Split-Domain TCP-Friendly Protocol For MPEG-4 Adaptive Rate Video Streaming Over 3G Networks

by Rick Wan Kei Ha

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Master of Applied Science

in

Electrical Engineering

Waterloo, Ontario, Canada, 2002

©Rick Wan Kei Ha 2002

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The imminent inception of third-generation (3G) mobile communication networks offers 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. A split-domain TCP-friendly streaming video transmission protocol is presented based on adaptive rate encoding in the MPEG-4 video format. Network simulations are conducted to demonstrate the benefits and viability of such a video streaming scheme over existing options. Further feature enhancements and refinements are necessary for the proposed protocol to achieve its full potential.

Acknowledgements

There are a number of individuals and institutions that I would like to express my deepest gratitude in making my endeavour in graduate studies such a relished experience. They are:

- Dr. Weihua Zhuang, my supervisor, for providing me with valuable advice and superb guidance in the research process,
- Dr. Guang Gong and Dr. Sagar Naik, my thesis readers, for contributing their valuable time and effort in perfecting my work,
- University of Waterloo and her many fine educators and staff, for enabling me to pursue top-quality undergraduate and graduate education at such an enlightening environment,
- GEOIDE, for its continuous financial support,
- Students that I taught, for not giving me a hard time during my T.A. tenures,
- And finally, family, friends and colleagues, for showing me a life outside of books, Internet and research.

Dedication

To Karen (Ah Wai), my soul (sole) mate in life.

Table of Contents

List of Tables

List of Figures

1 Introduction

The advent and proliferation of the Internet in the last decade have tremendously facilitated the spread of information amongst individuals throughout the world. At the same time, the explosive growth of personal wireless communication systems has rendered them 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 seamless Internet connections to mobile users.

However, because of the intrinsic differences that exist between wired and wireless domains, granting the Internet connection does not translate into full access to Internet resources and capabilities, such as for video streaming and other multimedia applications. For instance, 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.

Because of historical factors and other technical limitations, the currently implemented protocols such as Transmission Control Protocol (TCP) [2] and User Datagram Protocol (UDP) [5] are deemed inadequate in providing a genuine end-to-end cross-domain solution for video streaming. A number of research papers offer different strategies in dealing with the various problems associated with video streaming over the Internet [9], while others attempt to resolve similar difficulties in the wireless domain [10] [11]. 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 to determine the optimum video transmission rate for the instantaneous Internet traffic profile. Given the inadequacies of the existing protocols in implementing video streaming across both domains, this thesis proposes a relatively simple split-domain TCP-friendly protocol based on TCP. The protocol is optimized for the recently developed MPEG-4 video encoding standard [7] [8] to achieve video streaming over a hybrid wired and wireless link while addressing the following issues:

- Maintaining video QoS amidst wired network congestion

  Network congestion in the wired domain results in excessive transmission delay and packet loss that can seriously corrupt cross-domain connections. For delay-sensitive loss-tolerant data such as video traffic, odd packet losses can be tolerated or concealed through algorithmic recovery schemes. However, prolonged network congestion will severely degrade video quality, so a reduction in source video rate may be undertaken to maintain acceptable video QoS. The video streaming server should also be able to adapt to improving network conditions through gradually increasing the encoding rate for better video QoS. In either case, rate fluctuations must be minimized in order to guarantee video playback smoothness.

- Achieving fairness (or TCP-friendliness) in wired network resource allocation

  Although video streaming is inherently demanding to have a higher priority over other forms of Internet traffic, excessive bandwidth allocation for video transmission stifles other data connections. In other words, such network unfairness guarantees video quality at the expense of degrading the QoS of other sessions. It follows that all the video streaming applications should follow the principle of TCP-friendliness such that all of the connections receive a fair share of the total available bandwidth while still guaranteeing an acceptable level of video QoS.

- Coping with high bit error rate (BER) in wireless links

  Wireless transmission is subject to fading and delay dispersion, as well as interference and background noise, all of which lead to high BER on wireless data transmission. During high BER intervals, video packets are bombarded with random errors over their entire payload length. In case of MPEG-4 streaming, corruption to the TCP/IP header section or the header portion of the MPEG-4 video segment results in the dropping of the entire packet, while other errors can be partially recovered by MPEG-4 recovery tools. The video streaming application should rectify any packet loss occurred, possibly through frequent retransmissions, with promptness and efficiency.

- Minimizing the deployment of MPEG-4 error resilience tools at the mobile receiver

  Although packet losses in the transmission channel can be mitigated through MPEG-4 recovery mechanisms, these error resiliency tools should be kept as a backup strategy when other measures fail since the invocation of these corrective algorithms will considerably raise the demand for processing power and memory space at the mobile device. Therefore the video streaming application should rely on other less computation-intensive schemes for packet loss recovery and regard the MPEG-4 resiliency tools as a last resort.

- Exploiting split wired-wireless domain architecture

  Since each of the domains requires different sets of matching design criteria in video streaming, a split domain architectural approach for protocol design should be explored. The intermediate proxy can be enhanced to assume a more elevated role on regulating video streaming traffic between the two distinct domains.

The rest of the thesis is organized in the following manner. A summary of the various protocol design considerations and challenges regarding wired and wireless domains is provided in Chapter 2, while Chapter 3 provides a general description of some of the existing and proposed video streaming methods, and discusses their effectiveness and drawbacks. Next, a brief overview of the MPEG-4 video encoding standard is given in Chapter 4. The specifics of the proposed split-domain TCP-friendly protocol are presented in detail in Chapter 5, with the corresponding test strategies and simulations results placed in Chapter 6. Finally, Chapter 7 gives some concluding remarks and outlook for further research work.

2 Current Internet Video Streaming Environment

The modern Internet is essentially a ubiquitous, self-sustaining mesh that interconnects electronic computing devices around the world. Within this massive heterogeneous inter-network structure that is woven from a myriad of sub-networks such as the cross-continental trunks, local area networks, wireless wide area networks, satellite connections, etc., there exist two distinctive transmission domains: wired and wireless. Before delving into defining the specification for a cross-domain video transmission protocol, it is important to first closely examine the characteristics of each domain and identify the obstacles in implementing video streaming in both environments.

2.1 Wired Domain

Optical fibres, a transmission medium that is deemed to provide high-bandwidth transmission and low bit error rates, constitute the majority of the global network backbone structure. The probability of random bit errors occurring in data transmissions over this wired trunk is extremely low (~10^{-10}) [18] and therefore can be effectively approximated to zero. Despite the assumed absence of random bit errors, data transmission can be corrupted by the persistent likelihood of network congestion. Congestion happens when a large number of connections, both delay sensitive (real-time video, audio streams) and non-delay sensitive (text content, email, FTP, Telnet), contend for limited available bandwidth. Excess bandwidth demand leads to the occurrence of packet loss due to buffer overflow at intermediate routers. To reduce such undesired instances, congestion control should be implemented on the Internet transmission protocols so that when congestion occurs, data senders can throttle the data rates accordingly to prevent unnecessary packet loss and achieve overall network stability.

For video applications, however, frequent fluctuations in sending rate can seriously affect the perceived picture quality at the end user terminal, as illustrated in Figure 1. If the server is encoding the video stream at a constant rate, then intuitively the available video transmission rate should be at least as large as the encoding rate for optimum playback results. Unlike common bulk file transfers, streaming video is less demanding in achieving the maximum bandwidth possible since the video encoding rate remains relatively constant, as dictated by predetermined QoS settings, in response to increasing bandwidth availability. Excessive abundance of network bandwidth in comparison to a modest video encoding rate leads to possible bandwidth under-utilization that fails to exploit further video quality enhancements through the increase of video encoding rate. On the other hand, a reduction in the video transmission rate can create buffer underrun at the receiver. Eventually, the user will experience possible stoppage of video playback stream when sudden bursts of extremely congested traffic cause high packet loss at intermediate routers. So care must be taken in video application design to compensate for the congestion effects while maintaining acceptable video delivery QoS.

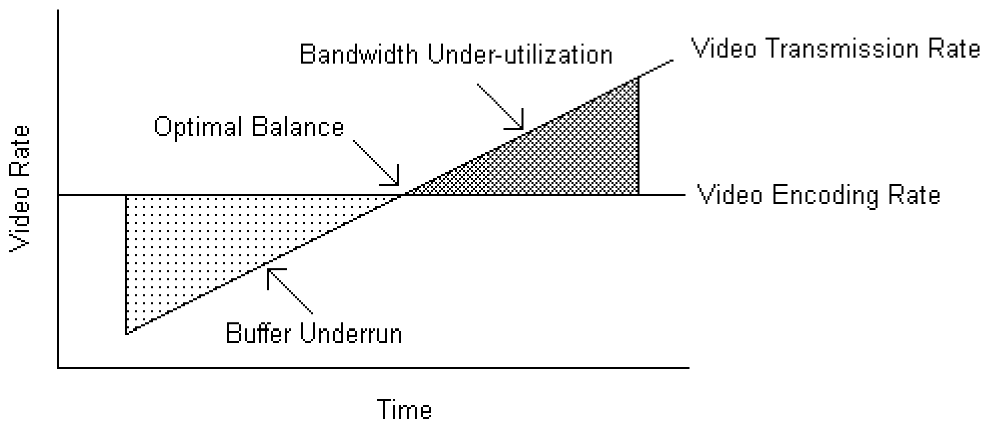


Figure 1: Effects of video transmission rate changes vs. video encoding rate.

When packet loss due to network congestion is detected during transmission of loss-sensitive media, the lost packets must be retransmitted to preserve data integrity. However, video streaming can tolerate some degree of data loss as a possible trade-off for guaranteeing real-time delivery. Often, video encoding algorithms are equipped with complex data resiliency tools to recover or conceal errors stemmed from packet loss or corruption. Therefore it is not necessary for the video transmission protocol to enforce stringent requirements on the retransmission mechanism. For instance, as Internet connections often span across the continents, the sheer transmission distance increases the round trip time (RTT) considerably. Whenever retransmissions are required during congestion periods, the high turnaround time experienced by the video packet may cause it to arrive at the receiver too late to be processed, thus forcibly discarded. As a result, before initiating retransmissions, the video transmission protocol should evaluate the probability of successful processing of retransmitted packets at the receiver to prevent such wasteful episodes from occurring.

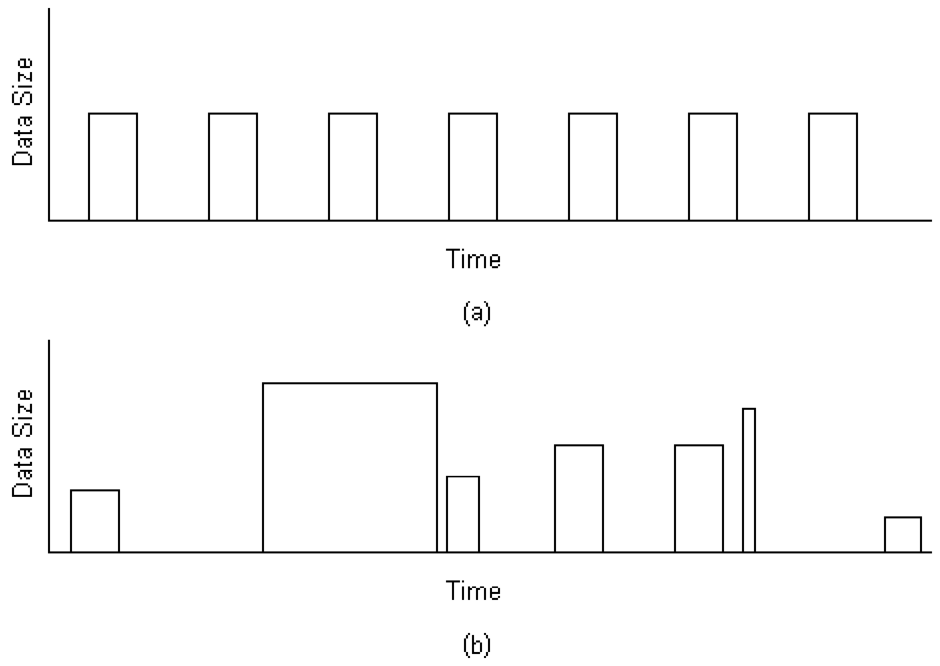


Figure 2: Video packet arrival patterns. (a) Ideal (b) Bursty.

Besides combating against network congestion, video streaming applications also need to be resilient against bursty arrivals of video packets. Ideally, video packets should arrive in regular intervals such that the inter-packet arrival time should exhibit little variance, as shown in Figure 2(a). Under this condition, only a small buffer space is needed for queuing the incoming video packets at the video decoder. However, when a video stream is passed through a long multi-hop Internet connection, the ever-changing network dynamics and routing path fluctuations introduce inter-packet delay jitter that increases the arrival time variance between consecutive video packets at the receiver. In addition, large bursts of video traffic can gather and arrive at the receiver within a short time span, as illustrated in Figure 2(b). If the receiver buffer space is not adequately allocated, then the video stream faces the risks of playback stoppage caused by extended inter-packet delay jitter that leads to buffer underrun, or lost data because of buffer overflow during high arrival bursts of video packets. Therefore during video streaming sessions, abundant buffer space should be made available at the receiver to absorb these bursty transmission effects.

While commonly the buffer length is set by the receiver based on user preference, its value in turn determines the number of retransmissions allowed for a given connection that ultimately affects the robustness of video packet transmission. As mentioned before, retransmissions are required whenever video packets are lost during congested periods. In that case, a longer buffer would naturally allow more retransmissions, as illustrated in Figure 3, and therefore ensuring high video stream integrity. In addition, a short RTT implies that a video packet can be acknowledged in the shortest amount of time, thereby permitting additional retransmissions during heavily congestion periods to expedite data packet recovery. However, smaller computing devices may not have the luxury to afford large buffer spaces, and the end user may have to tolerate the longer buffering time at the beginning of the video stream. So the design decision on optimal buffer length depends on connection path RTT, available storage resources at the video decoder, and user preference.

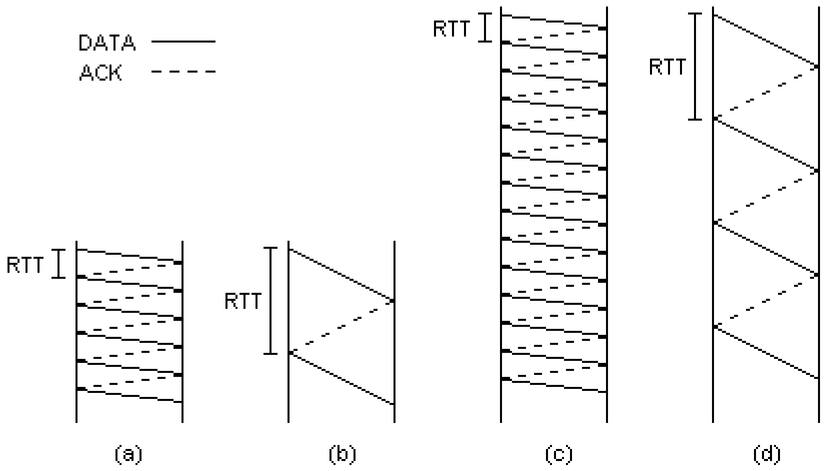


Figure 3: Relationship between buffer length and RTT. (a) Short buffer with short RTT. (b) Short buffer with long RTT. (c) Long buffer with short RTT. (d) Long buffer with long RTT.

2.2 Wireless Domain

Compared to its wired counterpart, the wireless network poses additional design challenges to implement video streaming since it suffers from a number of intrinsic deficiencies and shortcomings. First of all, the scarce radio spectrum within the wireless domain limits the capacity and bandwidth of wireless data transmission. For the impending third-generation (3G) mobile communications standards, the International Telecommunication Union (ITU) stipulates data rate requirements of up to 384 kbps outdoors and up to 2 Mbps indoors [19]. While the increase in available transmission rate is a major upgrade over previous first and second generation systems [20], it is still a far cry from the transmission bandwidth offering by fibre optics that is measured upwards in units of gigabytes/s. Therefore, bandwidth-intensive applications such as video streaming cannot be implemented over wireless links without severely degrading video quality as a trade-off for lower available transmission rate.

Second, during propagation through the physically open environment, radio waves are easily prone to attenuating effects such as path fading, multipath dispersion, cochannel interference and background noise [21]. As a result, random bit errors and burst errors frequently appear to corrupt data packets and instigate packet loss in the wireless channel. When detected, the mobile device must discard the corrupted packets and request for retransmission. In addition, heavy data traffic over the wireless link increases the overall interference level in 3G Wideband Code Division Multiple Access (W-CDMA) systems, thus degrading the signal-to-noise ratio and elevating the effective bit error rate (BER). A common approach to reduce BER is to apply error-correcting channel coding techniques such as Forward Error Correction codes (FEC) [17] at the link layer. Even with such added protection, the resultant BER over the wireless link can still reach up to 10^{-2} [22].

Third, wireless devices are often equipped with constrained processing power and battery life along with limited memory space and display screen size. Therefore the ideal transmission protocol for wireless devices should be simple, lightweight, and involve as few transmission overhead and buffering space as possible while not sacrificing any video QoS requirements. Also, the video decoder on the mobile device has to execute as few instructions and algorithm computations as possible to conserve power. In short, the processing burden for video streaming should be alleviated from mobile devices as much as possible with assistance from a more simplified transmission protocol structure and more efficient video encoding algorithms, as well as active involvement from fixed hosts such as the video streaming server and intermediate cross-domain proxy.

Fourth, while the cellular structure for wireless wide area network (WAN) offers user the ability to roam about the coverage area, the inter-cell handoff process, shown in Figure 4, can get quite complicated, especially for data transmission. During a video streaming session, the mobile user first obtains a unique identification marker, known as an IP address [24], which binds the wireless device to the service access point for that particular cell for the video packets to be delivered correctly to the receiver. As the mobile user switches from one cell to another amidst video data transfer, the mobile device either binds the current IP address to the new service access point, or a new IP address is assigned to the mobile user for the new service access point. In the first case, the routing of video packets tends to become problematic, considering the mobile device may continue to switch cells and incur frequent handoffs during a connection session. The second case is even worse since it is almost impossible to maintain end-to-end transport and higher-layer connections. Although these handoff issues are adequately handled by the Mobile IP protocol [23] and other ad hoc cellular network schemes currently employed by mobile telecommunications providers, there exists the problem of buffer space allocation at the intermediate proxy for split-domain video streaming connections during handoffs which is to be explained further in later section.

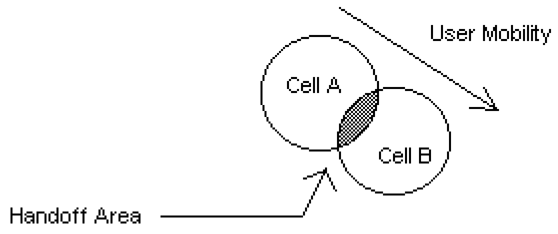


Figure 4: Inter-cell handoff.

Apart from the above adverse effects, the wireless network does possess one prominent design advantage over the wired network. That is, the wireless link between the mobile user and intermediate proxy often constitutes the last lop of a connection linking a fixed host web server located a long distance away. Since the wireless domain only involves several short network hops, the variance of delay jitter tends to remain relatively stable as long as the intermediate proxy can send the video packets at a constant rate to offset the delay jitter effects spilled over from the wired domain. Also because of the short distance between the base station and the mobile device, retransmissions due to packet corruption will be prompt and direct as the RTT in the wireless domain is short. Therefore the amount of buffer space required at the mobile terminal can be reduced accordingly.

2.3 End-to-End Cross-Domain Considerations

The main considerations and challenges in providing video transmission separately over the wired and wireless domains are summarized in Table 1.

Table 1: Main design considerations and challenges for wire and wireless domains.

	Wired Domain	**Wireless Domain**
Advantages	• High transmission bandwidth • Low BER	• Shorter RTT, faster retransmissions • Non-bursty arrival of video packets (assume no spill-over effects from wired network)
Drawbacks	1. Congestion causes transmission rate fluctuations 2. Longer RTT, slower retransmissions 3. Bursty arrival of video packets	• Low transmission bandwidth • High BER • Limited power and processing capability for wireless devices • Handoff complexity

As discussed in the above sections, the disparities in design characteristics between both domains make it evident that superimposing a single protocol over an end-to-end cross-domain video streaming connection (see Figure 5) is considerably difficult. Not only does the protocol have to exploit the design advantages for both domains, it has to harmonize the respective drawbacks such that the integration of the two domains does not exacerbate their disruptive effects further. For example, the bursty arrival pattern for video transmission in the wired domain will propagate over to the wireless link if the intermediate proxy does not intervene. As a result, the resource-strapped mobile device has to bear the load of providing large buffering storage and possibly requesting and processing more retransmissions, each of which is highly undesirable.

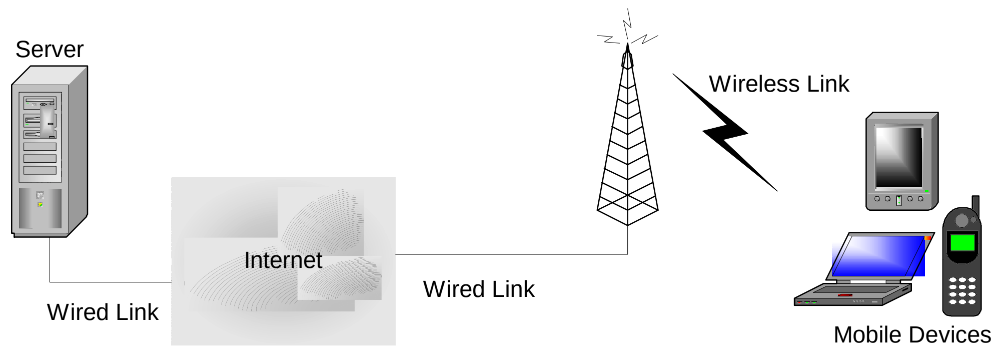


Figure 5: Cross-domain video connection.

Also for cross-domain video transmission, end-to-end delay takes into account the total time taken to traverse the network from the streaming server to the mobile receiver that includes transmission, buffering and processing delays. For real-time applications such as live broadcasting and video conferencing, prolonged end-to-end delay leads to poor QoS performance. For other video streaming applications, increasing the buffer size is an effective way to offset the negative effects of delay jitter. In return, buffering delay is increased accordingly that leads to an overall increase of end-to-end delay. Extended end-to-end delay protection should not affect the video stream quality, albeit a potentially lengthy buffering wait at the beginning of playback.

Another major issue that plagues cross-domain video connections is the urgency to maintain end-to-end security. As the link between the wired and wireless domains hinges at the intermediate proxy, it is very vulnerable to gateway attacks or unscrupulous system insiders attempting to steal original data that can gravely compromise end-to-end network security, even with other transport and application level security mechanisms protecting each of the two parts [25]. The optimum solution is to involve the intermediate proxy as little as possible in the end-to-end traffic processing. In other words, the intermediate proxy should simply act as a relay for video packets without breaking the end-to-end connection semantics or intervening in packet assembly and data processing. However, this approach may sacrifice the transmission efficiency on the wireless link as a single unbroken end-to-end connection often creates conflicts between wired and wireless domain characteristics.

Despite the monumental challenges faced in creating an end-to-end cross-domain Internet video transmission protocol, different proposals surfaced over the years that offer mixed approaches in supporting video streaming applications over either or both domains [9] [10] [11]. The next chapter explores the principal protocols, TCP and UDP, their technical features pertinent to video transmission, and their degree of adaptation to video streaming.

3 Current Internet Video Streaming Protocols

Any newly proposed video streaming protocol must coexist with other resident Internet transmission protocols since it is technically and financially improbable to overhaul the entire Internet to incorporate new protocols. Because of their design maturity and performance stability, current end-to-end video streaming applications mostly rely on TCP and UDP as the main vehicles for video packet delivery over the Internet [1]. Thoroughly investigating the specifics of these two protocols related to video streaming assists in defining a new video streaming protocol that is able to incorporate proven design methodologies and remedy deficiencies.

3.1 TCP

TCP [2] is a connection-oriented transmission protocol that was first contrived with the intent of operating over terrestrial wired networks to guarantee data transmission integrity while maintaining overall network stability, particularly through the universal deployment of the TCP congestion control [14]. Without this regulatory mechanism to promote voluntary restraint on sender transmission rates, unrestricted data flows would inundate the Internet to cause pervasive buffer overflows at the intermediate routers that eventually lead to congestion collapse [6]. Even if the network is able to handle the swarming traffic, the receiver needs a means to notify the sender of its buffer capacity such that it would not be overwhelmed with the incoming packets.

3.1.1 Congestion Control

The clever design of TCP congestion control is able to solve both problems simultaneously by maintaining two transmission windows, known as receiver window and congestion window, at the sender. The receiver window value is advertised by the receiver to signify the amount of data it is willing to accept, and the congestion window is periodically computed to determine the amount of data the sender can transmit without causing network congestion. The number of bytes that may actually be sent is the minimum of the two windows. Consider Table 2, which outlines the sender actions based on congestion window and receiver window, as an example on how the deliberation process works. With the receiver window size set at 8 kB and the congestion window at 4 kB, it is clear in the first case that the receiver has ample buffer space to digest the smaller incoming batch of data. Therefore the sender is given permission to send the whole 4 kB of data. In the second case, since both the congestion window and receiver window are listed at 8 kB, it is also safe to transmit the entire packet load. The third case has the congestion window larger than the receiver window. As a result, the sender is allowed to send the full amount advertised by the receiver window, which is 8 kB.

Table 2: Sender actions based on congestion window and receiver window.

Case	Congestion Window	Receiver Window	Sender Action
1	4 kB	8 kB	Sends 4 kB of data
2	8 kB	8 kB	Sends 8 kB of data
3	16 kB	8 kB	Sends 8 kB of data

While the receiver window is often set based on the amount of buffer space available at the receiver, the congestion window is dynamically determined by the slow start and congestion avoidance algorithms [14] during a data transmission session. When a connection is established, the sender initializes the congestion window to the size of the initial window, which usually is equivalent to the maximum segment size in use on the connection, where a segment is defined as any TCP/IP data or acknowledgment packet, or both. Afterwards, it sends one maximum segment and waits for either an acknowledgement (ACK) from the receiver or a loss event. If this segment is acknowledged before a loss event is sensed, it increases the congestion window by one maximum segment to make it two maximum size segments and sends two segments. The congestion window is subsequently increased by one maximum segment size as each of these segments is acknowledged. With the congestion window is *n* segments, if all *n* are acknowledged on time, the congestion window is increased by the byte count corresponding to *n* segments. In other words, each burst of successfully acknowledged packets effectively doubles the congestion window. The exponential growth of the congestion window continues until either a loss event occurs or the receiver's window is reached.

The goal of this algorithm, called slow start, is to determine the optimal congestion window size without incurring network congestion by using this trial-and-error method. For instance, the network condition permits congestion window sizes of *n*, *2n*, *4n* and *8n* bytes without any hitches, but complains (through loss events) when a congestion window of *16n* bytes is used by the sender. Even though the receiver window is advertised at a much large size, the sender should settle with a maximum congestion window of *8n* nonetheless.

In addition to the congestion and receiver windows, TCP congestion control also employs a third parameter, the threshold, for network probing and congestion avoidance. Initially it is arbitrarily set at a default value. When a loss event is detected, the threshold is set to half of the current congestion window, and the congestion window is reset to the value of loss window that always equals to one maximum segment. The slow start algorithm is then restarted to probe the network limits, until the exponential growth of the congestion window reaches the threshold.

