her current research interests include resource allocation and QoS provisioning for multimedia wireless communications.

Dr. Zhuang received the Premier's Research Excellence Award in 2001 from the Ontario Government for demonstrated excellence of scientific and academic contributions. She is an Editor of *IEEE Transactions of Wireless Communications*, and an Associate Editor of *IEEE Transactions on Vehicular Technology* and *EURASIP Journal on Wireless Communications and Networking*.