Henceforth, successive acknowledged segments increase the congestion window linearly (by one maximum segment for each burst of *r* segments) instead of one per segment. Another way is to increment the congestion window by 1 full-sized segment per RTT so that the sender does not need to maintain an extra state variable for segment count. Together, this process is known as the congestion avoidance algorithm and it tentatively increases the congestion window in linear mode to reach the previous threshold level and beyond until a loss event occurs.

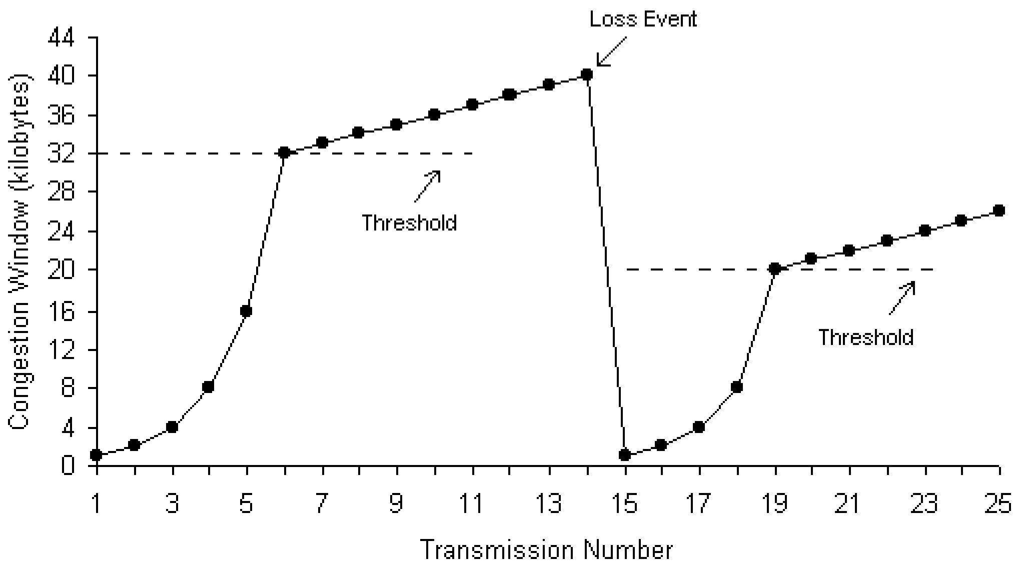


Figure 6: Example of TCP congestion control algorithm.

To better illustrate how TCP congestion control works, its sample characteristics are shown in Figure 6 as an example. For this connection, the maximum segment size is 1 kB. The congestion window starts at the initial window of 1 kB and grows exponentially until it reaches the threshold at 32 kB. Afterwards it increments linearly till the detection of a loss event. Immediately the congestion window reverts back to the loss window of 1 kB and restarts the slow start algorithm with the threshold set at half its original value to 20 kB. If no more loss event occurs, the congestion window will continue to grow up to the size of the receiver window.

After that, it will stop growing and remain constant as long as no more loss events occur and the receiver window does not change size.

3.1.2 Acknowledgement Generation and Loss Recovery

While the original purpose of the congestion control algorithm is to avoid brewing network congestion by controlling the sender transmission rate, a loss event eventually will happen. In other words, a transmitted segment has been lost in the network en route to the receiver. In response, the sender must retransmit the lost segments to preserve data integrity. The primary means in notifying the sender the occurrence of a loss event is through the retransmission timer. During data transmission, the receiver returns ACKs to the sender to notify the successful delivery of a particular segment. For increased Internet efficiency, the receiver can implement delayed ACK, which is to send fewer than one ACK per data segment received [28]. Whenever a segment is sent through TCP, the corresponding retransmission timer is started at the sender. If the segment is acknowledged before the retransmission timer expires, the timer is stopped. On the other hand, if the retransmission timer goes off before the ACK comes in, the segment is retransmitted with a restarted retransmission timer. The duration of the retransmission timer is computed dynamically according to the guidelines outlined in [26].

Typically, TCP uses a cumulative acknowledgement scheme in which received segments that are not at the left edge of the receive window are not acknowledged. Because of network dynamics, individual segments are often dropped or reordered in delivery sequence as they pass through a string of intermediate routers. When an out-of-order segment arrives at the destination, the receiver should send an immediate duplicate ACK, as shown in the sequence of events listed in Table 3. This ACK notifies the sender that a segment was received out-of-order and which sequence number is expected. In addition, the receiver should send an immediate ACK when the incoming segment completely or partially fills in a gap in the sequence space. This will generate more updated information for a sender recovering from a loss event through a regular retransmission timeout, a fast retransmit [14], or an experimental loss recovery algorithm, such as NewReno [27].

Table 3: TCP cumulative retransmission scheme.

Step No.	**Incoming Event**	**Receiver Action**
1	Packet *n* arrives	Acknowledges packet *n*
2	Packet *n+1* arrives	Acknowledges packet *n+1*
3	Packet *n+4* arrives	Acknowledges packet *n+1*
4	Packet *n+3* arrives	Acknowledges packet *n+1*
5	Packet *n+2* arrives	Acknowledges packet *n+2*

Specifically, if the sender receives multiple duplicate ACKs, the fast retransmit algorithm should be invoked to repair the loss. In practice, the arrival of 3 duplicate ACKs, i.e. 4 identical ACKs received without the arrival of any other intervening packets, is treated as an indication that a segment has been lost. Immediately the sender retransmits the assumed missing segment without waiting for the retransmission timer to expire. After the fast retransmit, instead of reverting the congestion window to the loss window according to the slow start algorithm, the congestion window should follow the fast recovery algorithm [14] that handles the transmission of new data until the reception of a non-duplicate ACK. The reason is that since the receiver can only generate a duplicate ACK when a segment has arrived, it shows that segment has left the network and it is no longer consuming network resources. Therefore it is not necessary to resort to drastic bandwidth reduction measures like slow start algorithm at that moment. Furthermore, since the ACK clocking is left unchanged, the sender can continue to transmit new segments, though transmission must continue using a reduced congestion window as computed below.

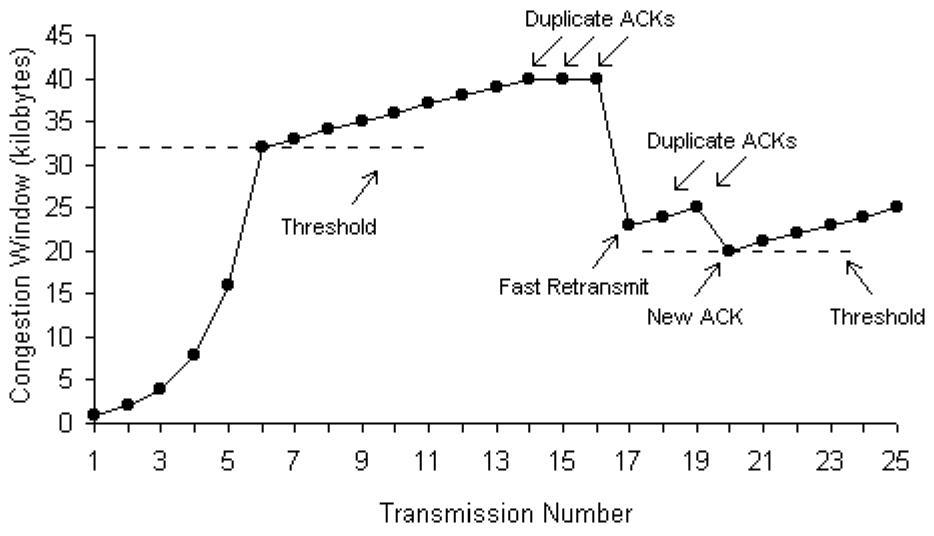


Figure 7: Example on congestion window changes during fast retransmit and fast recovery.

Figure 7 shows an example on how the fast retransmit and fast recovery algorithms are implemented together. From transmissions 1 to 14, the congestion window grows as dictated by the slow start and congestion avoidance algorithms. A segment is not received in-sequence at the receiver, thus duplicate ACKs start to appear at the sender in transmission 15. When the third duplicate ACK is received in transmission 17, the fast recovery begins with the threshold set to no more than half of the current congestion window. The lost segment is then immediately retransmitted and the congestion window is set to equal to the threshold plus 3 maximum segments. This intentionally inflates the congestion window by the number of segments, which is three, that have left the network and which the receiver has buffered. Further duplicate ACKs are received in transmissions 18 and 19, for which each increments the congestion window by another maximum segment. This intentional increase again refers to the additional segment that has left the network. If the new value of the congestion window and the receiver window permit, the sender may transmit a segment. In transmission 20, a non-duplicate ACK arrives, and it should be the acknowledgment originally expected by the sender, and it should acknowledge all the intermediate segments sent between the lost segment and the receipt of the third duplicate ACK, if none of these were lost. As a result, the congestion window is set to the threshold in order to deflate it. Thereafter, the congestion window proceeds according to the congestion avoidance algorithm to indicate the end of the fast recovery period.

The combination of fast transmit and fast recovery algorithms elevate the transmission efficiency and channel throughput over the sole use of retransmission timers in hastening retransmissions while not compromising the congestion window size as extreme as the slow start algorithm. However, in face of multiple losses in a single burst of packets, this combination is found not to recover very efficiently in such situations [29]. In general, multiple packet losses from a window of data, stemming from severe packet dropping at intermediate routers or burst errors on wireless channel (to be discussed below), could have a catastrophic effect on TCP throughput such that the TCP cumulative acknowledgment scheme forces the sender to either wait a RTT to find out about each lost packet, or to unnecessarily retransmit segments which have been correctly delivered to the receiver. Therefore, this prolonged round-trip wait for multiple dropped segments causes the retransmission timers to expire, thus reducing overall throughput.

The NewReno algorithm [27], as mentioned earlier, is an experimental loss recovery algorithm that slightly modifies the fast recovery algorithm to address this problem. In particular, it identifies the use of a partial ACK that acknowledges some but not all of the segments that were unacknowledged at the beginning of the fast recovery period. Instead of deflating the congestion window, partial ACKs received during the fast recovery period are interpreted to show that the packet immediately after the acknowledged segment in the sequence space has been lost and should be retransmitted. This procedure proceeds with retransmitting one lost segment per RTT until all of the lost segments from that window have been acknowledged.

Still, these additional changes do not solve the fundamental problem that the TCP sender can only learn about and retransmit at most one lost segment per RTT. A simpler and more intuitive solution is to use a selective acknowledgment (SACK) mechanism, combined with a selective repeat retransmission policy to overcome these limitations [15]. In this scheme, the receiver compiles a SACK packet that informs the sender about all the segments that have arrived successfully, so the sender needs to retransmit only the missing data segments. During the TCP connection establishment phase, the sender and receiver must mutually agree upon exercising the SACK option. When multiple segment losses occur during data transmission, sequence number gaps will appear in between contiguous blocks of correctly received segments at the receiver buffer, as illustrated in Figure 8(a). The construction of the SACK packet in relation to the receiver buffer profile is shown in Figure 8(b), in which the inter-block relationship is contained within the options area of a TCP packet. The left edge of a block is the first sequence number of this block, whereas the right edge of a block is the sequence number immediately following the last sequence number of this block. The segment that triggered the SACK is contained in the first block, unless that segment advanced the acknowledgment number field in the TCP header, to assure SACK reflects the most recent change in the receiver buffer. Upon reception of a SACK packet, the sender should immediately retransmit the missing segments without resetting the retransmission timer. In short, implementing the SACK option does not interfere with normal operations of the congestion control mechanisms while still being able to combat multiple segment losses.

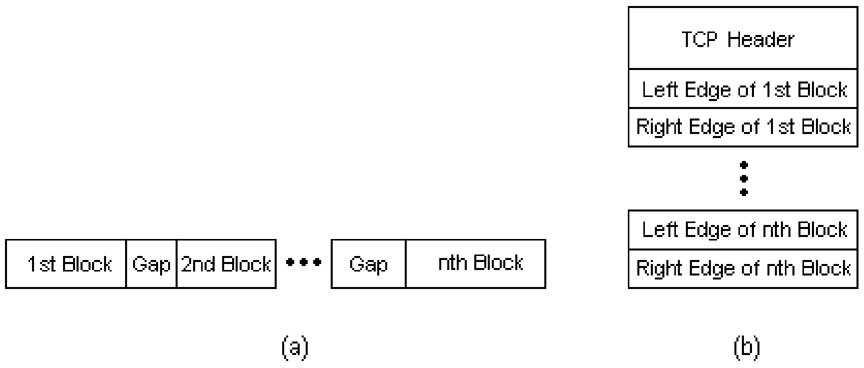


Figure 8: Multiple segment losses. (a) Receiver buffer profile. (b) Corresponding SACK packet.

3.1.3 TCP over Wireless Links

One of the reasons behind TCP's domination in regulating Internet traffic can be attributed to robust protocol design based upon a number of clearly defined underlying assumptions [2] [30], one of which is that TCP runs on network conditions pertinent to the wired domain, as discussed in Chapter 2. However, applying the same set of TCP specifications over wireless communication links leads to a list of problems. When the wireless domain is filled with high BER and burst errors that leads to multiple packet loss, the correct approach is to resend the same packets as vigorously as the sender can until the receiver confirms their arrival [17]. The problem with TCP over wireless links is its assumption that all packet loss is the result of network congestion rather than packet corruption [31]. In response to wireless BER, the TCP sender first attempts fast retransmits of the missing segments. If this does not correct the situation, the retransmission timer will expire, causing the sender to reset the congestion window and revert back to the slow start algorithm. Not only did TCP fail to resolve the packet corruption problem, it exacerbates the situation by degrading the overall connection throughput and transmission performance.

Beside adopting the TCP fast retransmit and fast recovery algorithms and SACK option, as described in the previous section, to deal with the high BER in the wireless channel, a number of other proposals have suggested TCP enhancements and other mechanisms to accommodate wireless domain characteristics. They can be classified into several categories:

- Split the end-to-end TCP connection into wired and wireless parts to implement separate transmission protocols optimized for each domain [3] [16] [34].
- Enable network routers to generate explicit congestion or packet corruption messages to inform the sender to take relevant recovery actions [35] [36].
- Rely on link-layer packet recovery mechanisms on the wireless link that automatically retransmits lost packets to preserve the TCP end-to-end connection [4] [32] [33].

Each set of solutions has its own strengths and drawbacks. For instance, the split approach can effectively decouple the TCP congestion control with BER effects by separating the TCP connection, but it is also plagued with issues such as additional proxy design and handoff complexity, and security concerns as discussed in Chapter 2. The second approach also allows the TCP sender to differentiate congestion and corruption instances, but any modifications to existing network routing structure are always highly undesirable. While the link-layer approach preserves end-to-end TCP semantics that avoids much of the problems surrounding the previous two approaches, it does not fully eradicate the possibility of ever invoking the congestion control algorithm at the sender due to wireless BER. In short, no matter how much the end-to-end TCP connection is bombarded with network congestion or wireless link BER, it is evident that the main goal is to avoid invoking the slow start algorithm as much as possible (since it is detrimental to the overall transmission throughput) while still maintaining data integrity.

3.1.4 Adaptation to Video Streaming

TCP has been instrumental in allowing the spontaneous, decentralized growth of the Internet since its birth while still maintaining overall network stability. However, based on the above discussions about the various TCP features, it can be concluded that TCP is not optimized for video streaming applications. First of all, the perpetual changes in the TCP congestion window, caused by the congestion control algorithm, introduce undesirable fluctuations and other side effects in video transmission rate. Typically during steady state operations, the TCP congestion window, bounded by the congestion avoidance algorithm, continues to increase linearly beyond the threshold until a loss event is detected. As a result, the saw-tooth profile of the TCP congestion window, shown in Figure 9, makes it difficult to maintain a constant video streaming rate that is critical in providing acceptable picture QoS.

After a loss event, the congestion window reverts back to the size of the loss window, causing a dramatic decrease in video transmission rate. For the congestion window to climb back to the previous size before the loss event if network conditions permit, it has to endure the linear congestion avoidance phase that may take a long time, as illustrated in Figure 10. In this figure, line 1 denotes the congestion window size that sustains the current video streaming rate. While this recovery period may seem insignificant for connections with short RTTs, users on distant links with much longer RTTs will certainly notice the painstaking climb through since the prolonged loss of transmission bandwidth causes rebuffering timeouts at the receiver.

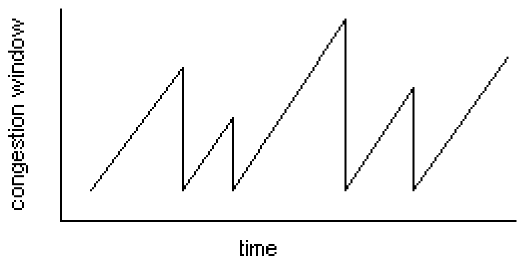


Figure 9: Saw-tooth profile of TCP congestion window.

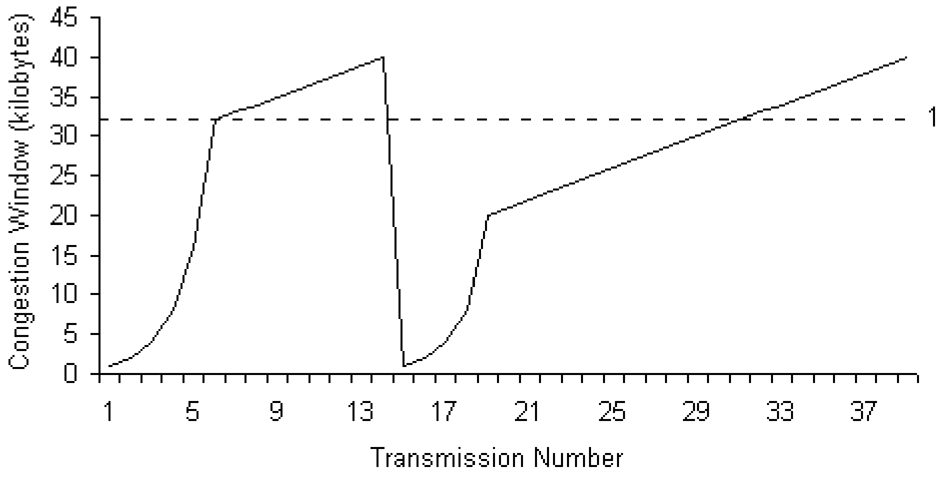


Figure 10: Congestion window recovery after loss event.

Another general example on the undesired fluctuating effects of TCP congestion control is demonstrated in Figure 11. Lines 1 and 2 represent the maximum and minimum congestion window sizes that support the maximum and minimum video streaming rates, respectively. Line 3 denotes the loss window size that is equivalent to the size of one maximum segment. The receiver window is set to be higher than line 1. In region A, the network traffic is able to support the video playback at a rate that lies between lines 1 and 2, but the congestion window's continuous growth triggers frequent loss events. Because of this, the congestion window is not able to maintain a steady transmission rate greater than the minimum video streaming rate and possibly causing buffering underrun at the receiver. Region B shows an even grimmer situation where the network is not permitting the congestion window to grow beyond line 2, thereby effectively disrupting normal video streaming since the minimum playback rate cannot be sustained. The best network conditions are shown in region C where the congestion window is able to grow above line 1 to achieve the maximum video streaming rate. However, the congestion window will take a much longer time to recover from any future loss event because of the linear congestion avoidance algorithm (refer to Figure 10)

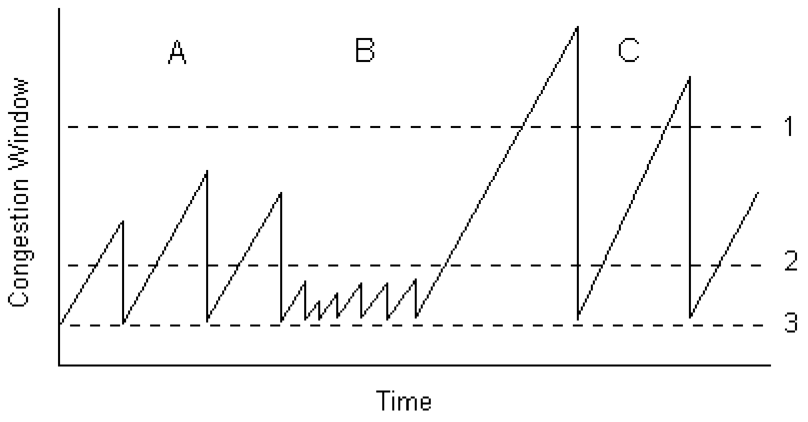


Figure 11: Example on undesired effects of TCP congestion control.

Secondly, the TCP retransmission scheme does not exploit the loss-tolerant nature of video traffic such that unnecessary retransmissions can be minimized. As discussed previously, TCP goes to great lengths to guarantee data integrity through a number of loss recovery schemes. However, because of the loss-tolerant nature of video streaming combined with advanced error recovery tools in video encoding standards such as MPEG-4, data integrity becomes lower in priority than providing in-time delivery of video packets. In situations where a video packet is consistently deemed lost but originally scheduled to be consumed by the video decoder, any further retransmissions of this packet will be meaningless and therefore unnecessary. However, TCP fails to comprehend this unique situation and will continue to execute useless retransmissions that otherwise can be bypassed.

Finally, in the cross-domain connections, TCP cannot distinguish the source of packet loss as due to congestion in the wired network or random bit errors in the wireless network, thus degrading the overall throughput. Other proposed TCP enhancements or new protocols discussed in Section 3.1.3 do alleviate the problem somewhat, but they do not specifically target video streaming applications. In summary, current TCP implementations, while effective in most cases, face a number of drawbacks in providing robust cross-domain video streaming connections. Additional changes to current TCP versions are necessary to deal with congestion window fluctuations, redundant retransmissions and wireless domain issues.

3.2 UDP

Unlike TCP, UDP [5] is a lightweight connectionless transmission protocol that is more suitable for real-time video streaming since it is not hindered by any congestion control or retransmission mechanisms. The streaming rate is guaranteed to be constant and smooth at all times without the constraint of a congestion window structure. Also, UDP-based connections have higher priority over TCP-based connections in competing for network bandwidth because of the lack of UDP congestion control [1]. This argument can be demonstrated as follows. Within a heterogeneous connection environment, different TCP and UDP sessions compete for limited bandwidth resources that reach the equilibrium as seen in Figure 12(a). During heavily congested periods, shown in Figure 12(b), additional UDP and TCP session joins into the competition for network bandwidth, thereby upsetting the delicate network balance and creating packet loss. As long as the UDP sessions keep their respective data rates unchanged, other TCP-based sessions will eventually reduce their output rate as bounded by the congestion control algorithms. In the end, the unrelenting UDP sessions gain extra bandwidth share at the expense of voluntary retractions on behalf of other "generous" TCP sessions, as illustrated in Figure 12(c).

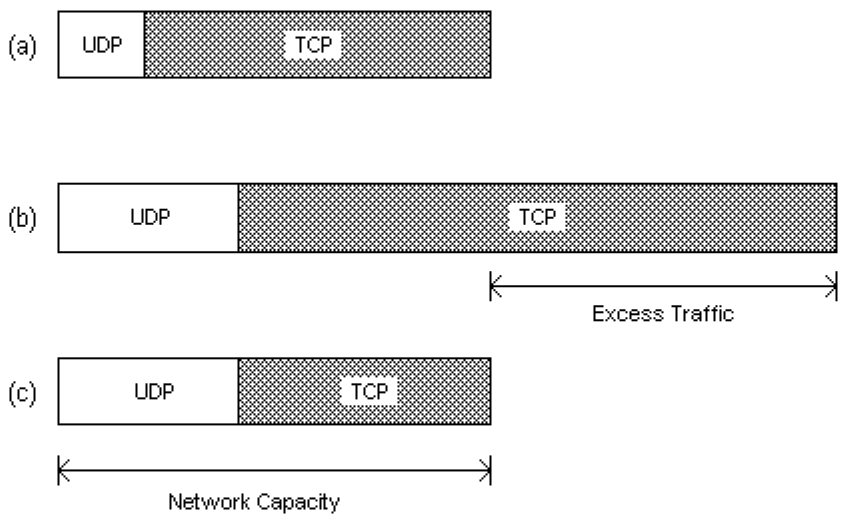


Figure 12: Competing UDP and TCP connections in heterogeneous network environment.

Despite these seemingly advantages over TCP, there exist a number of drawbacks, though, in the use of UDP on video streaming applications. First of all, without the restrictions of congestion control, multiple UDP sessions unfairly starve other TCP connections of network bandwidth. In the worst case, uncontrolled admission of new UDP sessions induces fierce network bandwidth competition that may eventually lead to congestion collapse [6]. Also since UDP video streaming rates are usually predetermined at the application level, they often fail to adapt to changing network dynamics. For instance, in Figure 13, a UDP video streaming session is outputting at a constant rate that is unchanged throughout the entire duration. The available network bandwidth, marked by diagonal lines, decreases initially and rises again during the middle of the UDP session. Apparently, the UDP session is wasting valuable network bandwidth that can improve playback quality near the start and finish of the video clip, and suffers from avoidable packet losses around the middle of the streaming session.

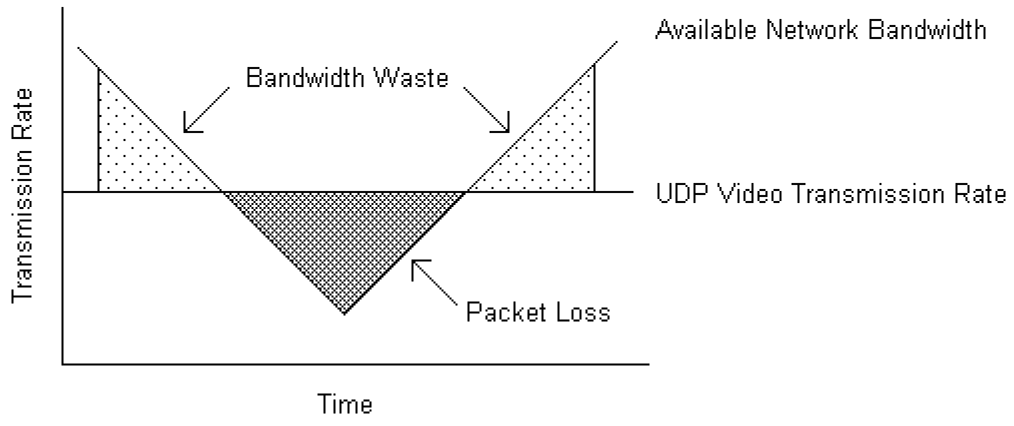


Figure 13: UDP video transmission behaviour.

Second, because UDP is so lightly equipped, its packet header lacks the essential processing details such as sequence number and timestamp for it to function properly in a video streaming session. As a result, the Real-Time Protocol (RTP) and Real-Time Control Protocol [37] [38] are developed to work on top of UDP/IP stack to provide more adapted video streaming services. However, these transport-level enhancements do not address the first problem above, so the RTP specification stipulates that the video streaming application implementers should take appropriate precautions to limit accidental bandwidth usage, which is sometimes difficult to monitor in reality.

Third, UDP connections tend to suffer high packet loss over noisy wireless links due to the lack of retransmissions, which substantially degrades video quality at the mobile host. Although the problem can be alleviated with the protection of error resiliency tools incorporated by video encoding standards such as MPEG-4 [7] [8], additional processor and battery power must be consumed at the mobile device for recovery computations, which may be undesirable as discussed in Section 2.2.

3.3 Cross-Domain Video Streaming Protocol Design Considerations

The deficiencies in applying TCP and UDP on video streaming applications breed new research into attempting to discover the optimum video transmission protocol with some focus on the wired domain [9] while others concentrate on the wireless link [10] [11]. From the discussions in Chapters 2 and 3, for cross-domain video streaming protocols, it is imperative to incorporate the following essential features:

- TCP-Friendliness - achieve optimal video streaming rate while maintaining overall fairness in bandwidth sharing with other Internet sessions in the wired domain
- Loss Recovery - follow best effort strategy in recovering lost video packets in both domains
- Cross-Domain Harmonization - compromise all the strengths and weaknesses that preside over both sections of the connection within a single protocol
- Lightweight Structure - avoid overburdening mobile device with excessive processing.

Before the specifications of the proposed protocol are presented, an overview of MPEG-4 standard and its error resiliency tools are presented in the next chapter. A good understanding into this standard helps to refine the design characteristics of the proposed protocol for better results in cross-domain video streaming.

4 MPEG-4 Overview

MPEG-4, a video compression standard developed by Moving Picture Experts Group (MPEG), is the latest algorithm that primarily targets low-bit-rate video applications up to 384 kbps for common interchange format (CIF) of 352 x 288 pixels resolution that is well-suited for mobile video devices over 3G wireless links [39]. Its development follows a couple of earlier video compression standards, namely MPEG-1 (interactive video on CD-ROM) [40] and MPEG-2 (the core compression technology underlying the transmission, storage and display of digitized moving images and sound tracks) [41], that are widely deployed in numerous applications. Besides focusing on low-bit-rate video communications, MPEG-4 provides functionalities that encompass all types of multimedia coding applications [8]. Specifically, they include:

1) the ability to efficiently encode mixed media data such as video, graphics, text, images, audio, and speech, called audiovisual objects (AVOs),
2) the ability to create a compelling multimedia presentation by compositing these mixed media objects by a compositing script,
3) error resilience to permit robust transmission of compressed data over noisy communication links,
4) the ability to encode arbitrarily shaped video objects,
5) multiplexing and synchronization of the data associated with these objects so that they can be delivered over network channels providing a QoS appropriate to the nature of the specific objects,
6) the ability to interact with the audiovisual scene generated at the receiver.

Besides the third and fifth points, all other major functionalities are beyond the scope of this thesis and their technical details are given in [7]. Here, the emphasis is on constructing a robust cross-domain video streaming protocol that can efficiently deliver MPEG-4 data, particularly through noisy wireless channels, via a combination of open-loop error resiliency and closed-loop loss recovery tools. In the following section, a brief description of the MPEG-4 error resilience tools is presented.

4.1 Error Resiliency Tools and Packet Structure

MPEG-4 provides error resilience to support delivery of image or video data over a wide range of storage and transmission media [7]. To achieve error resilience, MPEG-4 devises schemes to facilitate early detection of corrupted data, and reduction and concealment of playback errors. Four error resilience tools, namely resynchronization markers, data partitioning, reversible variable length codes (RVLC), and header extension code (HEC), are employed by MPEG-4 to ensure error resilience within a video packet [8]. These tools have been used by many researchers working in the area of video data error resilience, and therefore are not unique to MPEG-4. The MPEG-4 video packet structure is shown in Figure 14.

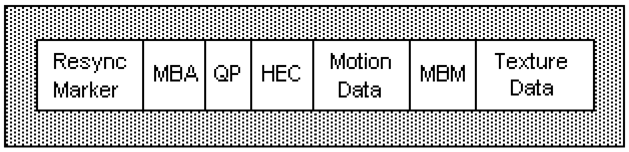


Figure 14: MPEG-4 video packet structure.

A single MPEG-4 stream is usually broken up into a series of video packets of roughly fixed lengths, with a resynchronization marker separating each packet. The resynchronization markers assist in regaining resynchronization at various locations in the bitstream in the event of channel errors. High-activity areas of the video stream possess a higher concentration of resynchronization markers since these markers are typically inserted into the MPEG-4 bitstream periodically every *K* bits.

In the presence of a short burst of errors, the decoder can quickly localize the error to within a few macroblocks in high-motion regions and preserve the corresponding image quality. To further enhance error resilience, MPEG-4 video packets are constructed to be independently decodable by removing all data dependencies between successive video packets within the same image. This is achieved by insertion of two additional fields, namely the macroblock address (MBA) and the quantization parameter (QP), after the resynchronization marker at the beginning of each video packet.

When error occurs, the header contains information that is necessary to restart the decoding process. However, any bit errors occur in the three fields of the header will result in the inevitable dropping of the entire packet. To reduce this likelihood, a 1-bit field of HEC is added to the video packet header. If this bit is set, the presence of additional resynchronization information within the packet is indicated. This redundant information is made available to protect against the case the packet header has been corrupted.

If an error in the bitstream is detected, resynchronization procedure enables the decoder to locate the error to be in the macroblocks between the two resynchronization markers. Normally, video decoders replace all the erroneous macroblocks by data from the corresponding macroblocks in the previous frame for error concealment. This is so because often the motion and texture data for each macroblock are coded together such that any bit error would corrupt the entire section. MPEG-4 reduces such occurrences by partitioning the motion and texture data by a unique motion boundary marker (MBM) such that errors in one section would not corrupt the other, and more video data can be salvaged through the computation of RVLC. Further explanation of these error resiliency tools can be found in [7] [8].

4.2 MPEG-4 Over Cross-Domain Connections

The MPEG-4 error resiliency tools provide additional protection against wireless channel errors, which is achieved at the expense of sacrificing extra processing and battery power at the mobile devices. Thus it may advantageous to couple the error resilient encoding scheme with a transmission protocol that applies appropriate loss recovery policies for video streaming in the wireless domain. On the other hand, the main purpose of applying MPEG-4 error resiliency tools over the wired domain is to compensate for packet loss due to network congestion. As discussed in the previous chapters, congestion control and relevant loss recovery strategies are needed to minimize deployment of MPEG-4 error resiliency tools in the wired network. As a result, extending the MPEG-4 video connection over a cross-domain environment requires a composite transmission protocol that is able to manage the unique combination of problems and challenges associated within each domain.

5 The Proposed Protocol

The main objective of the proposed protocol, named as Split-Domain TCP-Friendly Protocol (SDTFP), is to implement a TCP-based closed loop control mechanism (including dynamic source rate adjustment, congestion control and loss recovery), which minimizes the invocation of open loop control (MPEG-4 error control) at the mobile device and during end-to-end video transmission. The features of the proposed protocol include:

- *Split-domain approach*

  Because of the drastic condition disparity between the wired and wireless domains, a single end-to-end solution is intuitively very difficult. Therefore, a split protocol approach is used with the aid of a cross-domain proxy, similar to the one advocated by Wireless Application Protocol (WAP) [12], as shown in Figure 15. Although the split approach has its share of problems such as increased handoff complexity, broken end-to-end connectivity and security threats, it offers the ability to isolate the congestion and BER related complications which otherwise cannot be differentiated and resolved easily.

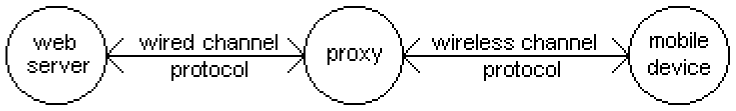


Figure 15: SDTFP system overview.

- *Window-based congestion control with proxy assistance*

  To manage congestion effects in the wired domain, SDTFP uses a window-based congestion control scheme to regulate video transmission rate in order to achieve TCP-friendliness. Unlike normal TCP additive-increase-multiplicative-decrease (AIMD) congestion window profile, SDTFP produces a smoother congestion window shape through active proxy participation to provide periodic proxy receiver window[1] adjustments. In other words, the SDTFP congestion control mechanism on the wired network is jointly managed by the web server and the intermediate proxy.

- *Large initial and loss windows in congestion control*

  To eliminate possible playback stalls due to the TCP slow start algorithm reset that leads to buffer underruns, the SDTFP congestion control initial and loss windows are set to be large enough to support the minimum playback rate allowed in MPEG-4 video streaming. As a result, any congestion window size would guarantee acceptable QoS at the video decoder while still being able to perform functional congestion control. In other words, the priority level of video traffic is inversely proportional to the instantaneous video encoding rate.

- *Adaptive rate video source encoding with respect to congestion window changes*

  To capitalize on the frequent fluctuations of the congestion window, the source video encoder dynamically adapts the source encoding such that a smaller congestion window entails a lower video streaming rate, and vice versa. By pegging the source encoding rate with congestion window changes, the probability of buffer underruns or overruns is greatly reduced. Although the existence of an adaptive rate video encoder is assumed true, discussions on the exact implementation details of this integral component is beyond the scope of this thesis and is subject to further research.

[1] Note the use of two receiver windows in SDTFP: 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 clarity, they are separately named throughout this thesis as proxy receiver window and mobile device receiver window.

- *Best effort loss recovery with selective acknowledgements*

  Traditional TCP offer 100% reliability on data reception that is sometimes not necessary for video streaming applications, especially with the help of MPEG-4 error resiliency tools. However, SDTFP is still required to provide a best effort loss recovery scheme in order to guarantee video QoS and minimize the deployment of data recovery tools at the mobile receiver. To further increase the efficiency of the loss recovery scheme, SACKs are used to deliver acknowledgement and retransmission information in both wired and wireless links.

In order to simplify the system model, the following assumptions are made:

- The current proposal on SDTFP primarily focuses on unicast connections as an initial step. Future proposals, if any, may include support for multicast connections.
- Both the intermediate proxy and mobile device have sufficient physical memory for buffering.
- Each SDTFP packet has the same header structure as a TCP/IP segment, and it encapsulates only one video packet to achieve simplicity in performance comparison during simulation.
- This study focuses only on the network transfer of video streams. Other aspects such as connection management and inter-cell handoffs are left for future investigations.
- Accurate measurements of RTT are readily available for calculating the correct dimensions of the congestion window, proxy receiver window and retransmission timeout period.

5.1 Source Initialization

For each video streaming session, the upper and lower bounds on 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 quarter common interchange format (QCIF) format at 176 x 144 pixels, 32 kbps should be adequate as being the barely acceptable minimum. The maximum playback rate is set as the highest possible video streaming rate, likely determined by mobile device parameters, wireless channel throughput and user preferences. Since the rate values are translated into congestion and proxy receiver window sizes via multiplying the RTT for each domain, so it is important to obtain accurate RTT measurements, possibly through the use of methods suggested in [13].

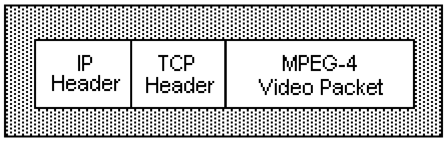


Figure 16: Encapsulation of MPEG-4 packets in TCP/IP header structure.

We start by first examining the behaviour of the connection between the web server and the proxy. After initial connection establishment, the MPEG-4 video encoder at the web server outputs a video stream at a playback rate that is dictated by the congestion window size. Each MPEG-4 video packet is encapsulated in a TCP/IP header structure, as shown in Figure 16, and branded with a timestamp and queued in the web server buffer for transmission. Since random bit errors within the MPEG-4 video packet can be recovered via the error resiliency tools, the checksum field of the TCP/IP header should be limited to provide cyclic redundancy check (CRC) protection only to the TCP/IP header in order to expedite the checksum process for each video packet at the mobile device. The video packet should remain at the source buffer until a positive acknowledgement from the corresponding SACK packet is received. If retransmissions are needed, the web server simply transmits the requested video packets again without changing the TCP/IP header timestamp.

5.2 Congestion Control Algorithm

Figure 17 illustrates the desired SDTFP congestion window profile. Initially, the proxy receiver window is set by the proxy to support the default maximum playback rate, for instance equivalent to 384 kbps, as indicated by line 1. Less aggressive approaches can lower the initial proxy receiver window value, but the congestion window will take longer to attain the maximum rate during steady-state operation. In accordance to TCP specifications, the original intent of the proxy receiver window should indicate the amount of buffer space available at the proxy such that the sender can regulate the transmission rate to achieve flow control [2]. As mentioned above, an abundance of physical memory is assumed to be allocated at the proxy, and the MPEG-4 video streaming rate is kept at a relatively low level for mobile applications. Since the probability of proxy buffer overflow is therefore reduced, the receiver window field in the TCP header can be reconfigured to assist in maintaining a constant video streaming rate.

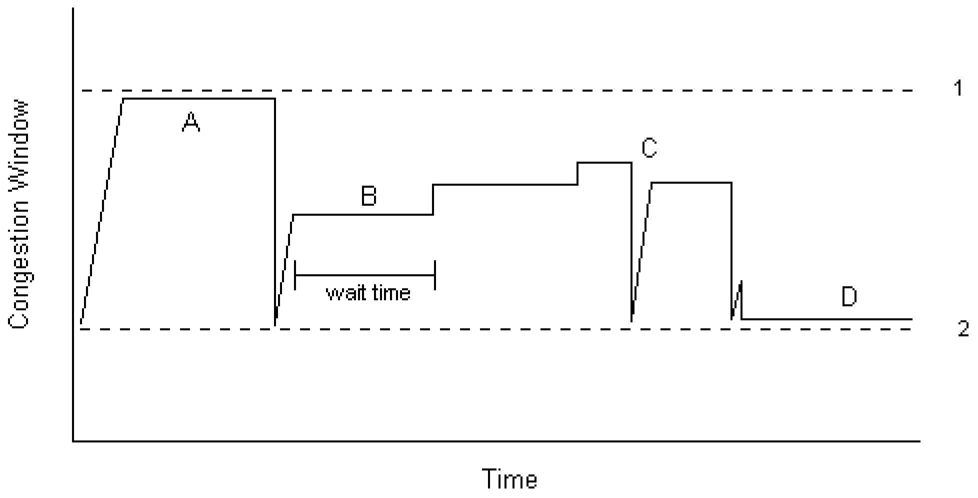


Figure 17: Desired SDTFP congestion window profile.

Line 2 in Figure 17 specifies the initial window/loss window that is small enough to support the minimum playback rate. However, the actual RTT for the video connection is likely not known at the initialization phase. Therefore a default value should be assigned to the initial window in the beginning. As the RTT estimation is refined after several packet transfers, the loss window value is adjusted accordingly to reflect the minimum playback rate. There are two advantages in setting the initial window/loss 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/loss window may exceed the initial window/loss window limit set by TCP [14], it avoids video stoppage instances when the slow start algorithm resets the congestion window to the loss window that leads to rebuffering delay. The second advantage is that with a large initial window/loss 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 enters the slow start phase and increases multiplicatively until it reaches the proxy receiver window size. Similar to the TCP slow start algorithm, successive SACK packets exponentially increase the web server's congestion window size. The video encoder at the web server dynamically references the congestion window to arrive at the optimal video encoding rate that can adapt to changing network conditions. On the other hand, unlike common TCP implementations, SDTFP does not implement the congestion avoidance phase in a continuous linearly sloped fashion. In other words, the threshold value of the slow start algorithm is pegged against the advertised proxy receiver window size such that the congestion window stops growing once it reaches the threshold. During video streaming, the congestion and proxy receiver windows should remain relatively constant for a stable period of time to avoid fluctuations that damage picture smoothness. In favourable 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 Figure 17.

When a timeout occurs due to increased network congestion, as indicated by region B, the video server first sends a timeout notification to the proxy along with the congestion window size set prior to the timeout event. The proxy should then decrease the proxy receiver window size accordingly to reduce bandwidth consumption and prevent further timeouts (to be explained in the next section). The web server congestion window then retracts to the loss window and goes through the slow start phase again. Because of the lack of slow start threshold, the congestion window is able to enjoy a rapid exponential recovery and bypass the slow congestion avoidance stage. The exponential growth phase stops when the congestion window size reaches the newly advertised proxy receiver window size.

Since the proxy receiver window size is less than the maximum allowable proxy receiver window size at region C, further increases of the window size are possible through the modified congestion avoidance algorithm that is governed by the proxy. After some wait time, say 1 minute, the proxy receiver window is increased slightly, perhaps by a single loss window, for more prudent congestion avoidance network probing. If no timeout events occur during the next wait time, the proxy receiver window is then increased yet by another loss window. This process continues until the proxy receiver window reaches the maximum proxy receiver window size, or experiences another timeout. In the latter case, the congestion window recovery process is once again repeated. This time around, however, the next wait time is increased by 2 times. 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 is then 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 periodic network probing for higher bandwidth without ending in rate oscillations as illustrated in Figure 18.

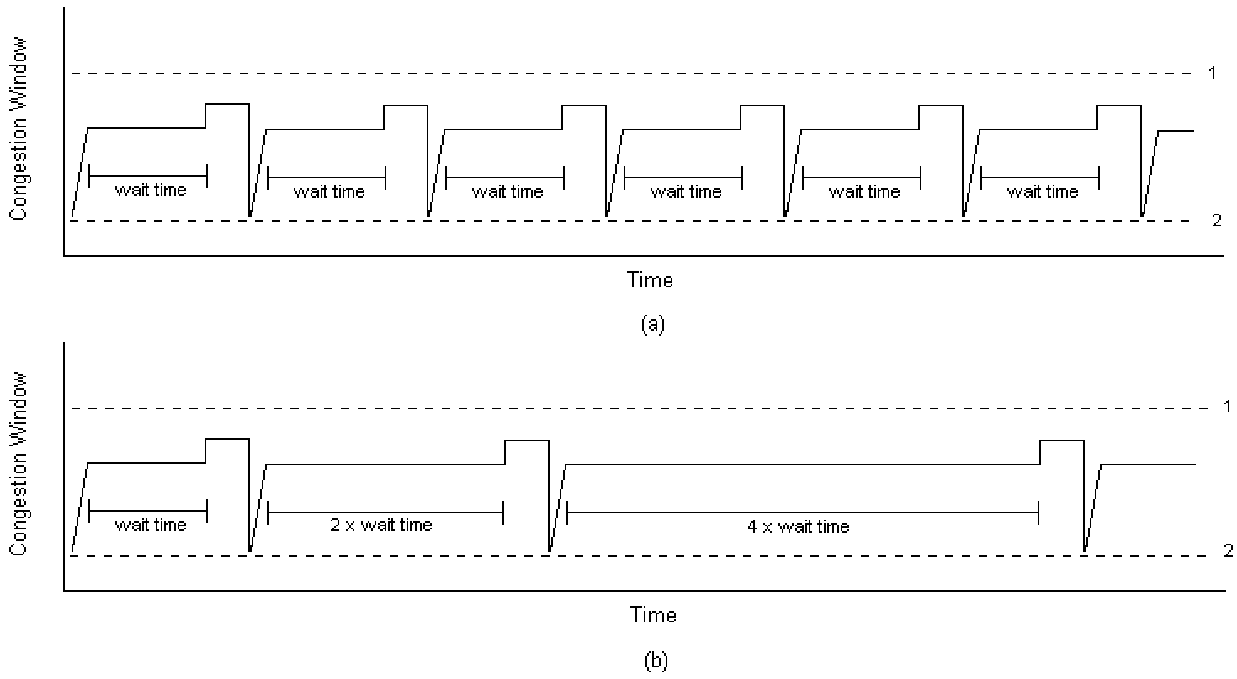


Figure 18: SDTFP congestion avoidance. (a) With rate oscillations. (b) Without rate oscillations.

If network conditions deteriorate substantially to a point such that excessive loss occurs even at the lowest transmission rate as indicated in region D of Figure 17, then video transmission is not viable at that instant. 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 does contradict the notion of TCP-friendliness, the inherent higher priority of video transmission should be precedent over other TCP connections to guarantee the minimal video QoS. If network deterioration persists, the connection can be terminated automatically when the detected packet loss rate exceeds a certain threshold when transmitting at the minimum rate, or manually by the understandably frustrated end user.

5.3 Acknowledgement Generation, Loss Recovery and Timeout Management

Selective acknowledgements (SACKs) in SDTFP are based on the TCP SACK option [15] and are used for acknowledging multiple packet reception. Given the higher transmission overhead required by SACK packets, the proxy should delay SACKs, which is to return a SACK after the reception of a predetermined number of packets, to cumulatively acknowledge these receptions. Since SACKs themselves offer excellent protection against multiple packet losses, TCP fast transmit and fast recovery algorithms are bypassed at the current deliberation for the SDTFP specification. Further research can be conducted to explore the viability of uniting these two TCP loss recovery schemes within SDTFP.

If multiple video packets are delayed or lost in the network, gaps would appear within the reception buffer, as shown previously in Figure 8(a). To the web server, these gaps within the SACK block sequencing space indicate the possible occurrence of increased network traffic, and retransmissions of the missing packets are necessary. Again, the construction of the SACK packet in relation to the receiver buffer profile is shown in Figure 8(b), in which the inter-block relationship is contained within the options area of a TCP packet. The left edge of a block is the first sequence number of this block, whereas the right edge of a block is the sequence number immediately following the last sequence number of this block. The segment that triggered the SACK is contained in the first block, unless that segment advanced the acknowledgment number field in the TCP/SDTFP header, to assure that SACK reflects the most recent change in the receiver buffer. Upon reception of such SACK packets, the web server should immediately retransmit the packets indicated by the gaps without waiting for the full timeout.

Because of the real-time nature of video traffic, some packet loss can be tolerated by bypassing the missing packet sequences. When the lost packets do not arrive in time at the proxy before they are due to be relayed to the wireless domain, the missing packets are 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. For instance in Figure 19, blocks of video packets sent from the web server are shown to be cached in the proxy buffer waiting to be forwarded to the wireless domain with gaps of lost packets intertwined within the buffer. The thick black line in the middle of the buffer indicates the virtual boundary between the wired and wireless domains in a sense that the packets on the right of the line have already departed for the mobile receiver. Therefore any future retransmitted packets arrived at the proxy that belong to gaps 3 and 4 will not be forwarded to the wireless device. The sequencing space in the SACKs should fully acknowledge the packets within gaps 3 and 4 to prevent these meaningless retransmissions and reduce bandwidth wastage.

As for gaps 1 and 2, the appropriateness of retransmitting those packets located within depends on the buffer consumption rate (i.e. the rate at which the packets leave the buffer and sent over the wireless link), remaining buffer length and wired domain RTT. For example, if the retransmitted packets of gap 2 arrive before the gap passes the thick black line, then the retransmission is rated as a success. In contrast, if the retransmitted packets are known to not be delivered in time beforehand, then it is better to alter the sequencing space of the corresponding SACKs to stop further such retransmissions.

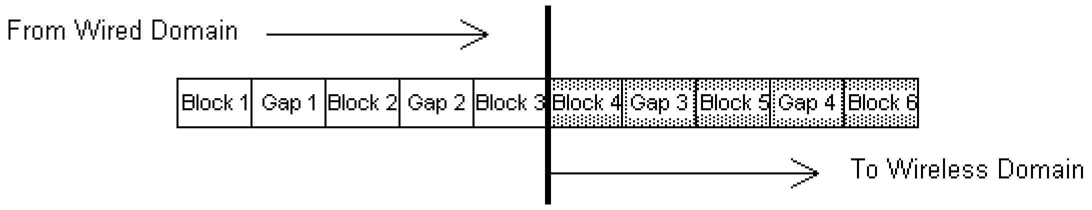


Figure 19: Best effort loss recovery at proxy buffer.

The retransmission mechanism is only suitable for odd lost packets. Persistent deterioration of network condition triggers significant packet loss and timeouts. In order 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 interpreting the changing dynamics of network traffic. The timeout interval should be set appropriately according to RTT estimates such that the web server can react to changing network conditions promptly. Similar to TCP, the duration of the retransmission timer can be computed dynamically by following the guidelines stated in [26].

Immediately following a timeout, the web server sends a timeout notification to the proxy that contains the congestion window size set prior to the congestion window size that presides this timeout event. The logic is that if the web server was able to increment the congestion window from its previous value through congestion avoidance and then a timeout occurs, it indicates that the immediate network conditions are able to support a video rate that is dictated by the previous congestion window but not the current one. Therefore by reverting the congestion window to its previous value, the video stream should be delivered without complaint from the network. In other words, the proxy should decrease the proxy receiver window to equal to the congestion window size specified in the timeout notification in order to reduce bandwidth consumption and prevent further loss events. Like TCP, web server reiterates the slow start phase after a timeout, and all unacknowledged packets are continually retransmitted until they are acknowledged by successive SACKs.

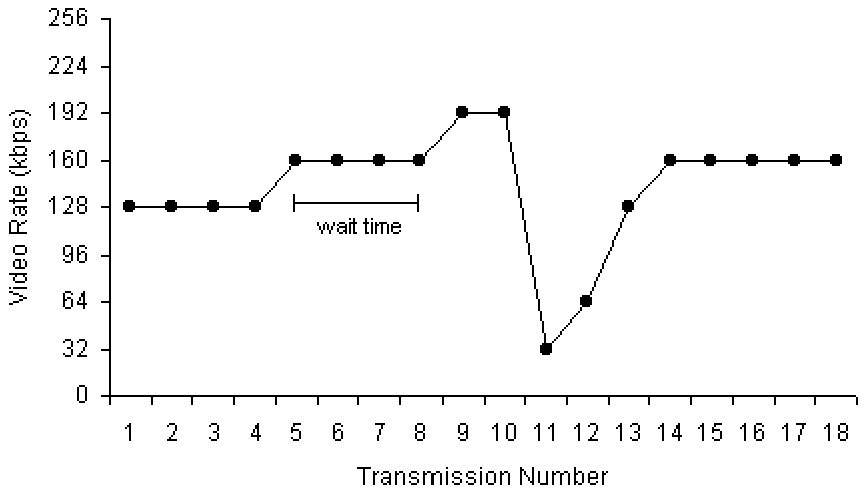


Figure 20: SDTFP steady state timeout recovery.

To put the timeout recovery mechanism into perspective, first consider Figure 20 where it shows how the video sending rate responds to a steady state timeout (i.e. a timeout that occurs during the modified congestion avoidance period). The first 4 transmissions run at 128 kbps and then the video rate is increased by 32 kbps to 160 kbps due to the modified congestion avoidance algorithm. After 4 more successful transmissions, the video rate is increased yet again to 192 kbps. However, the current network conditions cannot support the video stream at such a rate and therefore causes a timeout. The web server responds by sending a timeout notification to the proxy and it contains the congestion window size set prior to the congestion window that presides the current timeout event. In this case, it equals to the congestion window corresponding to a video rate of 160 kbps. Afterwards, the proxy receiver window is reduced accordingly and the web server goes through the slow start phase again, starting at 32 kbps. The exponential growth of the video rate will stop at 160 kbps, as restricted by the new proxy receiver window. As mentioned in the previous section, the current wait time is increased by 2 times, so the next congestion avoidance network probing will happen 8 RTTs later. The video rate will be increased if the network conditions improve then, or it will repeat the recovery process again if another timeout occurs.

In other instances, timeouts may occur in the slow start phase either during the video streaming initialization or the congestion avoidance recovery process. Figure 21(a) shows the exponential growth of video streaming rate from 32 kbps to 384 kbps without the occurrence of timeouts. This suggests the current network conditions are able to sustain the maximum video streaming rate at 384 kbps as bounded by the proxy receiver window. When the network condition is only able to support a lower video rate, as indicated in Figure 21(b), timeouts will occur if the source video streaming rate exceeds a certain value, and in this case it is at 256 kbps. This implies the current network conditions are most likely able to maintain a video rate between 128 kbps, where no timeouts have been recorded, and 256 kbps. To recovery the congestion window or video streaming rate, the web server sends a timeout notification to the proxy with the congestion window size corresponding to 128 kbps. Upon its arrival, the proxy will then reduce the proxy receiver window to match a video rate of 128 kbps. Therefore when the slow start phase repeats again at the web server, the video streaming rate will stop at 128 kbps to prevent further timeouts. The constant rate will be changed only after the wait time where the congestion avoidance network probing happens. Similarly in Figure 21(c), the network conditions cannot support any video rate higher than the minimum playback rate. Consequently, timeouts occurring at 64 kbps suppress the subsequent video rate at 32 kbps for the next wait time interval.

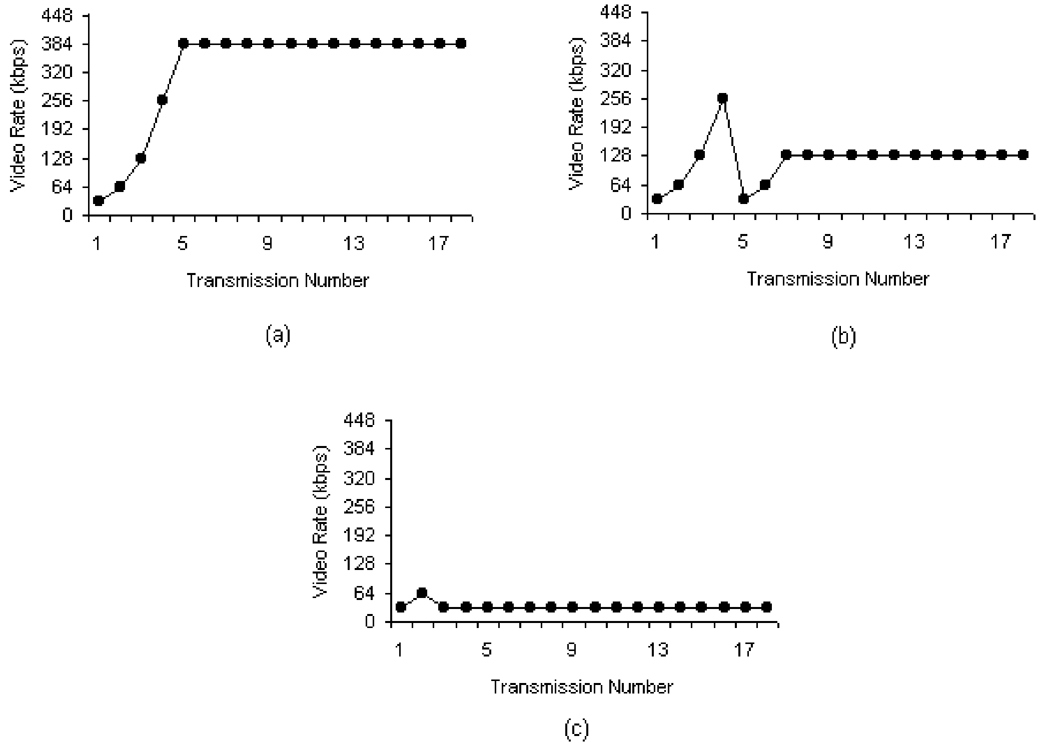


Figure 21: Timeout recovery during slow start. (a) No timeout. (b) Timeout at medium video rate. (c) Timeout at low video rate.

5.4 Proxy Processing

Typically, the intermediate proxy simply passes every packet received from one domain onto the other domain. While this ensures simplicity on proxy implementation, it puts the burden of congestion control and loss recovery mechanism on the receiver, which in our case is a processing power-limited mobile device. In the proposed system, the proxy is responsible for handling rigorous processing duties that span both wired and wireless domains, especially the congestion control algorithm of the wired link. In addition to managing two separate connections, the proxy should meticulously manage the interchange of packets between the two domains.

Packets received from the web server are first placed in the proxy buffer, waiting to be forwarded to the mobile device. During a steady state period, 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 occurs at the proxy. 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 for the proxy is to deliver video packets to the mobile device at a rate that is exactly the same as the server transmission rate.

As discussed in Section 3.1.3, 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 the frequency of retransmission to compensate for the fading dispersive channel while keeping the current video streaming rate constant. As congestion seems to be less of a factor in radio communications, the concepts of slow start and congestion avoidance that are prevalent in TCP are not particular applicable in such an environment. Also since the intermediate proxy relays the video packets according to a predetermined rate based on inter-arrival timestamps, this implies that there is no direct relationship between the congestion window dimensions and the eventual video streaming rate over the wireless link. Therefore the wireless video connection is valid as long as the video streaming rate does not create buffer overflow at the mobile device and is below the maximum data throughput rate supported by the wireless channel. According to this reasoning, the congestion window structure seems irrelevant over the wireless link, so the SDTFP approach is to eliminate all congestion control mechanisms for this section of the end-to-end connection.

Similarly, retransmission timers over the wireless link are deemed optional since the link itself is not regulated by a congestion window mechanism just as the proxy is continuously forwarding video packets from the wired domain, thus preventing the streaming connection to stall. Bypassing the retransmission timer option relieves some of the proxy processing load, while maintaining this option may offer marginal performance enhancements in packet loss recovery, especially during high BER periods. If implemented, the retransmission timers can be set upon transmissions of individual video packets, where the timeout interval should be set appropriately according to RTT estimates such that the proxy can react to changing channel conditions promptly. Similar to the SDTFP wired domain implementation, the duration of the retransmission timer on the wireless link can be computed dynamically by following the guidelines stated in [26] that offer a different result because of the discrepancy in RTT estimation of both domains. On the return channel, acknowledgement packets are often corrupted because of high probability BER. Therefore adjustments to the timer duration estimation may be necessary to compensate for this effect.

Again similar to the wired domain implementation, packet loss retransmissions are triggered by the gaps in sequencing space indicated by SACK packets. An extremely poor 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 often experiences periods of high bit error loss, the frequency of SACK generation should be increased accordingly to more expediently recover corrupted video packets and to minimize messaging errors due to SACK packet loss. The delay SACK estimation should also exploit the short RTT and simple network topology of the wireless link.

5.5 Mobile Device Processing

Normally, the mobile device allocates enough buffer space that is able to sustain video streams of any rate as defined during the initialization process. The amount of available buffer space should be advertised to the proxy via the mobile device receiver window field in SACKs. Apart from managing the reception of video packets over the wireless SDTFP connection, the mobile terminal caches all received packets that feed to the MPEG-4 decoder. Because of technical scope limitations, details about the implementation of the MPEG-4 decoder will not be discussed in this thesis and will be left for future research. Any erroneous receptions are recovered through the error resiliency tools, though the proposed transmission protocol should have minimized such algorithm invocations. Similar to the wired network approach, any missing packets at playback time is deemed as unrecoverable. The corresponding gap within the SACK packet sequencing is filled accordingly.

5.6 SDTFP Advantages, Disadvantages and Other Considerations

Analyzing solely from the proposed protocol implementation details without considering actual simulation performance results, tentative conclusions can be drawn on SDTFP regarding its advantages and disadvantages in realizing reliable cross-domain video streaming connections in comparison with other existing options and proposed candidates. These issues and other design considerations mentioned elsewhere in this chapter are collectively summarized in the following sections.

5.6.1 Advantages

The principal feature of SDTFP is its split architecture in spanning both wired and wireless domains to isolate the respective design issues with separate dedicated protocols. In the wired domain, the TCP-friendly congestion control algorithms and loss recovery schemes coupling with adaptive rate MPEG-4 video encoding assure video playback QoS by effectively dampen the crippling congestion effects on video streaming applications, while still are able to induce fairness in bandwidth sharing of network resources. By opting for an end-to-end approach in implementing wired domain congestion control for video applications, SDTFP avoids complicated and often expensive network upgrades as required for network-assisted schemes, given the enormous capital and effort already invested in today's Internet infrastructure. The

SDTFP TCP-based congestion control mechanism combines sender-based slow start algorithm with receiver-based congestion avoidance scheme to provide smooth video picture quality as well as fair bandwidth sharing with prudent incremental network probing simultaneously. In recovering from loss events, large initial and loss windows in congestion control expedite congestion window growth to limit QoS degradation.

The wireless portion of SDTFP, on the other hand, combats channel errors with adaptive transmission policies and loss recovery techniques to complement open loop error resiliency of MPEG-4 encoding standard. By walling off congestion effects of the wired domain, SDTFP allows dedicated techniques for video delivery on the wireless link. For instance, the elimination of slow start algorithm in the wireless domain enables the video transmission rate to be sustained during high BER periods, as opposed to being reverted back to the minimum rate in the case of selecting TCP as the wireless protocol.

For loss recovery, SDTFP adopts the best effort delivery strategy that is effective in removing redundant retransmissions, especially when buffering space is not abundant at the proxy or the mobile receiver. When packet loss happens, SDTFP guarantees the receiver with some degree of packet loss recovery, depending on receiver buffer length, without sacrificing streaming video fluidity or incurring unnecessary retransmissions. In this aspect, SDTFP triumphs over both UDP, which does not provide any loss recovery mechanisms, and TCP, which decreases overall streaming video efficiency in allowing redundant retransmissions.

Apart from the split architecture and best effort delivery strategy, the design of SDTFP incorporates a lot of methodologies imported from TCP. Other than the web server, the intermediate proxy and the mobile device, SDTFP require no additional processing involvement or upgrade from other TCP network nodes such as routers, switches and bridges for an end-to-end video streaming connection since all of the SDTFP features and options are embedded within the TCP header. Therefore the migration of video applications to the new protocol should be less tremulous than to other protocols that are based on drastically different principles. Also, the resonating breadth and depth attained by researchers regarding TCP over the years should greatly assist further development of SDTFP. Perhaps the only aspect that requires special attention is the intermediate proxy design in facilitating the interchange of cross-domain traffic. Given the opportunity, this issue will be further investigated in later research projects.

5.6.2 Disadvantages

In incorporating within SDTFP the list of design attributes that yield the above system advantages, some level of protocol performance trade-off must be expected and tolerated. For example, the SDTFP split architecture emphasizes on the need for a robust intermediate proxy that should encounter as few critical failures as possible. Unlike a network router breakdown where data traffic can be re-routed to bypass the blockage, crashes of the intermediate node completely terminate all mobile connections and render the situation unrecoverable. Given the heavy load of proxy processing envisioned by SDTFP, the expected frequency of such fatal crashes may be increased. Therefore stringent software reliability standards must be enforced in constructing resilient proxy executables.

Also, since the proposed method relies heavily on a precise RTT estimation, it is important to use the most accurate mathematical model to calculate RTT. Correct dimensions of the congestion window, timeout period, and proxy receiver window all depend on the accuracy of RTT. However, accurate measurement of RTT may be difficult both in theory and in implementation [13]. Often, RTT is determined through the use of the TCP Timestamp option that relies on cumulative empirical estimates of RTT. In comparison, it is more involved to obtain accurate RTT estimation for the wired domain because of the shear number of possible inter-network routes, high variance of router processing delay and the constant occurrence of network congestion.

During the initial phase of transmission, the SDTFP proxy receiver window is determined by the product of maximum playback rate and RTT, and the initial/loss window is calculated by multiplying the minimum playback rate and RTT. Both values may not be accurate because of the lack of cumulative data for RTT estimation, and so the initial playback quality may fluctuate as a result of constant adjustments of RTT estimation. This effect will be smoothed out as historical data for RTT estimation accumulates. In order to have a consistent interpretation of RTT, both the sender and receiver should periodically exchange network information such that discrepancies in RTT estimation between the two endpoints are minimized.

As demonstrated in Section 5.2, SDTFP exhibits UDP-like bandwidth hogging at the minimum playback rate. Though this feature slightly deviates from the TCP doctrine on congestion control and bandwidth sharing, the inherent higher priority of video transmission should be precedent over other TCP connections to guarantee the minimal video QoS. Furthermore, the low value of minimum playback rate should pose minimal impact on other network connections unless an unforeseen number of SDTFP connections are running concurrently.

Finally, transmission protocols that implement acknowledgements typically have intrinsic protection mechanisms, particularly through packet timeouts, in recovering from loss ACKs. While SDTFP is no exception, it faces additional vulnerability in the potential loss of timeout notification such that the proxy is not informed of the most recent web server timeout. This results in no reduction in proxy receiver window and the web server will continue to assume the previous proxy receiver window size as the bandwidth maximum that likely leads to another timeout. The incorrect proxy receiver window advertisement should be rectified by the next SACK packet or timeout event.

5.6.3 Other Considerations

The following issues are general impediments in developing cross-domain video streaming applications that are not unique to SDTFP. This thesis does not intend to address these design considerations in detail, nor provide thorough examples or solutions on how SDTFP may be implemented in these situations. It is up to later research projects to come up with further suggestions on this matter.

- *Handoff*

  Although the user mobility issue is largely solved by Mobile IP as mentioned in Section 2.2, there exists the problem of buffer content migration for SDTFP video connections. Assuming the intermediate proxies are located within the base stations, then when a mobile device moves outside the coverage of a cell with the channel handed off to a new base station, the video packets stored inside proxy buffers at the previous base station will have to routed to the new base station. This causes extra processing overhead that may result in temporary video playback stoppages. Fortunately, 3G systems (in particular CDMA-based networks) employ soft handoffs [48] that allow the mobile device at the cell coverage boundary to maintain contact with both base stations such that the previous base station will have extra leeway in flushing out buffered video packets while any new arrivals will be directed to the new destination. In addition, if the wireless WAN is configured to have a single intermediate proxy administrating several geographically adjacent base stations, then the probability of a full buffer migration during inter-cell handoff will be further reduced.

- *Video seeking*

  While real-time video broadcasts must synchronize with the server, other video streaming applications may allow video seeking that dynamically alters the current starting point of video playback. Whenever the video seeking function is invoked during a streaming session, all the buffered video packets at both the proxy and the mobile receiver must be discarded and be replaced with newer video packets that pertain to the new starting point of playback. This process will incur extra buffering delay that is dependant on the predetermined buffer length. Again, there exists a trade-off of having a larger buffer to offset packet loss, and the need to have swift response to video seeking with a smaller buffer.

- *Security*

  A split-domain connection such as a SDTFP video streaming session can suffer gateway attacks at the base station junction where communication between a mobile handset and an existing web server travels through. Gateway attacks or an unscrupulous system insider attempting to steal original data can gravely compromise network security. One of the ways to circumvent this problem is to establish a separate end-to-end connection between the web server and the mobile device purely for control and security information exchanges. Therefore the proxy will have a dual role of both actively regulating cross-domain video packets and passively forwarding end-to-end sensitive data. Another solution is to employ other security mechanisms above the transport layer, like TLS [46] or SOCKS [47], for end-to-end security.

- *Multicast*

  Multicast applications are useful particularly in real-time broadcasting, but designing a good multicast congestion control protocol is much more difficult since it has to scale to large receiver sets and be able to cope with heterogeneous network conditions at all the receivers. As mentioned before in the design assumptions for SDTFP, multicast support is not considered at this moment.

6 Simulation Strategy and Results Analysis

To ideally evaluate the effectiveness of the proposed SDTFP scheme, the performance of video transmission over the SDTFP should be compared with the throughput of the same video stream delivered over UDP, TCP, and other proposed TCP-friendly protocols respectively within a heterogeneous cross-domain network environment. In all cases, the video components such as commercial MPEG-4 streaming codecs and playback software should be used and the wired and wireless domains should be placed under identical sets of channel and congestion parameters. Performance analysis should based on subjective tests such as visual comparison of input/output video signals (picture crispness and smoothness, buffering delays), and raw numerical results (error recovery statistics, retransmission rates, actual transmission throughput) for contrasting sets of input attributes. After thorough analysis, the optimal sets of design parameters can be determined under different application scenarios.

In reality, because of technical limitations, manpower constraints and other adverse factors, extensive verification of the SDTFP system model cannot be undertaken within the short time frame of this master research program. As a result, several structural concessions have to be made in order to simplify the simulation process while still is able to produce meaningful results, as listed below:

- *Forgo adaptive rate MPEG-4 video encoder/decoder components*

  The success of the SDTFP in some way hinges on the ability of the MPEG-4 video encoder to adapt to the dynamic changes of the congestion window. However, the construction of such an encoder involves much more in-depth research and therefore cannot be completed with the currently available resources, likewise for the decoder. As replacements for these two integral system components, a software program will run at the web server that produces a continuous stream of equally sized packets to mimic the MPEG-4 video packet stream, and another software program will emulate the video decoder at the mobile receiver to log the relevant statistics with respect to the arrival of the data packet stream.

- *Emulate network conditions artificially*

  The main purpose of SDTFP is to allow cross-domain transmission of video streams to withstand or mitigate the negative effects of congestion in the wired network and transmission errors in the wireless channel. Rather than verifying the theoretical results directly with real-world unsettling Internet traffic, a controlled network environment is needed for maintaining consistent input parameters and testing conditions in order to facilitate the simulation process. In this way, more tangible simulation results can be readily obtained without expending undue research efforts and resources.

- *Compare SDTFP with separate end-to-end UDP and TCP connections only*

  As mentioned in previous chapters, there exist a number of TCP-friendly video transmission protocols as proposed by the research community with only a handful that can apply in cross-domain connections well. The simulation process can be further simplified by reducing the comparison subjects to include only end-to-end UDP and TCP connections. In other words, SDTFP is intended to only demonstrate its validity in claming TCP-friendliness and predicted performance superiority over both end-to-end UDP and TCP at the current research stage. Later projects can involve broad comparisons of SDTFP with other similar video transmission protocols under a variety of probable network conditions.

6.1 System Setup

The simulation system model for SDTFP consists of five entities, shown in Figure 22, constructed using the C programming language. Each of the system components, i.e. sender, wired network emulator, proxy, wireless network emulator, and 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 SDTFP and UDP/RTP video streams, the interconnecting protocol of choice is UDP, while TCP is intuitively used for end-to-end TCP video sessions. During a simulation run, test data streams originate from the sender, passing through the wired network emulator onto the proxy. Then the proxy delivers the packets via the wireless network emulator to the receiver. The following sections explore the implementation details about each of the system components.

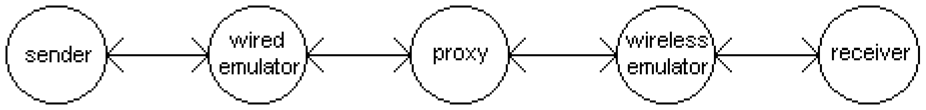


Figure 22: Simulation system model configuration.

6.1.1 Sender

The SDTFP sender is mainly responsible for handling data generation, congestion control, loss recovery, and timeout management on the wired domain. When simulation commences, the sender emits a flow of equally sized packets, each with its own unique sequence number and timestamp with simulated MPEG-4 video data, directed towards the wired network emulator for a specific length of time. On the return channel, the sender processes SACKs relayed by the wired network emulator and performs relevant congestion window adjustments and loss recovery procedures. The sender should record the sequence numbers and timestamps of departing packets, reception of incoming SACKs, timeout instances, and congestion window changes throughout the entire simulation process.

The TCP sender manages the similar duties as the SDTFP sender, but extended to cover the whole end-to-end connection. Amongst the discussed data recovery schemes in Section 3.1.2, the TCP sender will only implement SACK in order to achieve a better performance comparison with SDTFP simulated results. In case of UDP connections, the sender only provides data generation at a fixed constant rate without catering to packet losses.

6.1.2 Wired Network Emulator

The wired network emulator attempts to reproduce wired network characteristics such as congestion and RTT in a controlled manner. To simulate a network delay, each incoming packet is delayed at the emulator for a random amount of time with a predetermined mean RTT value. Likewise, heavy congestion packet loss is simulated by selectively dropping packets by the emulator with respect to a preset network traffic profile for the duration of the simulated video stream. Separate network traffic profiles are constructed and tuned to distinctly portray probable congestion patterns in individual test cases (refer to Section 6.3). For data gathering, the emulator should record the nature and relevant information of all incoming packets from both downlink and uplink directions, and sequence numbers of intentionally dropped packets for the test duration.

6.1.3 Proxy

The SDTFP proxy takes on the laborious roles described in Section 5.4 to regulate cross-domain simulated video traffic. Because of the busy cross-domain interactions and transactions, the proxy should log a larger set of data that include the sequence numbers and timestamps of incoming and departing packets, generations and receptions of SACKs, receptions of timeout notifications, and proxy receiver window changes in the wired domain. In simulating TCP and UDP/RTP connections, however, the proxy realizes a much simpler role in relaying packets from one domain onto another without any intermediate buffering or processing.

6.1.4 Wireless Network Emulator

Similar to its wired counterpart, the wireless network emulator introduces wireless characteristics such as RTT and BER to traversing data packets. While imitating RTT behaviour can be easily accomplished by the emulator, replicating channel errors artificially involve precise bit manipulation of the data packets that may complicate the simulation process. A simpler solution prescribes that each packet is attached with a flag that indicates packet corruption and it is set randomly at the emulator using a 2-state Markov model to be further explained in Section 6.2.5. Therefore, bit corruptions can be instantly interpreted by the receiver without going through intensive CRC, FEC or HEC computations. Again, the emulator should record the nature and relevant information of all incoming packets from both ways, and sequence numbers of corrupted packets for the whole test duration.

6.1.5 Receiver

The SDTFP receiver simply handles the task of the mobile device as indicated in Section 5.4. The receiver log should record the information about incoming data packets and generation of SACKs that would eventually determine the overall transmission throughput and efficiency. End-to-end UDP/RTP connections are even simpler for the receiver to manage, as no return acknowledgements are required. The TCP receiver functions in a similar fashion as the SDTFP receiver with tasks such as buffering of incoming data packets and returning SACKs.

6.2 System Variables and Parameter Derivations

After outlining the individual components within the simulation model in the previous section, a set of systems variables are needed to be defined in order to carry forth the simulation process in a controlled network environment, as discussed in the following sections. In parameter derivations, the use of the asterisk differentiates terms that are defined exclusively for the wireless domain.

6.2.1 Maximum and Minimum MPEG-4 Video Streaming Rates

As defined in Section 5.1, the minimum playback rate is the lowest video streaming rate at which the minimum video QoS requirements are guaranteed, whereas the maximum playback rate is the highest possible video streaming rate, likely determined by mobile device parameters, channel conditions and user preferences. Given a mobile device at QCIF (176 x 144 pixels) format, 32 kbps should be adequate as being the barely acceptable minimum. At the other end of the spectrum, 3G networks permit a maximum rate of 384 kbps for outdoor connections. To facilitate for simpler simulations, the range of video streaming rates are reduced such that the upper and lower bounds are set to be 256 kbps and 64 kbps respectively.

6.2.2 Wired and Wireless Network RTTs

Table 4 lists the estimated RTT values based on geographical distances and expected processing delays for a number of connection types [49]. Although this does not give accurate measurements of actual RTT, it demonstrates the possibility of assembling simulation scenarios from a combination of RTT values. For simulation simplification, a constant RTT is assumed for both wired and wireless domains in a given test case such that the network conditions are reasonably portrayed. This value is known globally by all participating nodes at the start of transmission and remains unchanged through the video session.

Table 4: Wired and wireless RTT estimates.

Connection Type	Example	Approximate Distance	Estimated RTT
Wireless WAN	Base station to mobile device	3 km	2 ms
Inter-city	Toronto to Vancouver	4 500 km	125 ms
Cross-continental	Toronto to Vancouver	4 500 km	250 ms
Inter-continental	Vancouver to Hong Kong	12 000 km	500 ms

6.2.3 MPEG-4 Packet Size and Header Length

Referring to the MPEG-4 packet structure as mentioned in Section 4.1, assume the length of a MPEG-4 video packet to be *L*, the length of the header portion, which includes resynchronization marker, MBA, QP and HEC, to be *A*, and the length of the data portion (motion and texture data) and to be *B*. The average header portion length is

$$\overline{A} = \frac{L\overline{A_{MB}}}{\overline{A_{MB}} + \overline{B_{MB}}} = LH \tag{1}$$

where $\overline{A_{MB}}$ is the average number of bits per macroblock in the header portion, and $\overline{B_{MB}}$ is the average number of bits per macroblock in the data portion. Therefore, *H* corresponds to the proportion of a video packet that is occupied by the header information. If *V* is the proportion of a video packet assigned to the data portion, then the average data portion length is

$$\overline{B} = \frac{L\overline{B_{MB}}}{\overline{A_{MB}} + \overline{B_{MB}}} = LV \qquad (2)$$

Given a relatively low and narrow range of video streaming rates along with various values of RTTs for both wired and wireless domains, an arbitrary value of 1 kB (8 kb) is selected as the size of each video packet. Since the relationship between *A* and *B* are largely dependent on the MPEG-4 encoder attributes, a value of 0.25 is assigned to *H* that is to be constant for all test scenarios. This selected value is typical for MPEG-4 bitstreams encoded for low bit rate wireless applications [42]. Equations (1) (2) can be later used to derive the packet loss probability P_l^* due to channel error probability of P_e^* over the wireless link in Section 6.2.5.

6.2.4 Wired Network Traffic Profile and Packet Loss Rate

To demonstrate the performance of SDTFP, TCP and UDP/RTP in video streaming applications, test cases will be constructed with various system parameters along with a wired network congestion profile, shown in Figure 23, that represent different realistic network scenarios. They are:

- Maximum video streaming rate (0 to 90 seconds)
- Gradually degrading network conditions (90 to 180 seconds)
- Intermediate video streaming rate sustained by the network (180 to 270 seconds)
- Gradually improving network conditions (270 to 360 seconds)
- Drastic reduction of video streaming rate to the minimum value (360 to 400 seconds)

During simulations, the packet loss rate in the wired domain largely depends on the difference between the instantaneous video transmission rate allowed by the network that is specified by the test scenario during initialization, and the current video transmission rate dictated by the sender congestion window. If the latter figure is larger than the former value, then the excessive traffic will be deemed lost by the wired network emulator. If the reverse is true, then intuitively no packet loss should be recorded, though odd packet drops may occur in order to intentionally test the loss recovery mechanisms. To simplify the performance analysis, the return channel for SACK packets is assumed to be free of congestion effects.

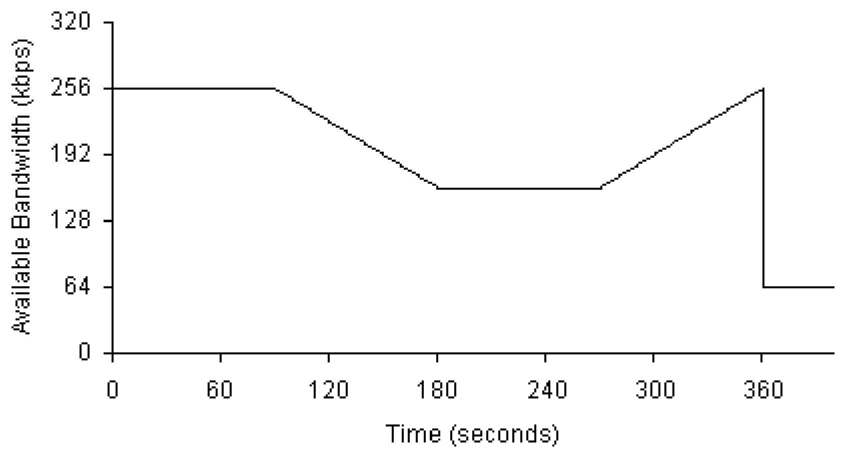


Figure 23: Wired network congestion profile for simulations.

6.2.5 Wireless Network BER and Packet Loss Rate

Since packet loss in the wireless domain is instigated by channel errors rather than congestion, intuitively the BER values must be specified prior to computing for packet loss rate of the wireless link. Previous studies have shown that a first-order Markov chain, such as a two-state Markov model illustrated in Figure 24, is adequate in producing a good approximation in modeling the error process at the packet level in fading channels [10]. In this model, the channel alternates between a "good state" and a "bad state", S_0 and S_1, respectively, with transitional probability matrix

$$P = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}. \tag{3}$$

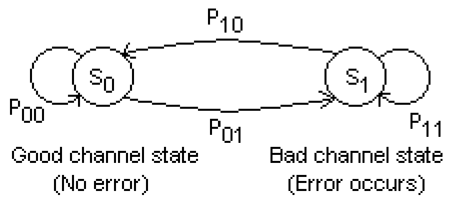


Figure 24: Two-state Markov channel model.

The two-state Markov model can be extended to a N-state Markov model, shown in Figure 25, to incorporate additional channel states defined as S_n, n = 0,…, N-1 in which S_0 indicates no bit errors due to channel fading, and all other states represent fading channel conditions. When the channel is in state S_n, $n \in$ {0,…, N-2}, the transition of the channel state is either to the next higher state or back to state S_0 based on the status of the currently received data. If the channel is in state S_{N-1}, it will always return to state S_0. Therefore it is only possible to generate burst errors of at most length N-1 with this model.

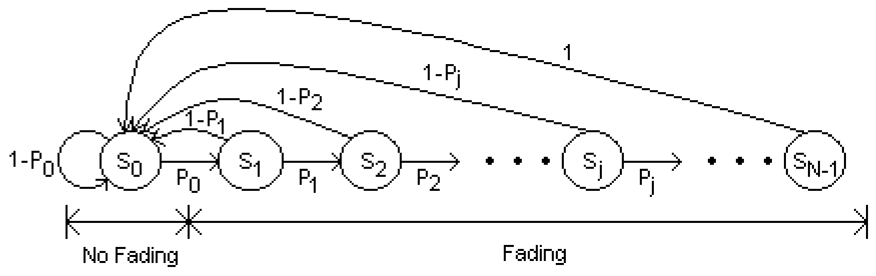


Figure 25: N-state Markov channel model.

As noted in Section 4.1, any bit errors occurring in the MPEG-4 header will result in the inevitable dropping of the entire packet. In addition, bit errors detected within the SDTFP/TCP/UDP/IP encapsulating header also cause the packet to be discarded. Therefore 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 less relevant as long as the probability of any bit error occurring in the packet header is known. So for the current simulations, the simpler two-state Markov model is selected to calculate the packet loss rate.

First, assume the first header bit of every video packet begins at state S_0. Given the wireless channel error probability P_e^*, the transitional probability matrix at state S_0 can be written as

$$P = \left[1 - P_e^* \quad P_e^*\right], \tag{4}$$

where P_e^* is selected to be in the range from 10^{-4} to 10^{-6} and be kept constant in each test case.

The probability of bit n being the first corrupted bit within a video segment in the wireless channel is

$$P_n = (1 - P_e^*)^n P_e^* . \tag{5}$$

For the protocol headers, all SDTFP, TCP and UDP/RTP headers occupy 20 bytes of packet space, with the IP header itself covering another 20 bytes of data. So the effective error probability for the protocol header is

$$P_{eff\,1} = \sum_{i=0}^{319} (1 - P_e^*)^i P_e^* . \tag{6}$$

The summation in Equation (6) can be simplified to

$$P_{eff\,1} = 1-(1-P_e^*)^{320}\,. \tag{7}$$

As defined in Section 6.2.3, the length of am MPEG-4 packet is set at 1 kilobyte with a header ratio of 0.25, which equals to 256 bytes. The effective error probability for the MPEG-4 video packet header given no errors occur in the TCP/IP header can be expressed as

$$P_{eff\,2} = (1-P_e^*)^{320}(1-(1-P_e^*)^{2048})\,. \tag{8}$$

Combining Equations (7) and (8), the overall packet loss probability due to random bit errors in the wireless forward channel is

$$\begin{aligned} P_l^* &= P_{eff\,1} + P_{eff\,2} \\ &= 1-(1-P_e^*)^{320} + (1-P_e^*)^{320}(1-(1-P_e^*)^{2048}) \\ &= 1-(1-P_e^*)^{320}(1-P_e^*)^{2048} \\ &= 1-(1-P_e^*)^{2368}\,. \end{aligned} \tag{9}$$

Since random bit errors are also assumed to be prevalent on the reverse wireless channel, the possibility of SACK packet loss for TCP and SDTFP connections must also be put into consideration. The SACK packet loss rate is calculated in a similar manner as Equation (7) with the inclusion of 64 bits of additional SACK data to the TCP/IP/SDTFP header. The uplink and downlink packet loss rates are computed are shown in Table 5.

Table 5: Wireless Channel Packet Loss Rates.

P_e^*	Downlink P_l^*	Uplink P_l^*
10^{-4}	0.210860	0.037674
10^{-5}	0.023402	0.003833
10^{-6}	0.002365	0.000384

6.2.6 Proxy and Mobile Device Buffer Length

During system initialization, abundant physical memory is allocated at the proxy and mobile device to buffer incoming video packets. The choice of buffer length is often dependent on application preferences as there exist performance trade-offs as discussed earlier in this thesis. For the simulations, a buffer length of 5 seconds for both the mobile device will be applied against various network conditions for each of the transmission protocols under scrutiny. This translates to having to provide a buffer size with upper bounds of 320 kB to 1.28 MB, which should be sustainable in a functional mobile device. In addition, a 5-second proxy buffer is allocated for SDTFP test cases, whereas TCP and UDP/RTP proxies will simply forward inter-domain packets without buffering, assuming the data throughput rate supported on the wireless link is greater than the maximum video streaming rate.

As an aside, given a proxy buffer length (in seconds) of T_{proxy} and wired network RTT of T_{RTT}, the maximum number of retransmission attempts for each video packet on the wired network is approximated as

$$N_{\mathrm{Re}Tx} \approx \left\lfloor \frac{T_{proxy}}{T_{RTT}} \right\rfloor. \tag{10}$$

Similar to the wireless domain, the maximum number of retransmissions allowed for a video packet given a mobile device buffer length (in seconds) of T_{mobile} and wireless link RTT of T^{*}_{RTT} is approximately given by

$$N^{*}_{\mathrm{Re}Tx} \approx \left\lfloor \frac{T_{mobile}}{T^{*}_{RTT}} \right\rfloor. \tag{11}$$

6.2.7 SDTFP Window Dimensions and Congestion Avoidance Parameters

Since the minimum video playback rate is defined to be 64 kbps in Section 6.2.1, the initial and loss windows for SDTFP are correspondingly calibrated in proportion to 64 kbps. During the congestion avoidance phase, it is at the proxy's discretion to select its degree of aggressiveness in probing for network resources. In this simulation, a more prudent approach is preferred such that each probing step equals one loss window and the minimum wait time is set at 15 seconds.

6.2.8 TCP Window Dimensions

In establishing TCP connections, it is important to first determine the values of TCP initial and loss windows that are often equal to the maximum segment size in use for the path. This in turn is computed by using the path maximum transfer unit (MTU) discovery algorithm [43] that involves using Internet Control Message Protocol (ICMP) messages [44] to garner path MTU information dynamically from network routers. As this algorithm obviously cannot be implemented within the simulated environment, both initial and loss windows are therefore preset to be 1 kB in order to simplify the simulation process. This also implies that the practice of increasing TCP initial window to enhance transmission performance [45] will not be considered in this set of simulation runs.

6.2.9 SDTFP and TCP Loss Recovery Parameters

Since both SDTFP and TCP use SACK and retransmission timers as part of their loss recovery schemes, sharing the same simulation parameters helps in arriving at better performance comparisons. Typically, the length of the retransmission timer is calculated via the most recent estimate of RTT. Because the RTTs are assumed to remain constant throughout each test session, all retransmission timers should be arbitrarily set to be around 2 times RTT for each domain.

As for the delay SACK frequency, the maximum allowed number of video packets collectively acknowledged by a single SACK packet is determined by

$$N_{SACK} = \left\lfloor \frac{R_{\min} * RTT}{L} \right\rfloor, \tag{12}$$

where R_{min}, RTT and L represent the minimum MPEG-4 video streaming rate, wired network RTT and the length of a video packet respectively for a particular simulation session. Since R_{min} and L are already defined to be 64 kbps and 8 kb, Equation (12) can be simplified to

$$N_{SACK} = \lfloor 8 * RTT \rfloor. \tag{13}$$

If the delay SACK frequency exceeds N_{SACK}, there exists the possibility of stalling the video connection when the streaming rate is running at the default minimum rate. During simulations, each test case can select a delay SACK frequency value that is less than or equal to N_{SACK} in order to demonstrate the relationship between delay SACK frequency, transmission throughput and packet loss recovery efficiency. For example, a wired network RTT of 125 ms produces a N_{SACK} value of 1. It implies that each SACK packet can acknowledge at most 1 video packet, thus effectively discarding the SACK option. On the other hand, N_{SACK} equals 4 when the RTT is set at 500 ms. Therefore, a delay SACK frequency value ranging anywhere from 1 to 4 can be assigned in such a circumstance.

Unlike TCP, SDTFP incorporates a separate SACK scheme for packet recovery in the wireless link as explained in Section 5.4. In order to reduce transmission overhead and exploit the short

RTT between base station and mobile device, a default value of 4 packets per SACK is used in the SDTFP simulations. In addition, these SDTFP simulations elect not to implement retransmission timers in order to lighten the processing load of the proxy.

6.2.10 UDP/RTP Transmission Rate

The transmission rate for UDP/RTP connections is determined during initialization and is kept constant throughout each simulation session. For each test case, three separate UDP/RTP sessions are transmitted at 256 kbps (maximum), 128 kbps (medium) and 64 kbps (minimum) to demonstrate the alleged UDP/RTP shortcomings in network bandwidth adaptation.

6.2.11 Summary of System Parameters

Table 6 lists the testing attributes specific to each protocol, whereas Table 7 records the important global system parameters defined above that are to be incorporated in the various test cases.

Table 6: Specific protocol parameters for simulations.

Parameter Name	**SDTFP**	**TCP**	**UDP/RTP**
Proxy Buffer Length	*5 seconds*	*0 seconds*	
Initial Window	*64 kbps * RTT*	*3 * 1 kB*	*N/A*
Loss Window	*64 kbps * RTT*	*1 kB*	*N/A*
Retransmission Timer Length	*2 * RTT*		*N/A*
Delay SACK Frequency (Wired)	*1 to 4 packets per SACK*		*N/A*
Delay SACK Frequency (Wireless)	*4 packets per SACK*	*N/A*	
Transmission Rate	*varies*		*64, 128, 256 kbps*
Probing Step	*64 kbps * RTT*	*N/A*	
Minimum Wait Time	*15 seconds*	*N/A*	

Table 7: Global system parameters for simulations.

Parameter Name	**Value(s)**
Video Streaming Rate Range	*64 to 256 kbps*
Wired Network RTT	*125, 250, 500 ms*
Wireless Network RTT	*2 ms*
MPEG-4 Packet Size	*1024 bytes*
MPEG-4 Header Ratio	*0.25*
Wireless Network BER	10^{-4} *to* 10^{-6}
Mobile Device Buffer Length	*5 seconds*

6.3 Implementation Notes and Simulation Results

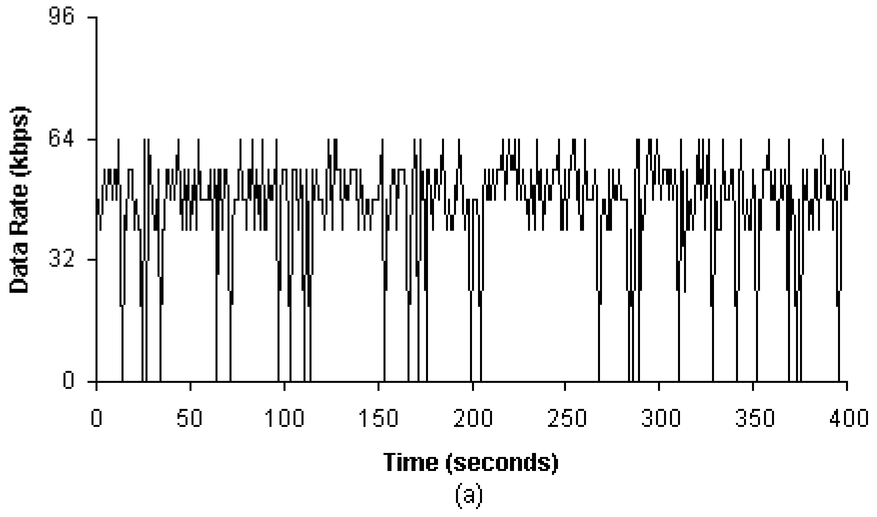
The simulation runs were conducted in the Broadband Communications Research Laboratory in July 2002. Each of the 5 network components ran on separate SUN Ultra 60 terminals with 512 RAM, all interconnected in a 100 Mbps LAN configuration. Typically, a simulation run involves transmitting imitated video data from the sender continuously for 400 seconds whilst exposed to the effects of a set of selected system parameters for that particular test case. As identified in the previous section, the major system variables for the simulation model are wired network RTT, delay SACK frequency, and wireless link BER.

6.3.1 UDP/RTP Test Results

In total, 9 UDP/RTP test cases are assembled with different combinations of data rate and wireless channel BER values as shown in Table 8. Since UDP/RTP connections contain no acknowledgement structure, simulating all test cases with a single wired network RTT value of 250 ms should be adequate to reveal meaningful results. The raw data produced by each test case is arranged in graphs, shown from Figure 26 to Figure 34, to depict two performance criteria: video packet throughput detected at the receiver and the overall packet loss rate (expressed as a percentage) in the end-to-end cross-domain connection.

Table 8: UDP/RTP 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}



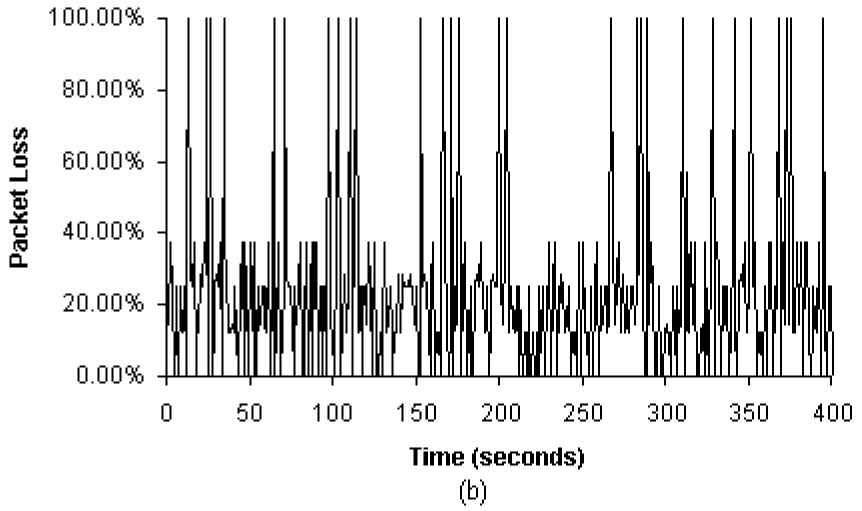
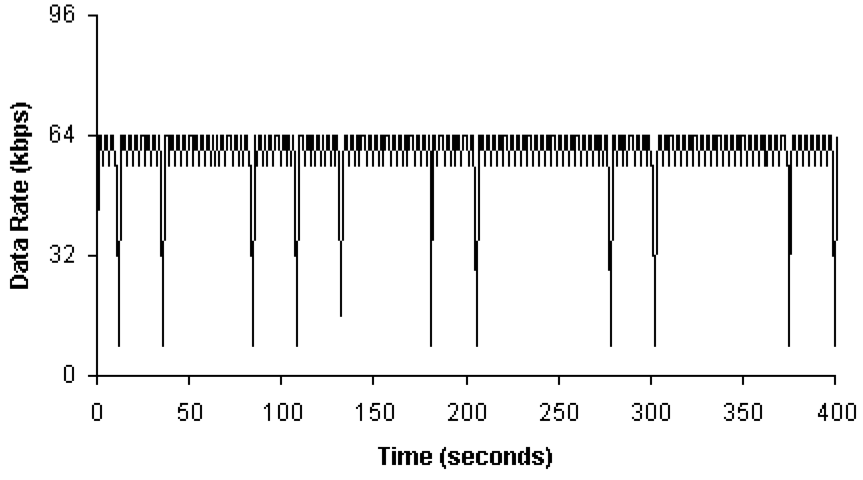
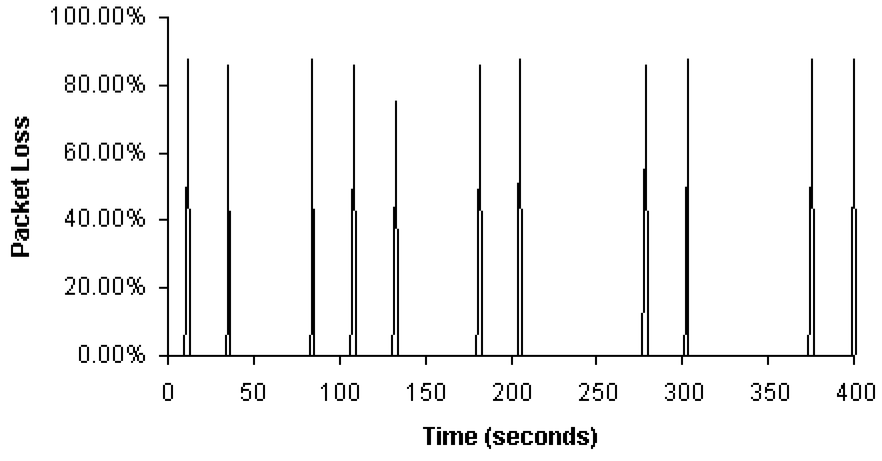


Figure 26: UDP/RTP test case 1 results. (a) Throughput. (b) Overall packet loss rate.

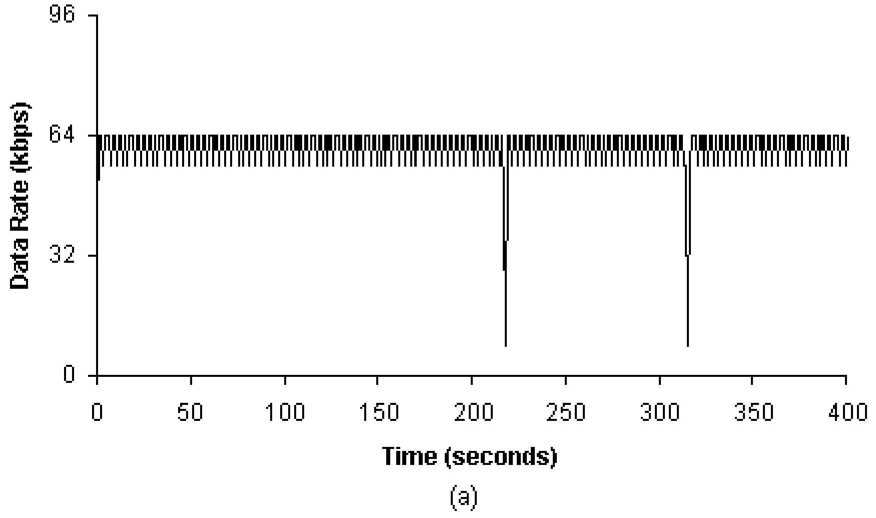


(a)



(b)

Figure 27: UDP/RTP test case 2 results. (a) Throughput. (b) Overall packet loss rate.



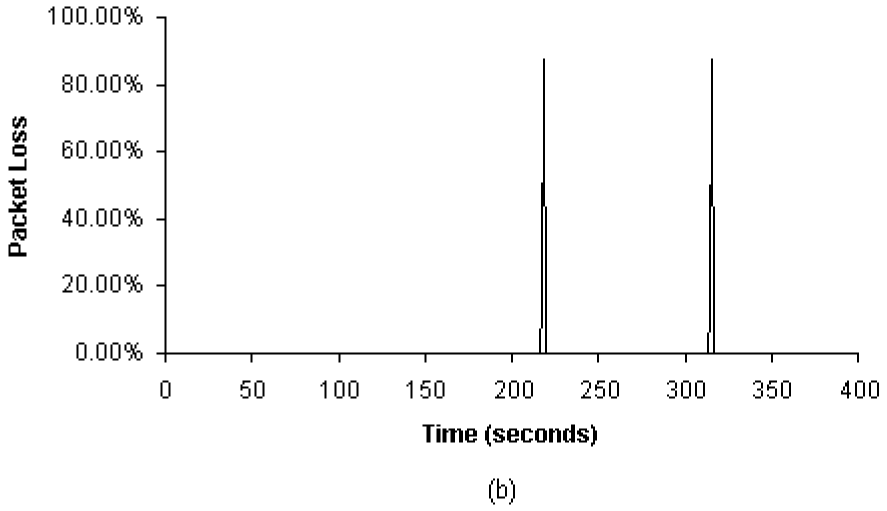


Figure 28: UDP/RTP test case 3 results. (a) Throughput. (b) Overall packet loss rate.

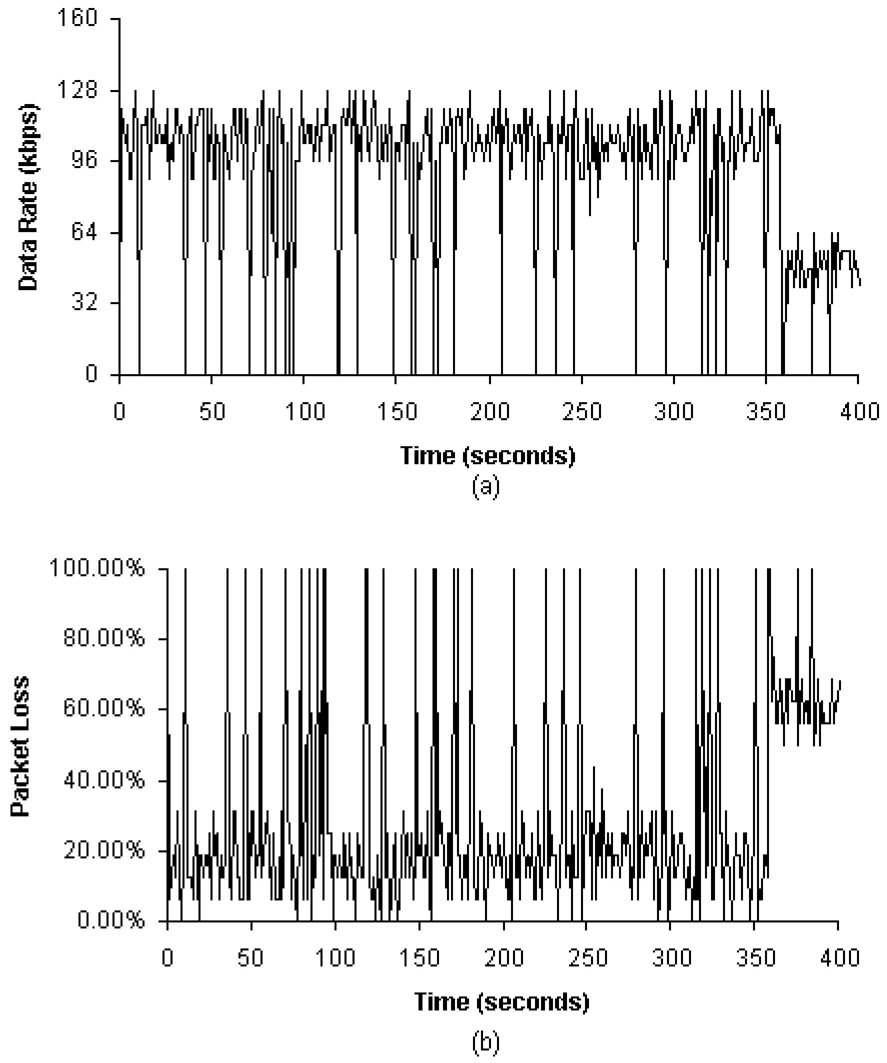
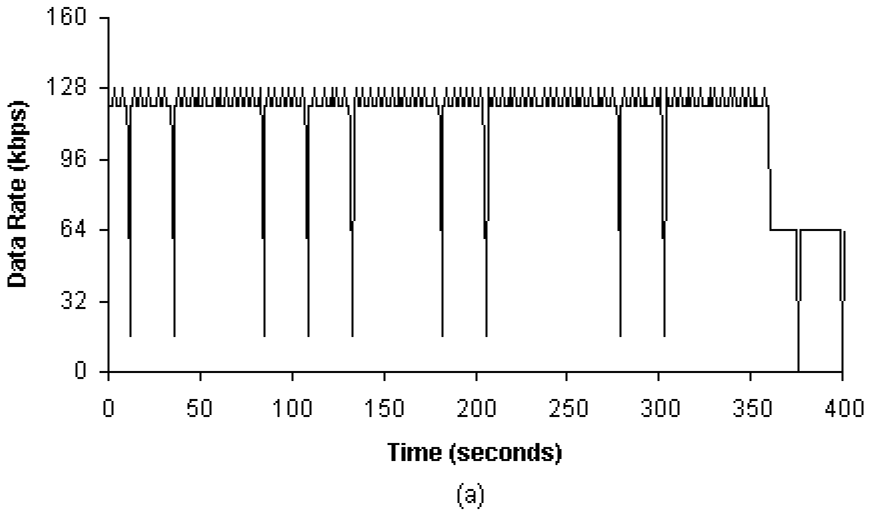


Figure 29: UDP/RTP test case 4 results. (a) Throughput. (b) Overall packet loss rate.



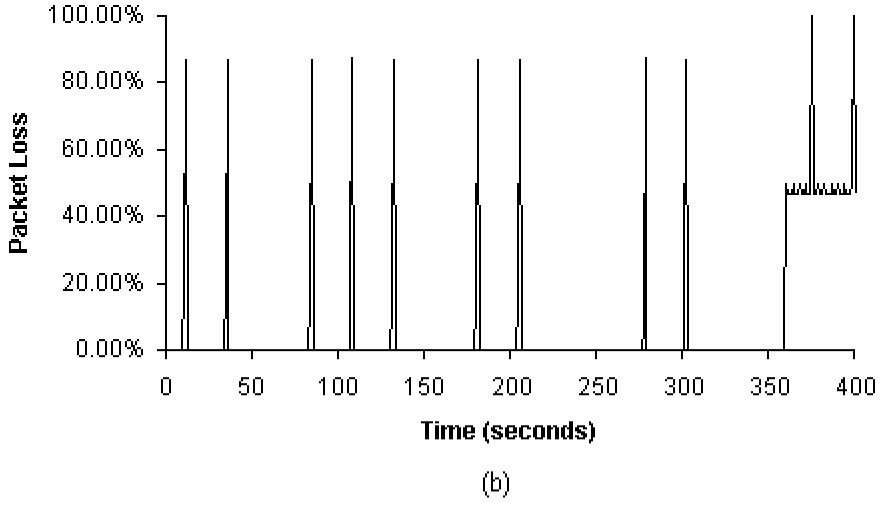
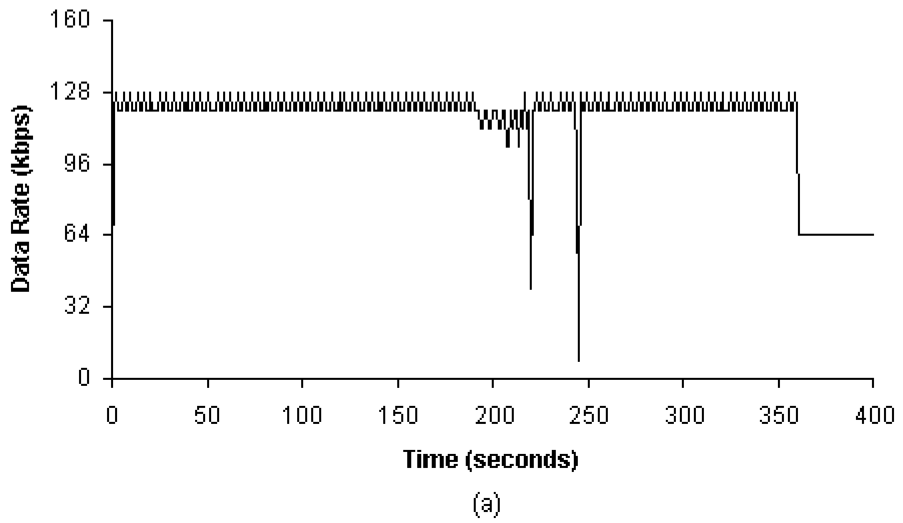


Figure 30: UDP/RTP test case 5 results. (a) Throughput. (b) Overall packet loss rate.



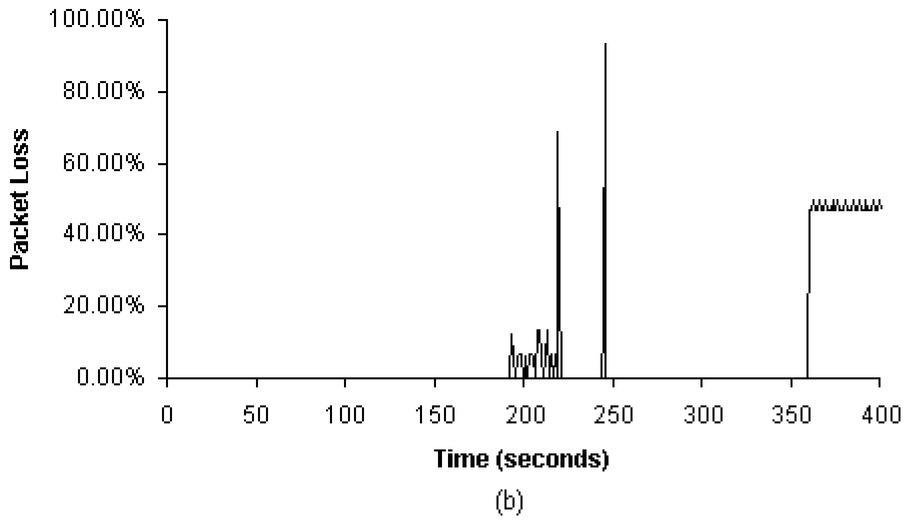
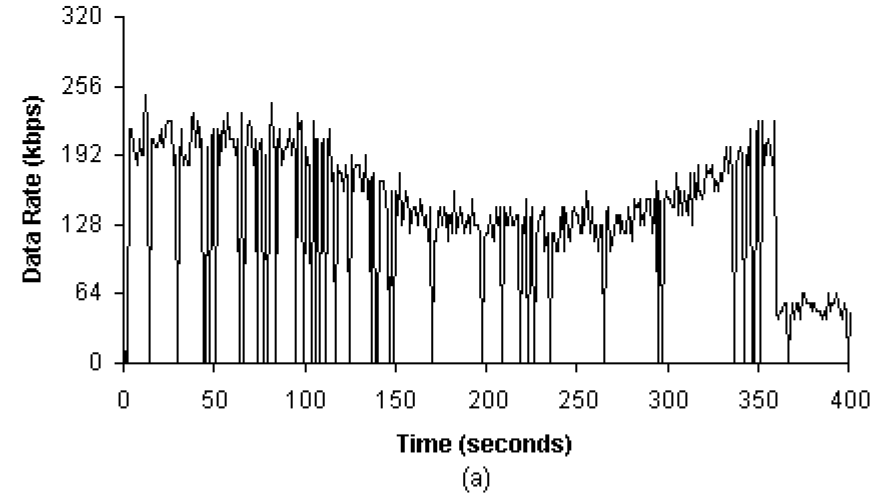


Figure 31: UDP/RTP test case 6 results. (a) Throughput. (b) Overall packet loss rate.



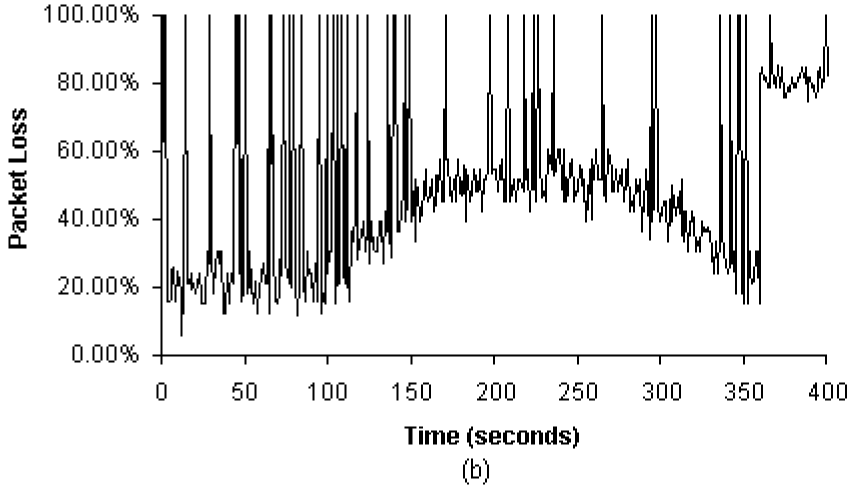
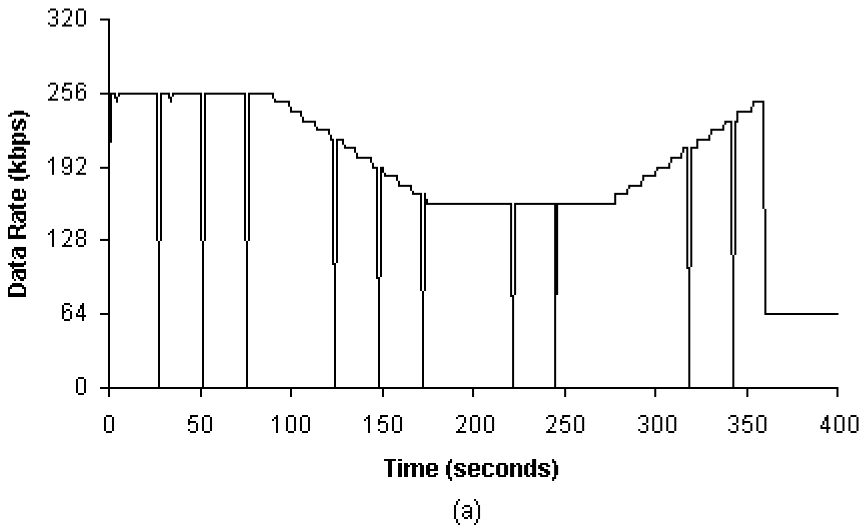


Figure 32: UDP/RTP test case 7 results. (a) Throughput. (b) Overall packet loss rate.



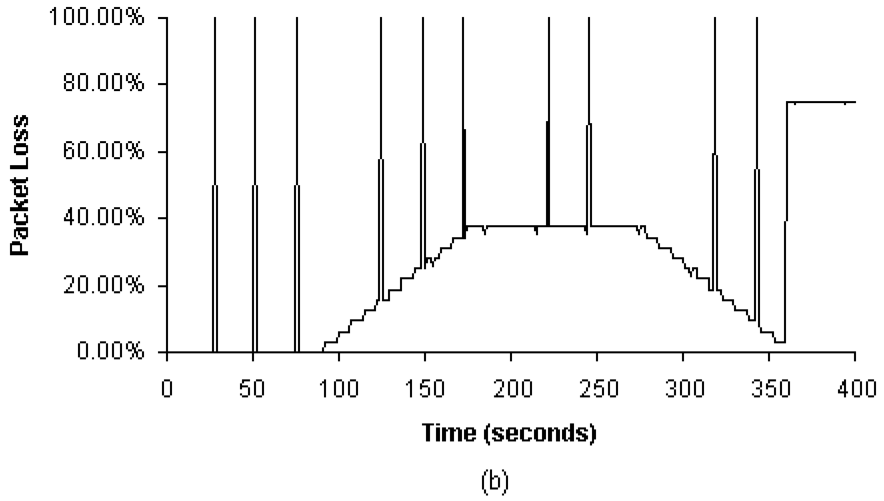
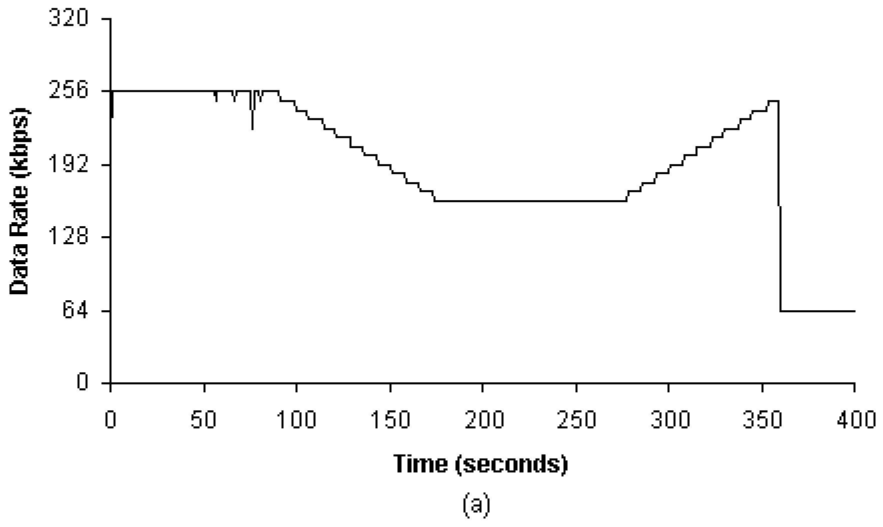


Figure 33: UDP/RTP test case 8 results. (a) Throughput. (b) Overall packet loss rate.



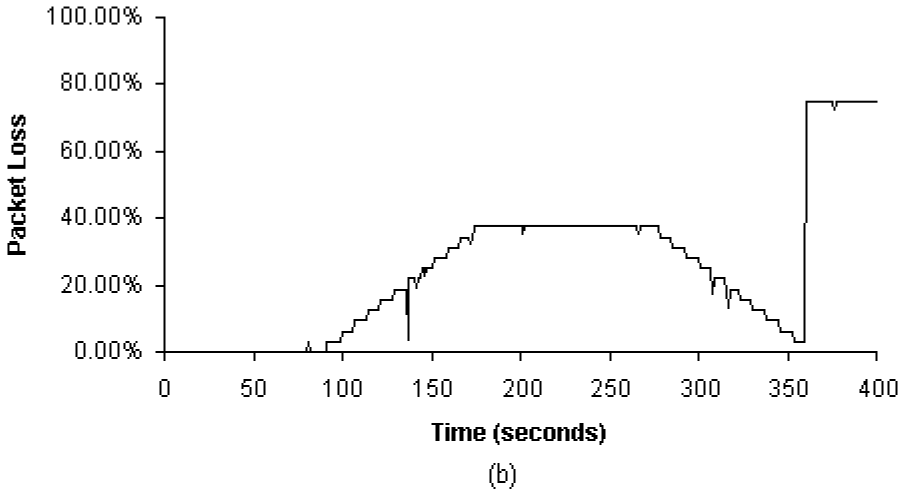


Figure 34: UDP/RTP test case 9 results. (a) Throughput. (b) Overall packet loss rate.

6.3.2 TCP Test Results

Similar to UDP/RTP simulations, a total of 9 test cases are constructed as listed in Table 9. 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 TCP connection. As for the collected simulation data, a third performance criterion, namely congestion window changes, is graphically presented from Figure 35 to Figure 42 along with the two performance criteria already utilized in UDP/RTP simulations.

Table 9: TCP test cases.

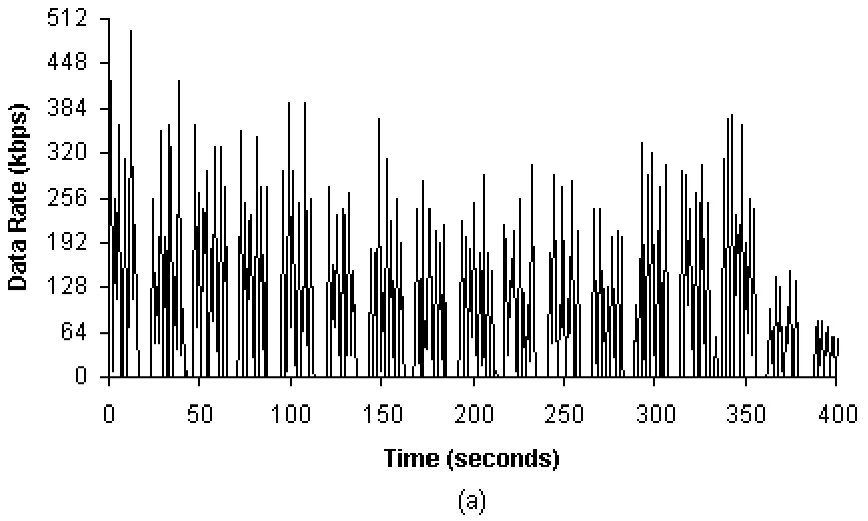
Test Case	Packets per SACK	Wired Network RTT (ms)	Wireless Channel BER
1	1	125	10^{-4}
2	1	125	10^{-5}
3	1	125	10^{-6}
4	2	250	10^{-4}
5	2	250	10^{-5}
6	2	250	10^{-6}
7	2	500	10^{-4}
8	2	500	10^{-5}
9	2	500	10^{-6}

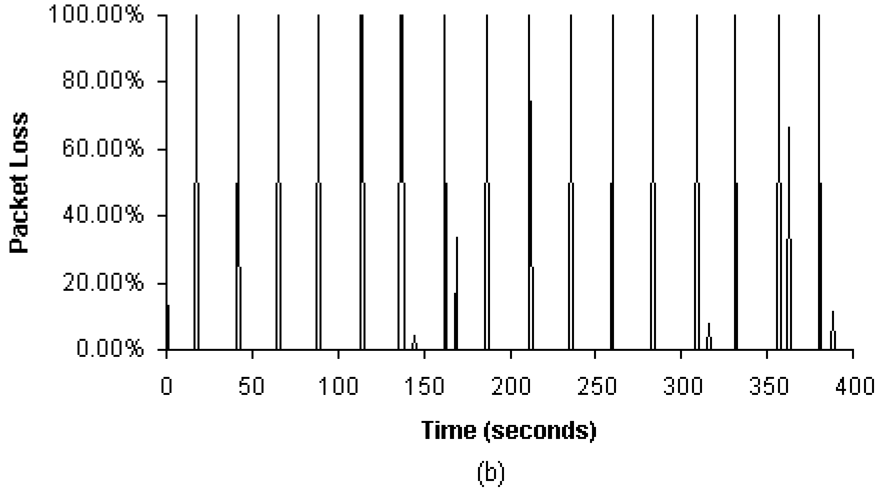
6.3.3 SDTFP Test Results

Table 10 shows the 12 test cases created for SDTFP simulations, which include the testing of the attribute of 4 packets per SACK. Similar to TCP simulations, the resultant data is organized graphically from Figure 44 to Figure 55 against the three aforementioned performance criteria.

Table 10: SDTFP test cases.

Test Case	Packets per SACK	Wired Network 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}





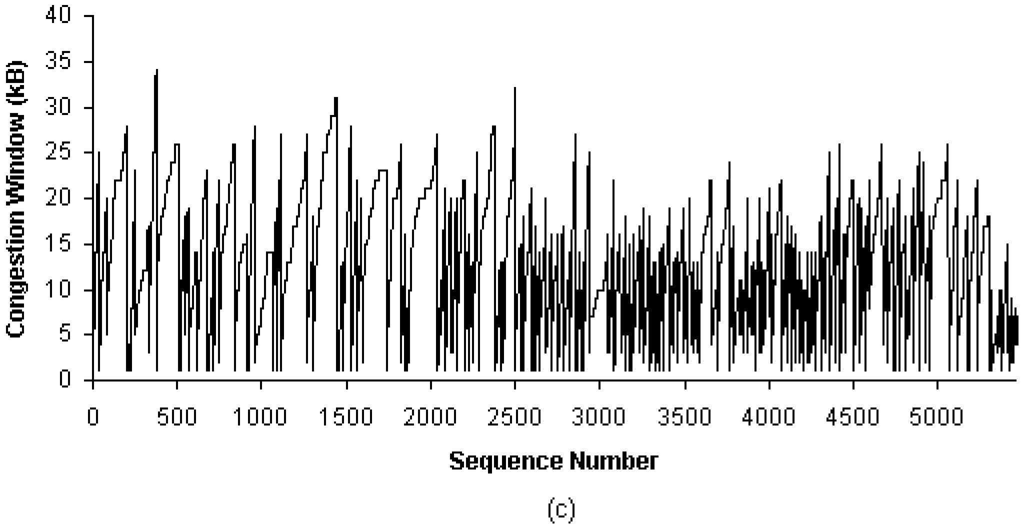


Figure 35: TCP test case 1 results. (a) Throughput. (b) Overall packet loss rate. (c) Congestion window changes.

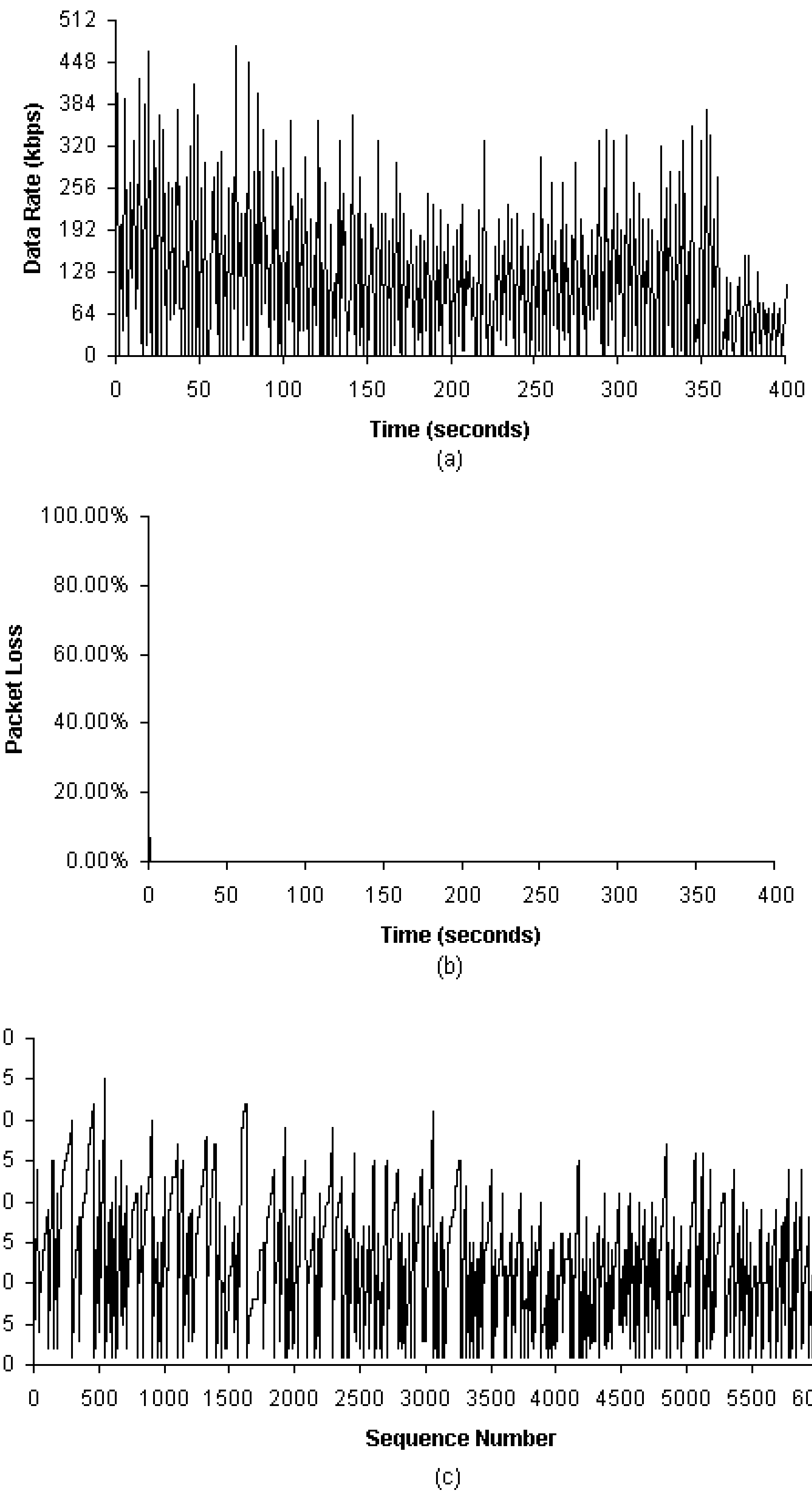
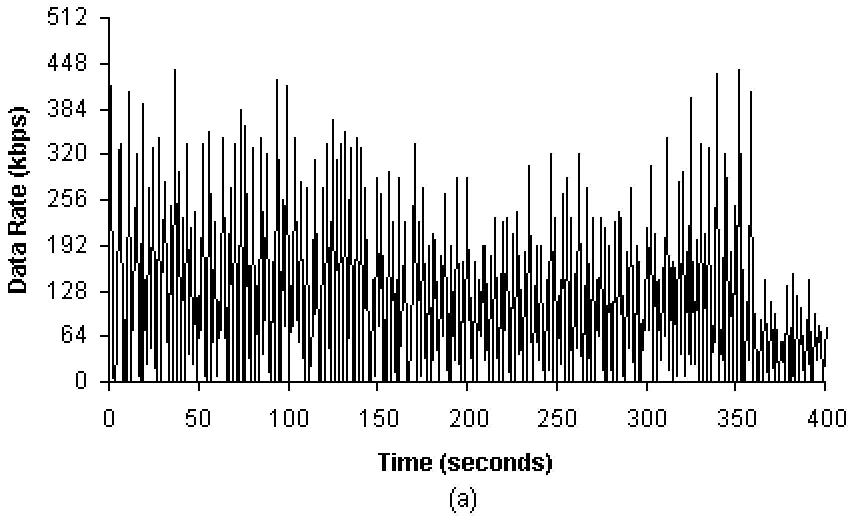
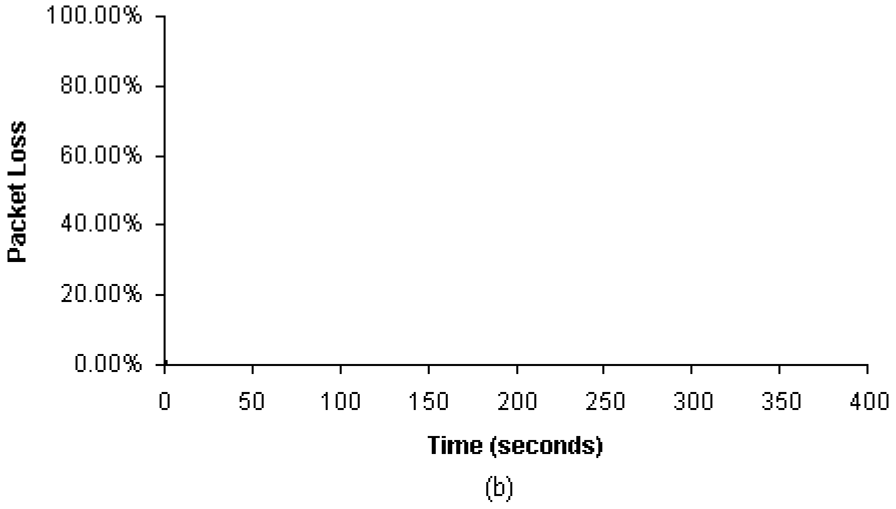


Figure 36: TCP test case 2 results. (a) Throughput. (b) Overall packet loss rate. (c) Congestion window changes.





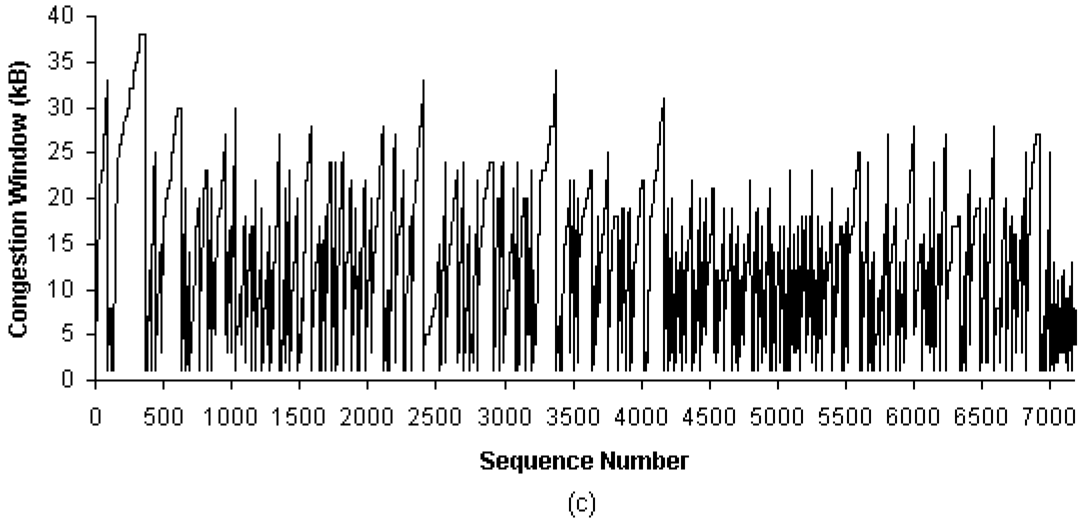


Figure 37: TCP test case 3 results. (a) Throughput. (b) Overall packet loss rate. (c) Congestion window changes.

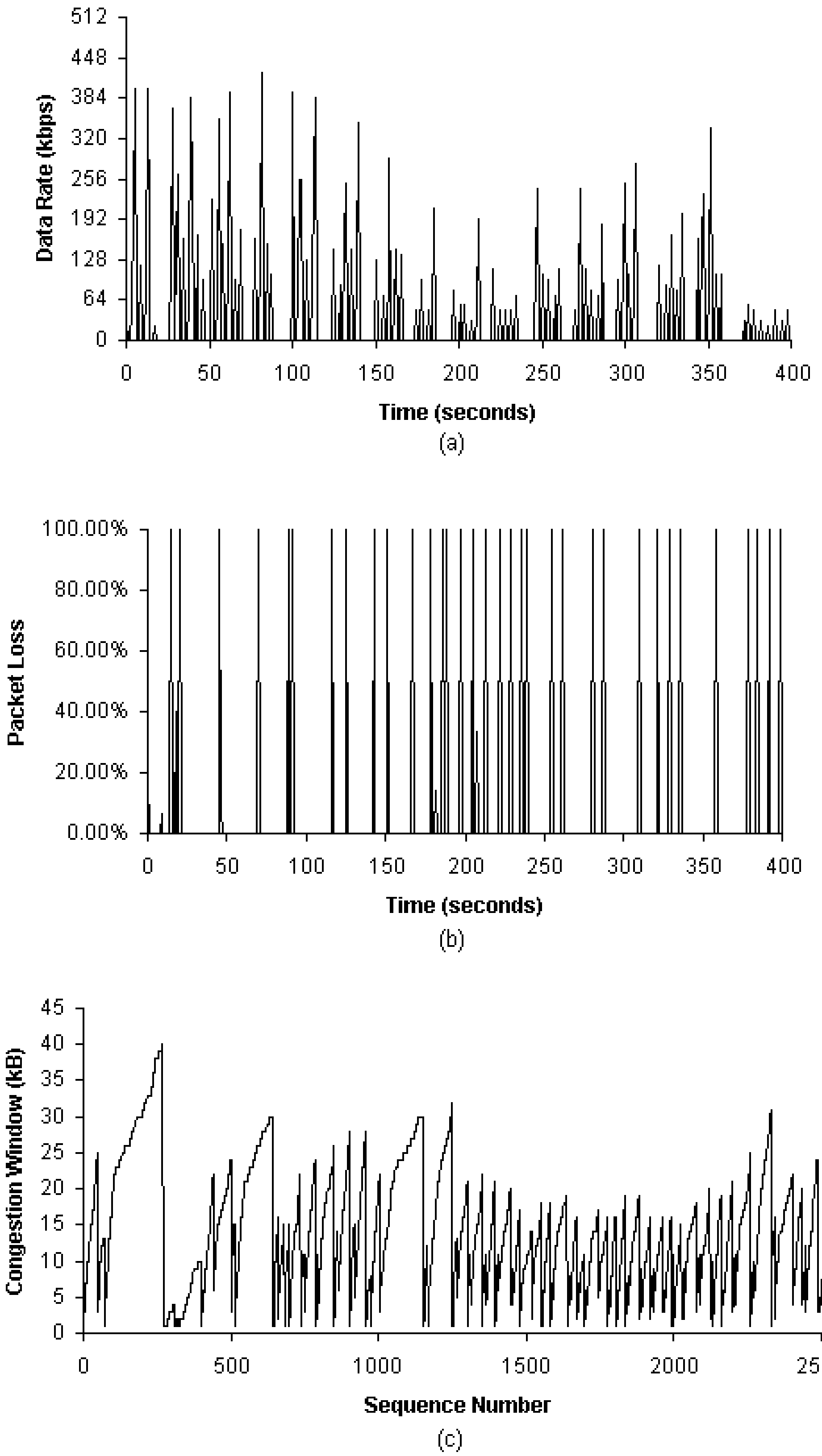
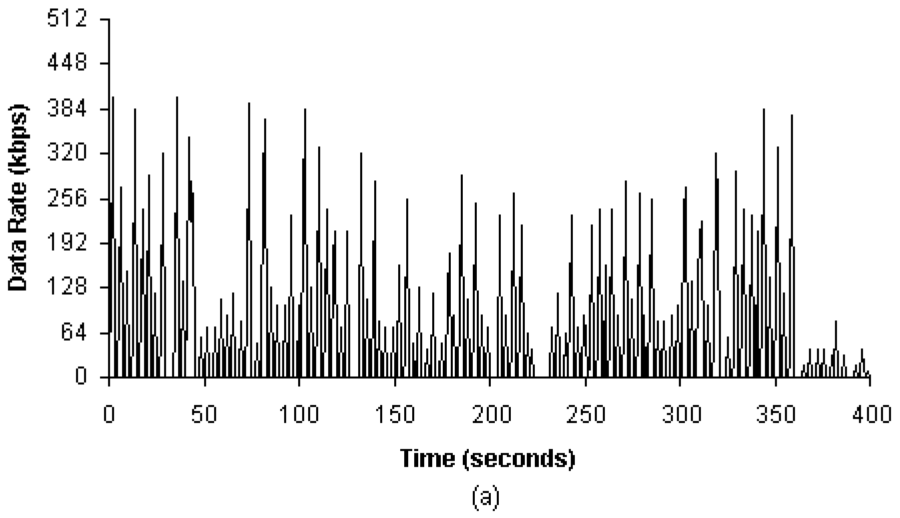
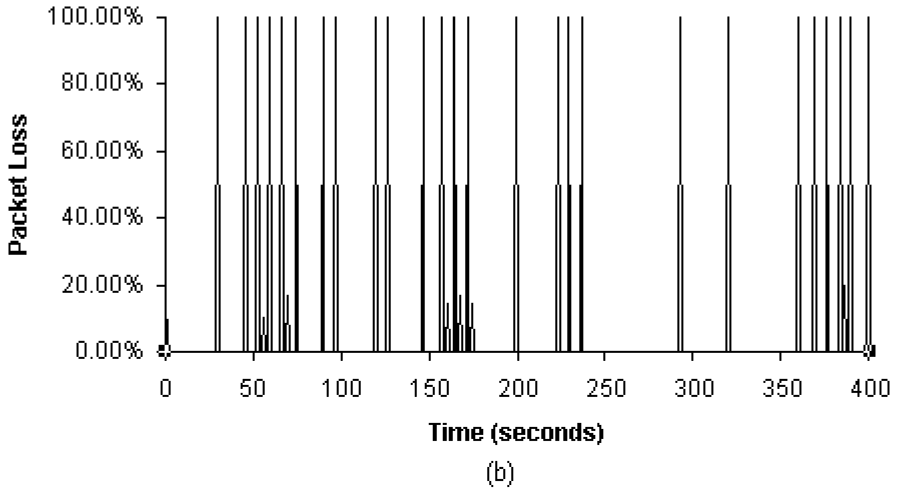


Figure 38: TCP test case 4 results. (a) Throughput. (b) Overall packet loss rate. (c) Congestion window changes.





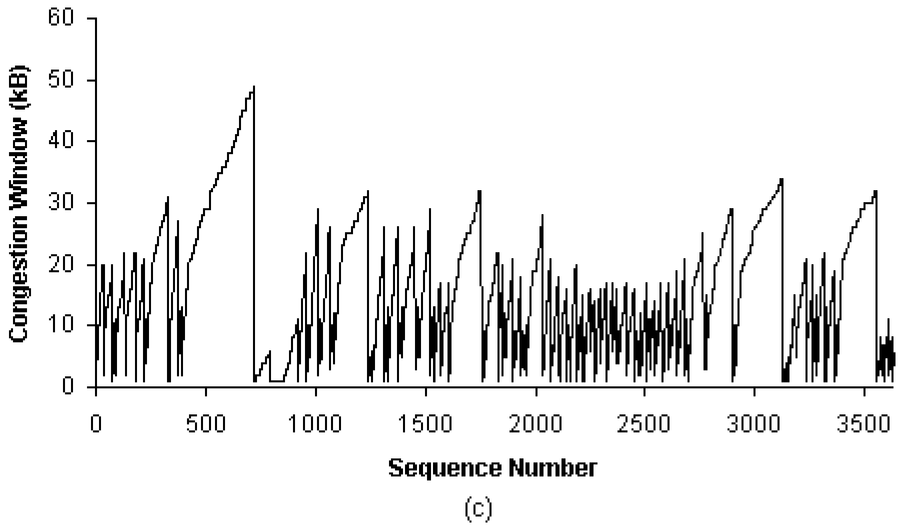


Figure 39: TCP test case 5 results. (a) Throughput. (b) Overall packet loss rate. (c) Congestion window changes.

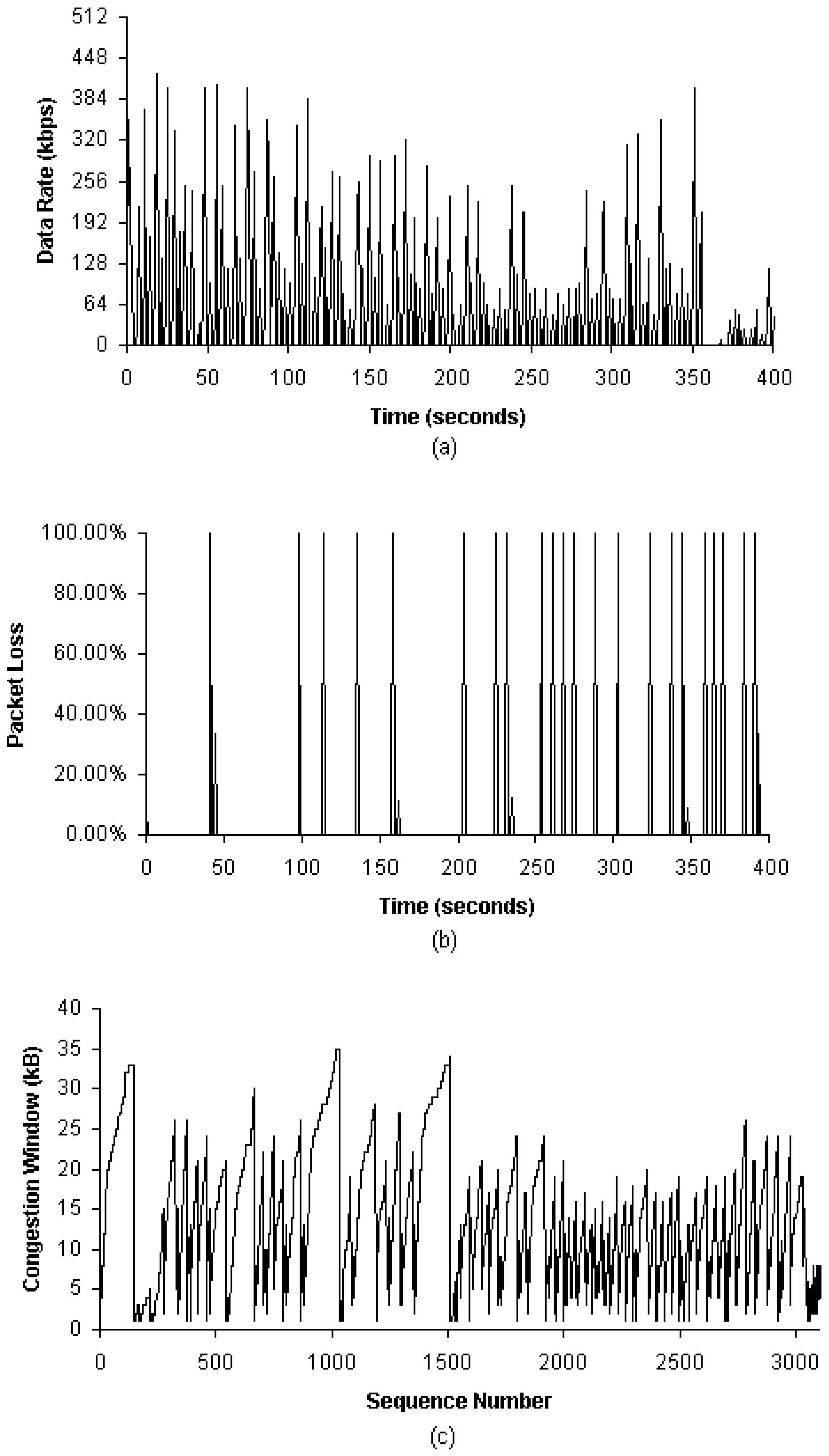
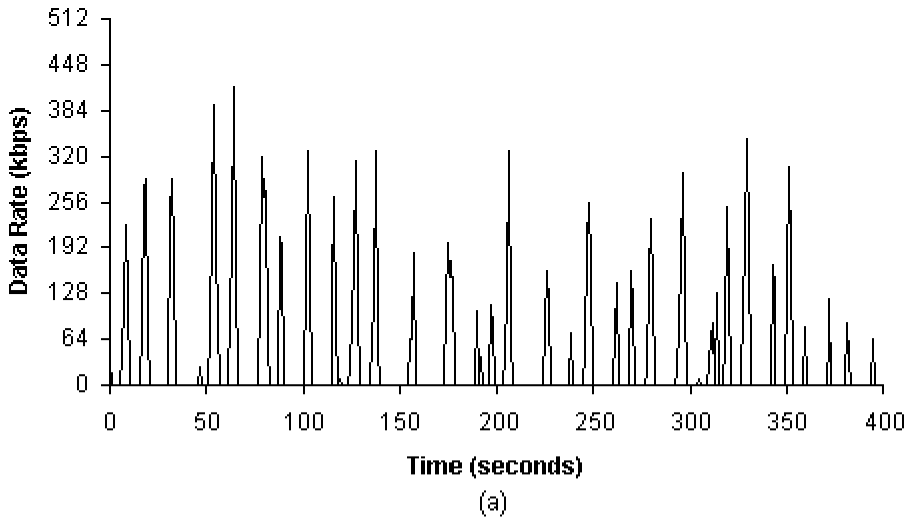
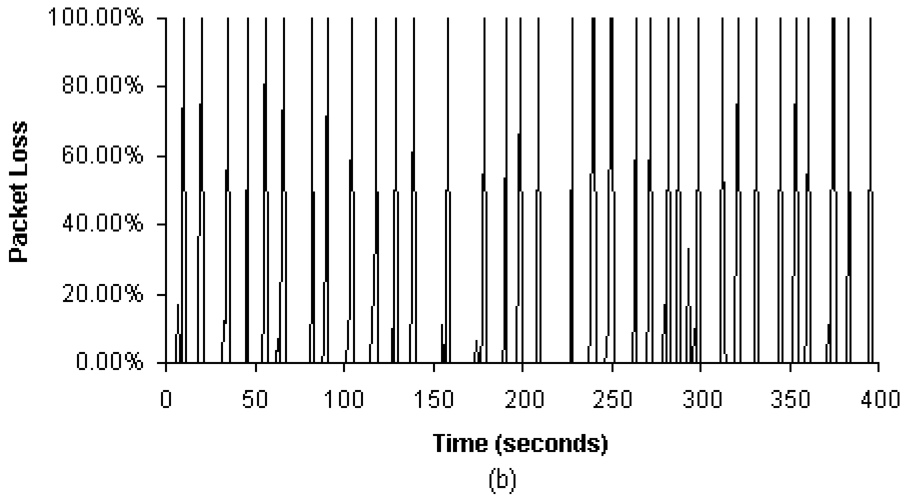


Figure 40: TCP test case 6 results. (a) Throughput. (b) Overall packet loss rate. (c) Congestion window changes.





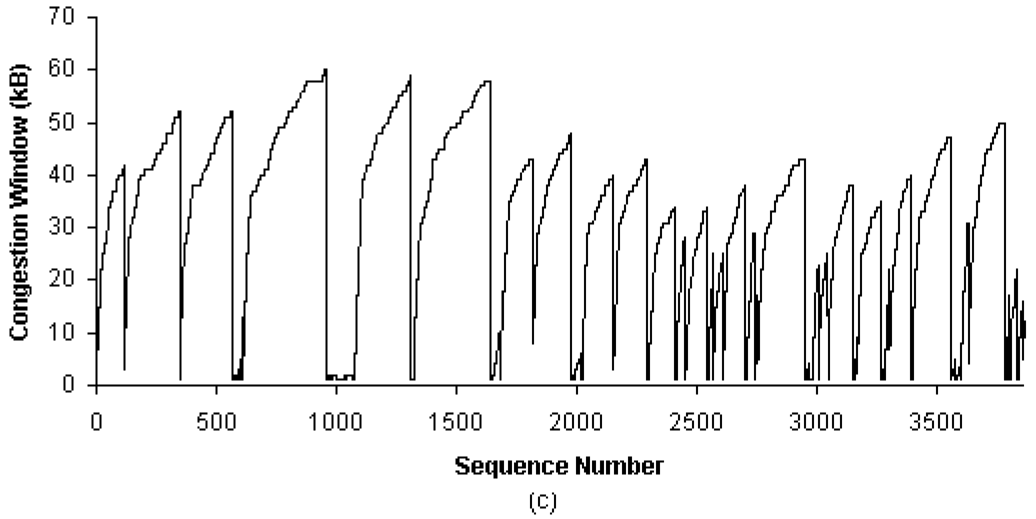
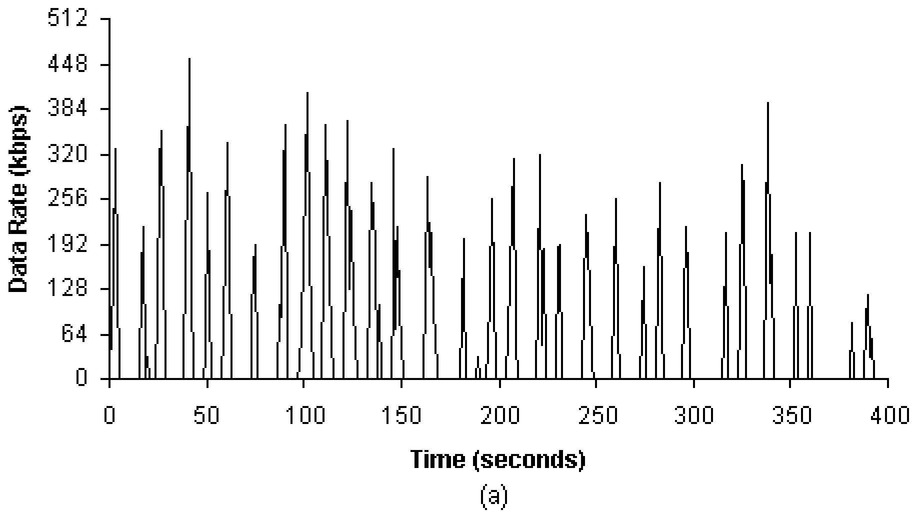
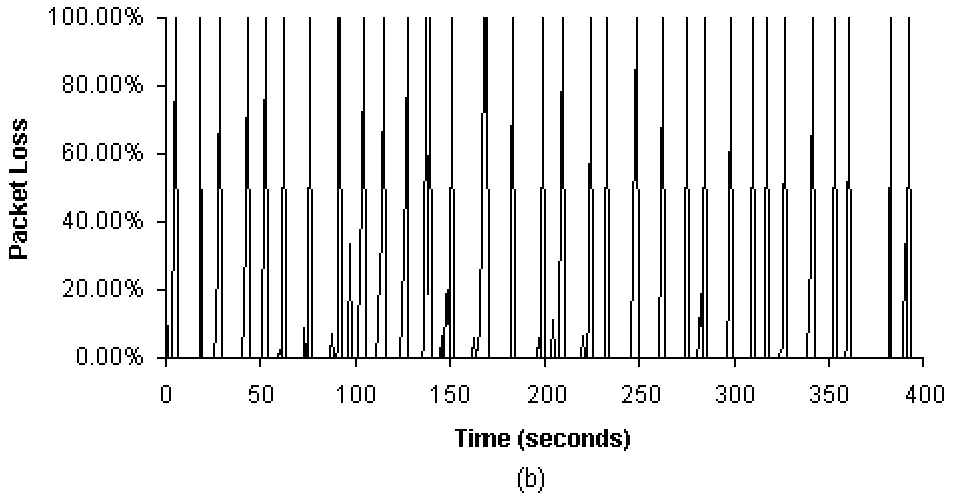


Figure 41: TCP test case 7 results. (a) Throughput. (b) Overall packet loss rate. (c) Congestion window changes.





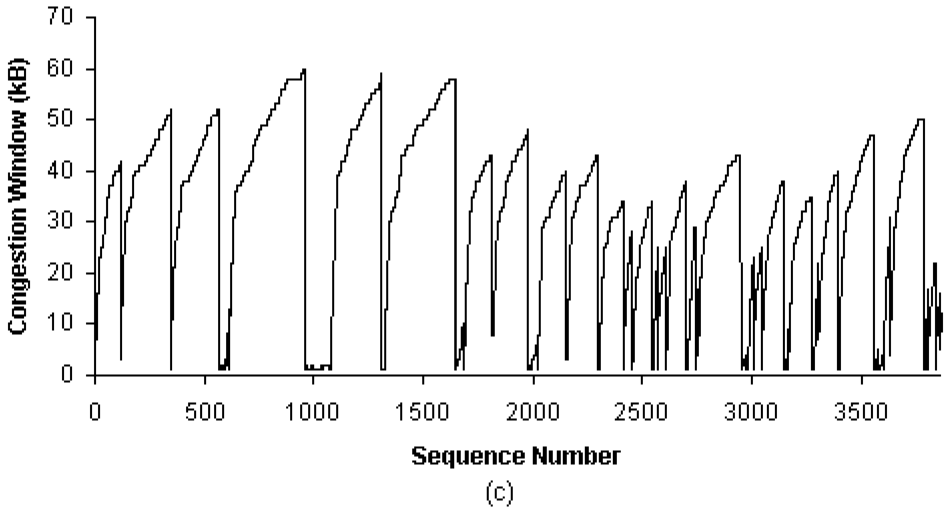
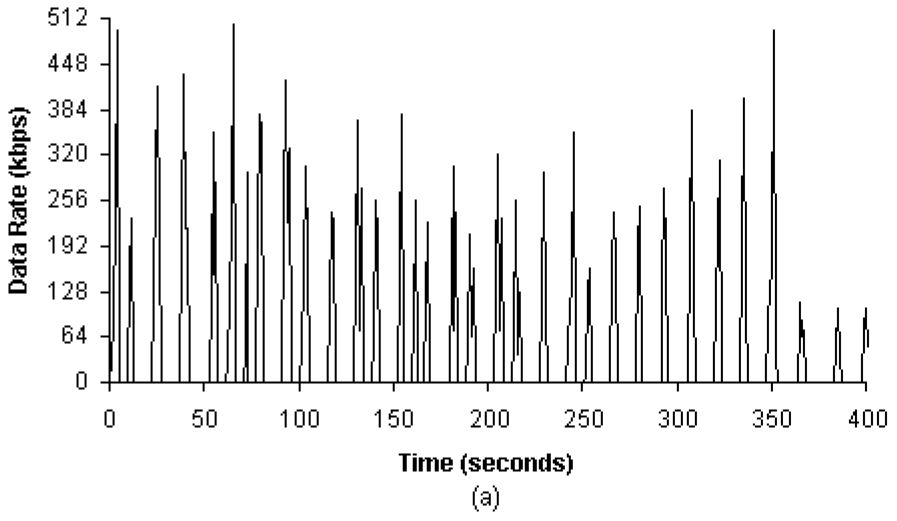
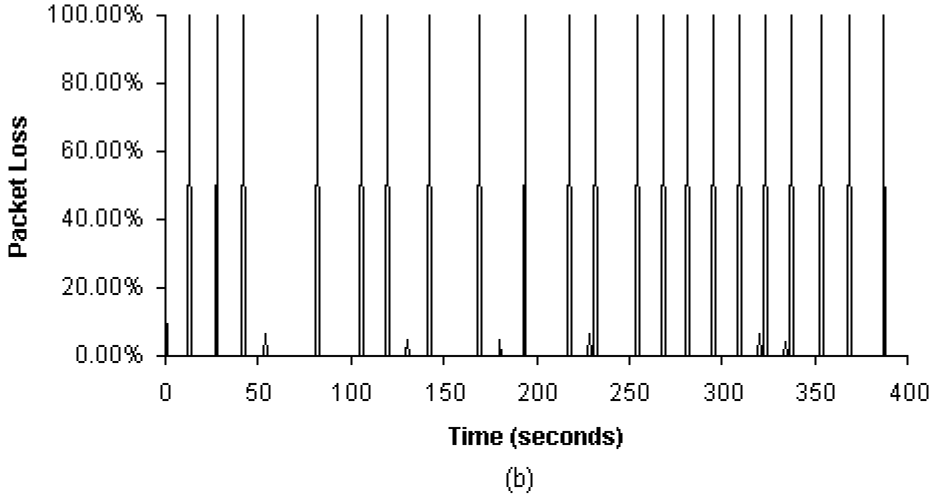


Figure 42: TCP test case 8 results. (a) Throughput. (b) Overall packet loss rate. (c) Congestion window changes.





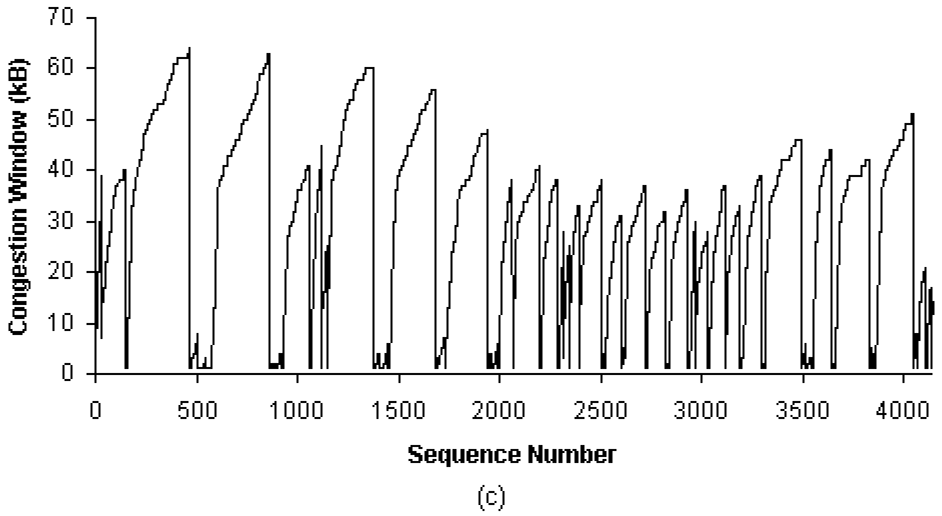


Figure 43: TCP test case 9 results. (a) Throughput. (b) Overall packet loss rate. (c) Congestion window changes.

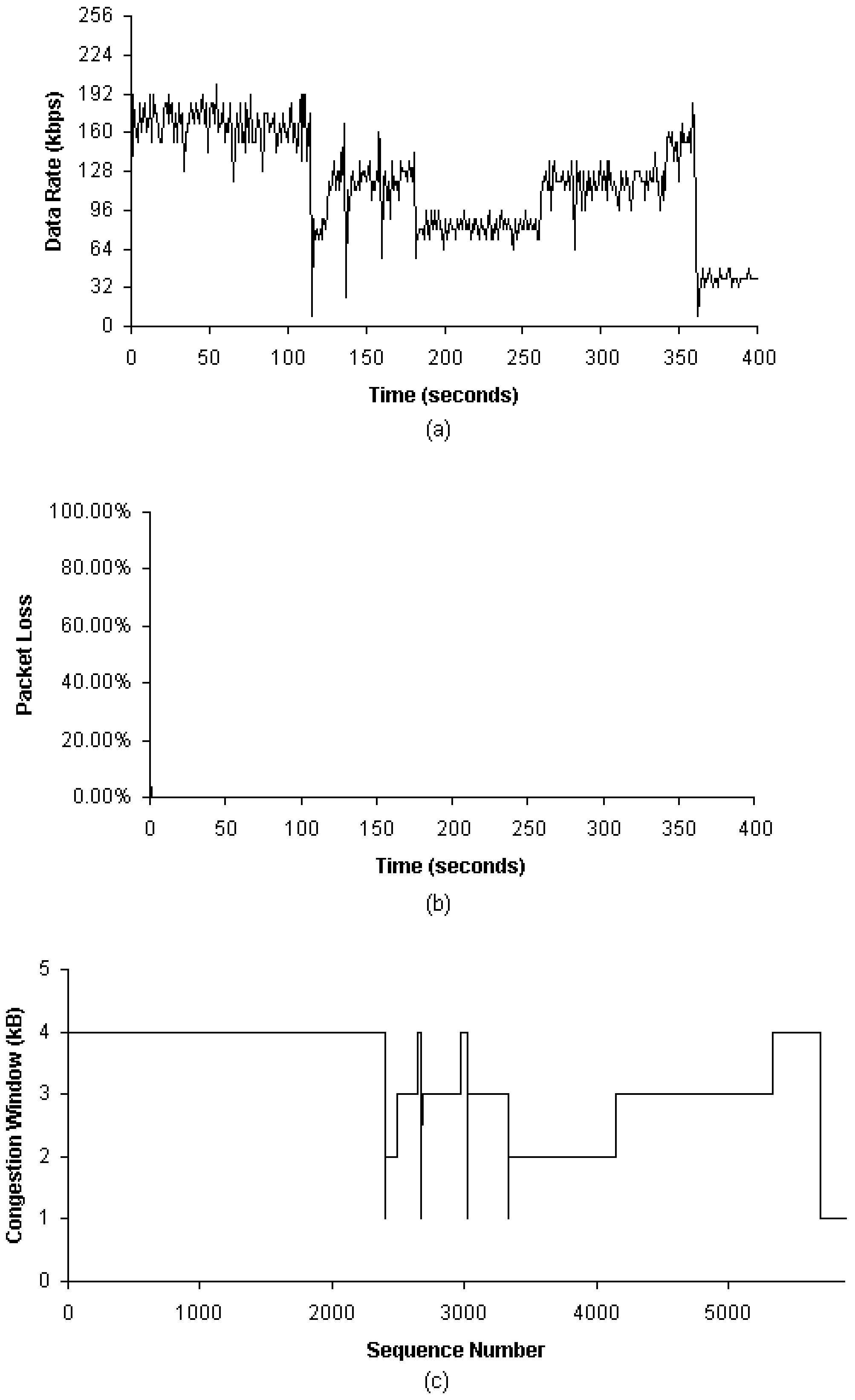
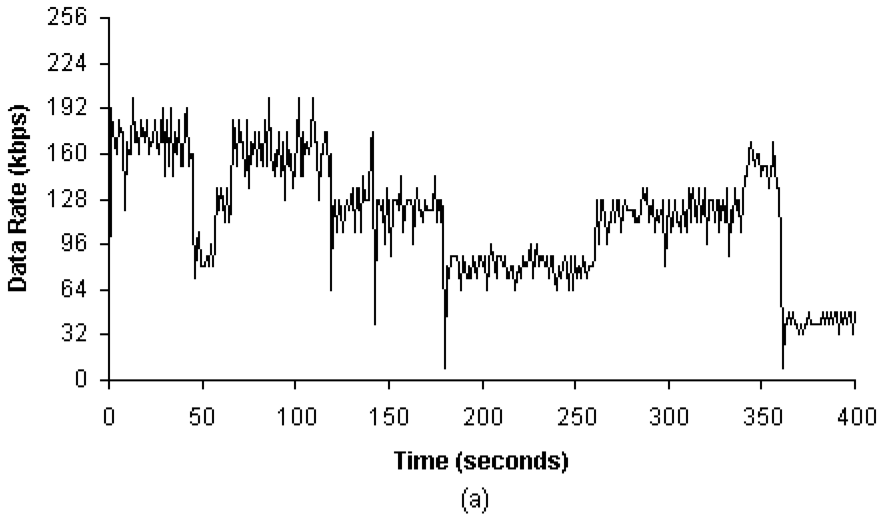
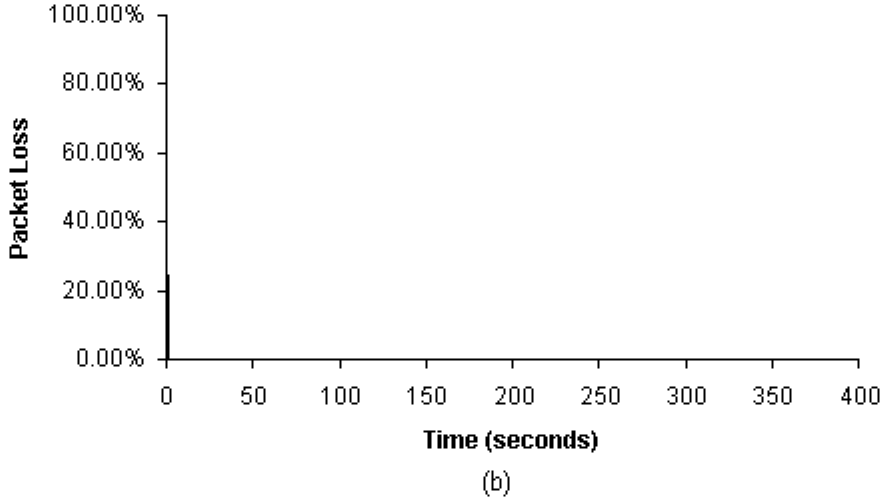


Figure 44: SDTFP test case 1 results. (a) Throughput. (b) Overall packet loss rate. (c) Wired network congestion window changes.





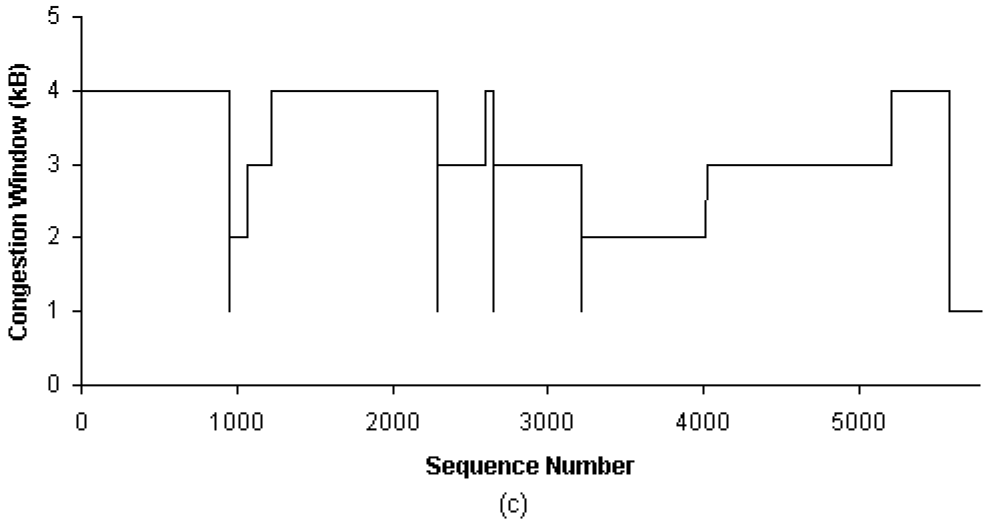
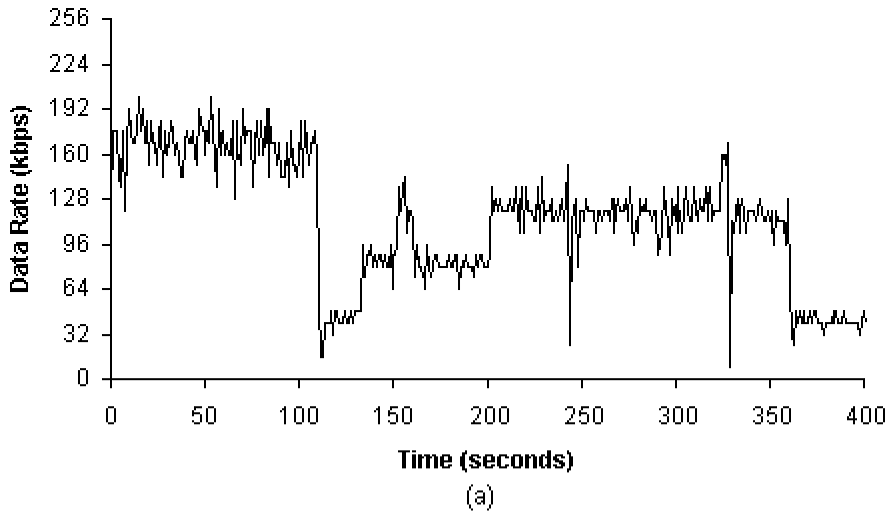
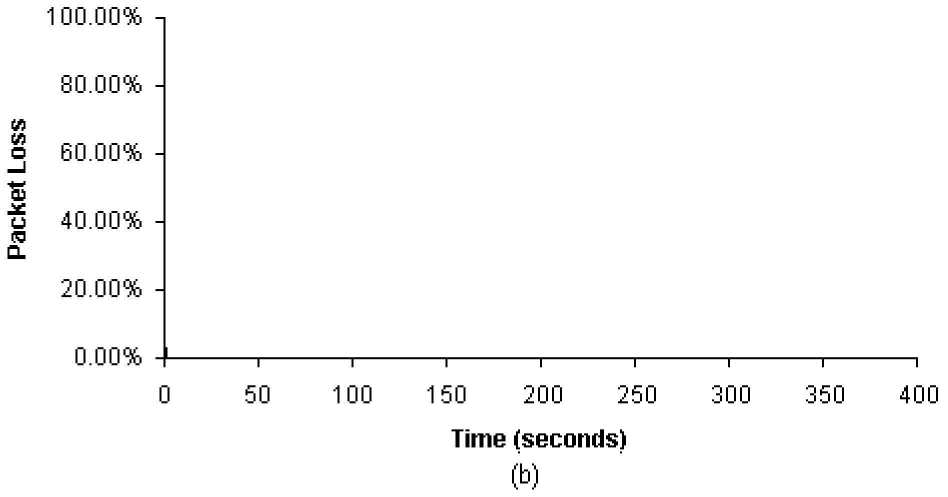


Figure 45: SDTFP test case 2 results. (a) Throughput. (b) Overall packet loss rate. (c) Wired network congestion window changes.





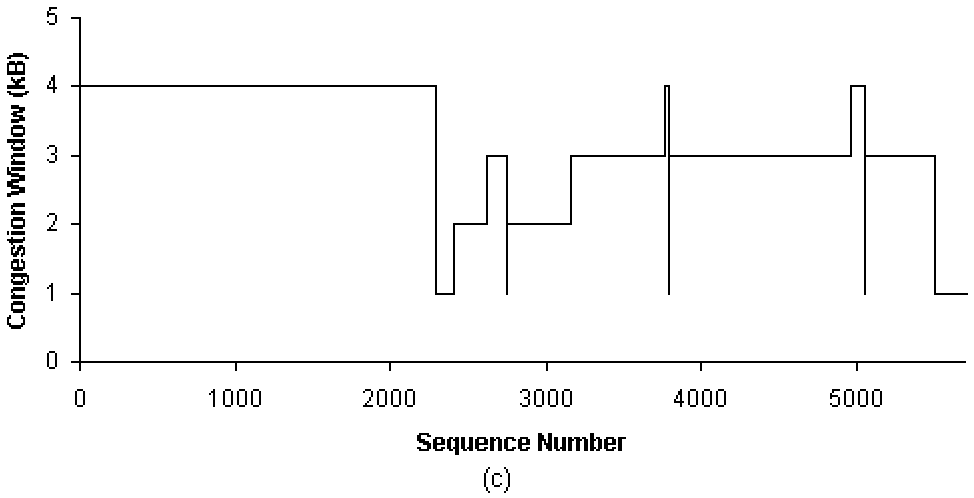


Figure 46: SDTFP test case 3 results. (a) Throughput. (b) Overall packet loss rate. (c) Wired network congestion window changes.

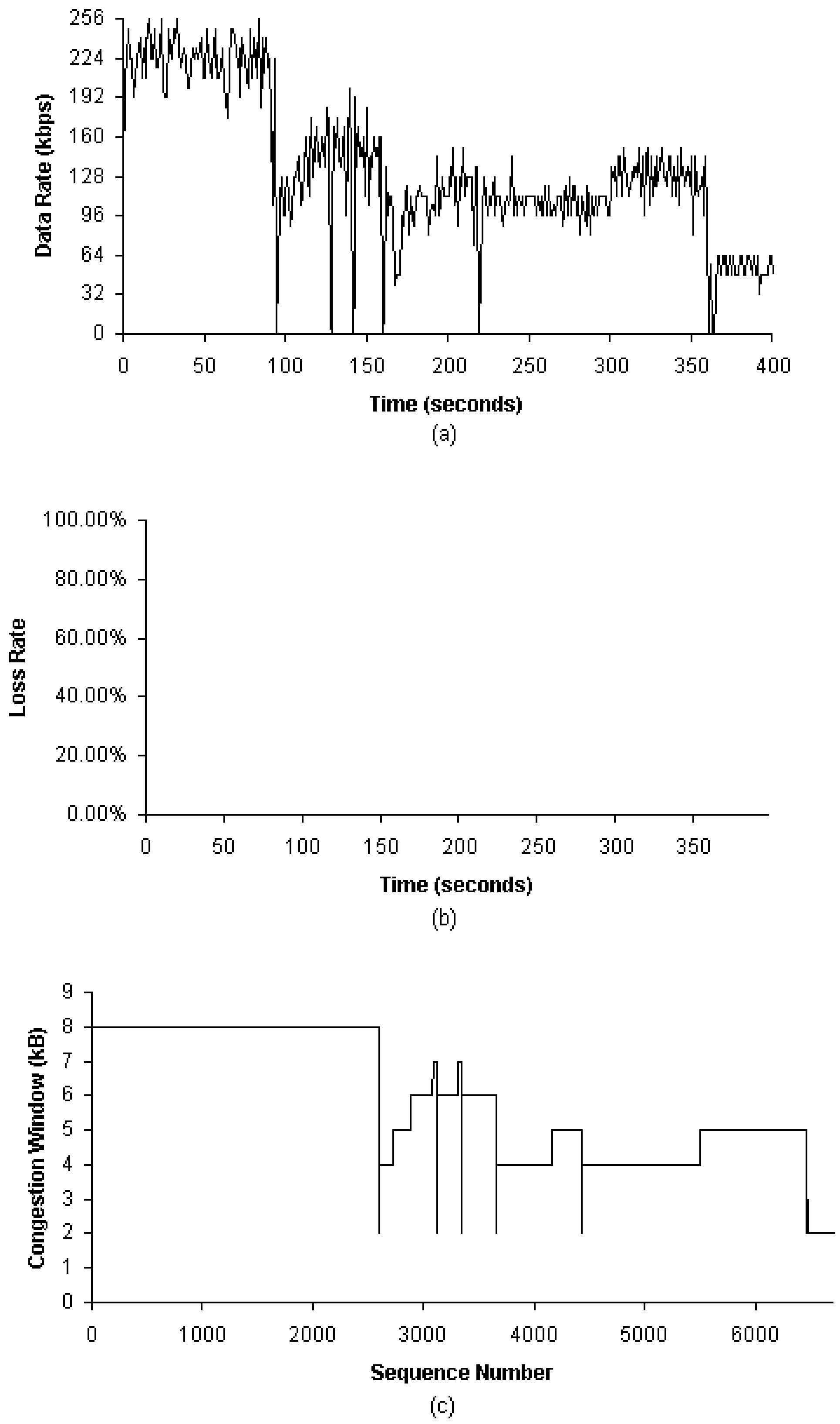


Figure 47: SDTFP test case 4 results. (a) Throughput. (b) Overall packet loss rate. (c) Wired network congestion window changes.

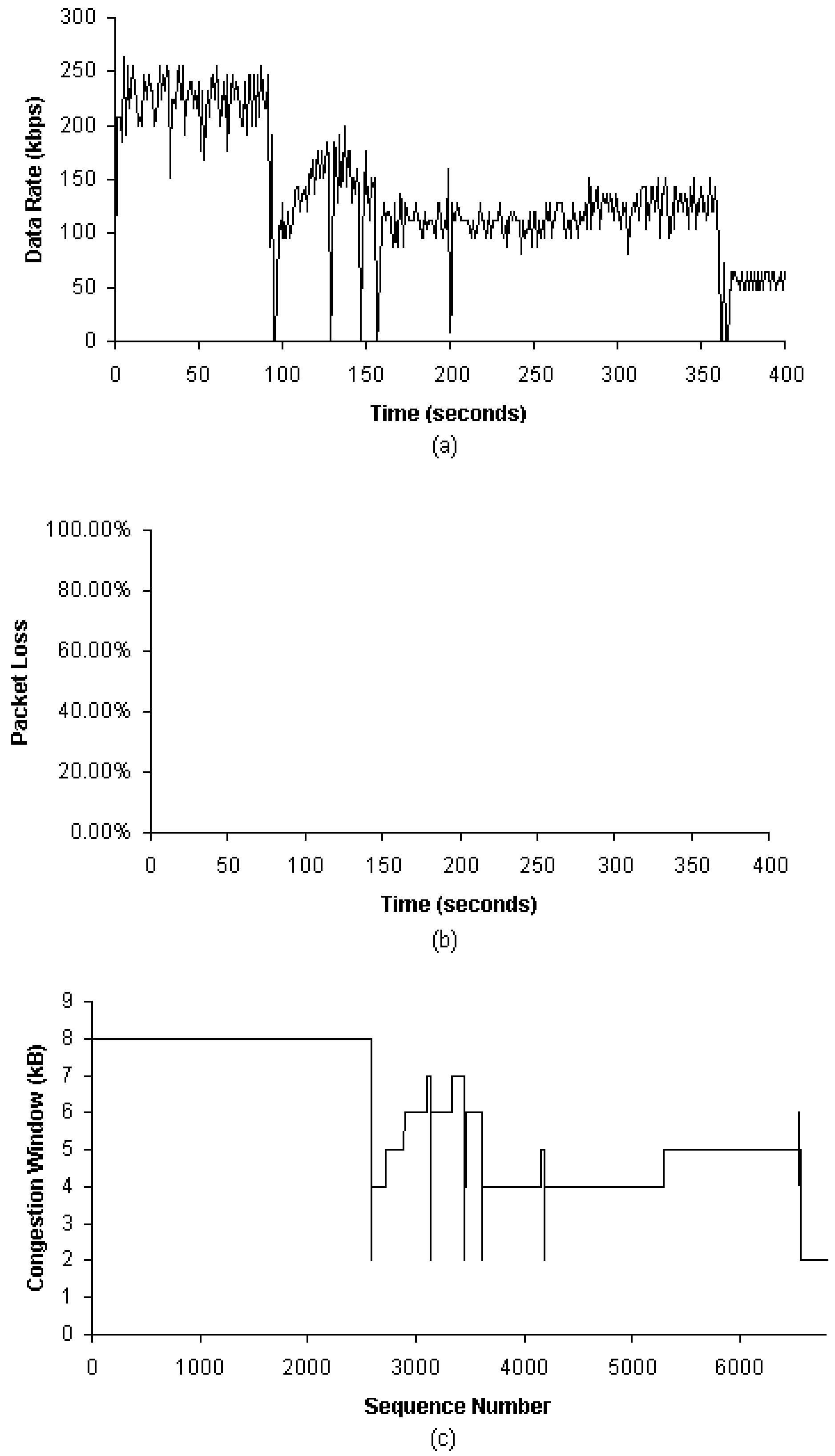
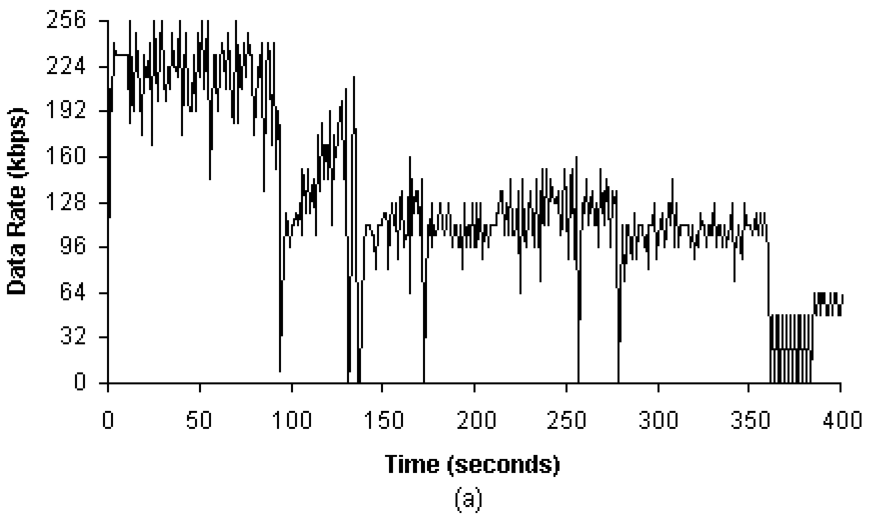
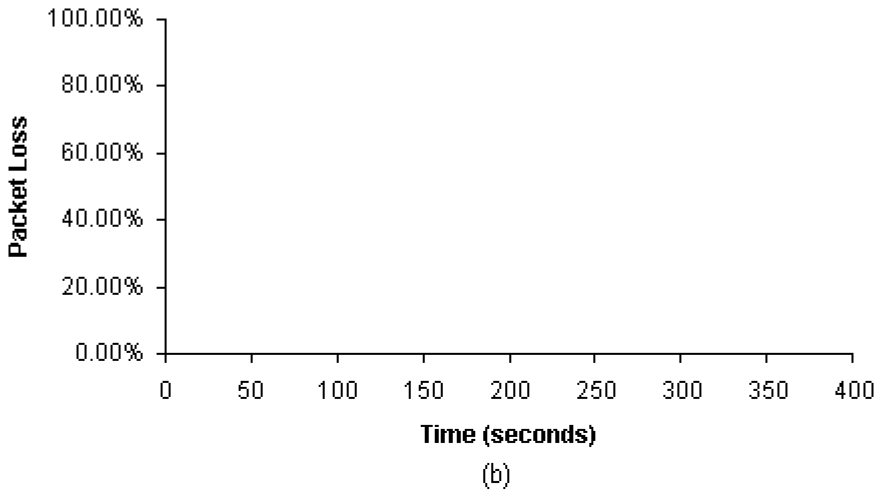


Figure 48: SDTFP test case 5 results. (a) Throughput. (b) Overall packet loss rate. (c) Wired network congestion window changes.





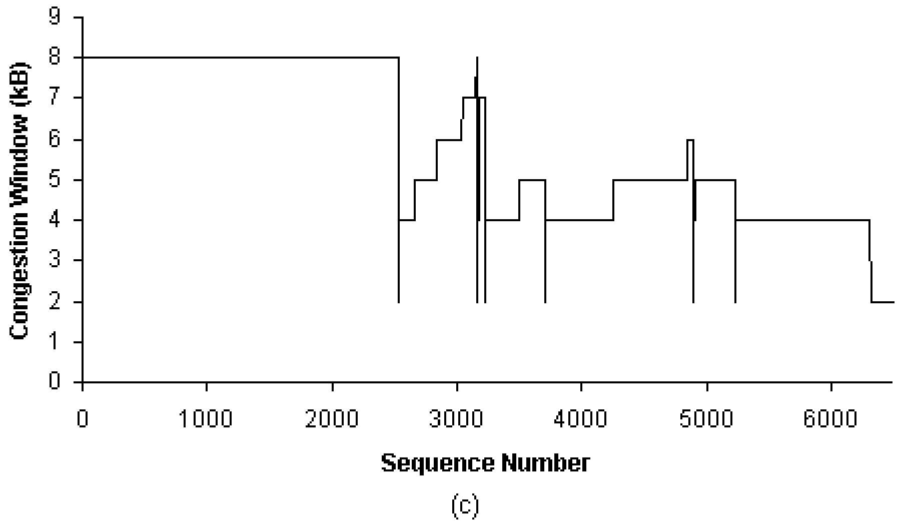
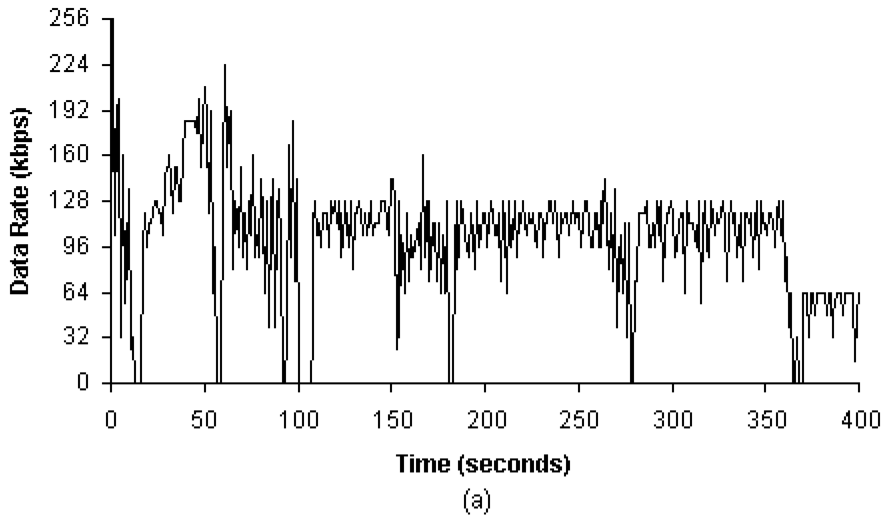
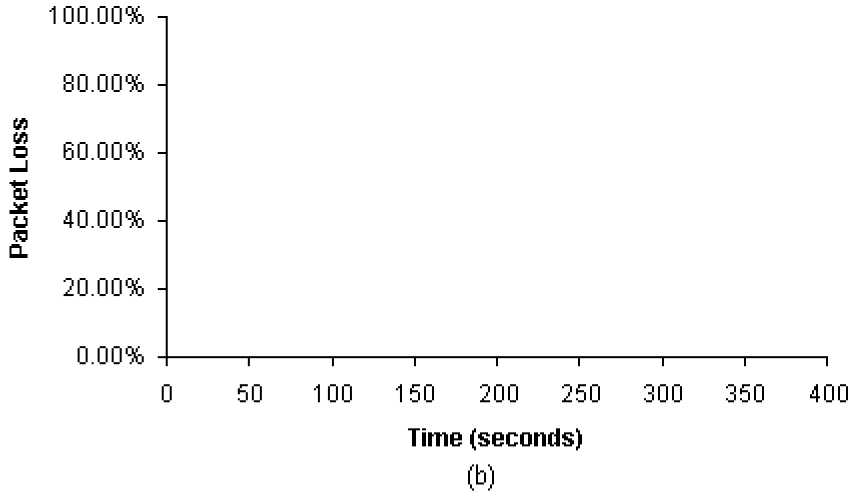


Figure 49: SDTFP test case 6 results. (a) Throughput. (b) Overall packet loss rate. (c) Wired network congestion window changes.





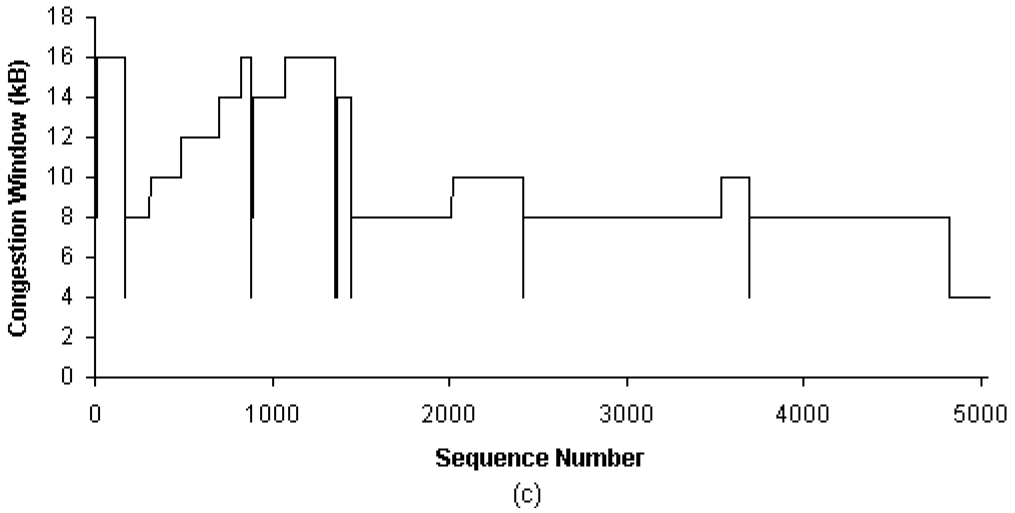
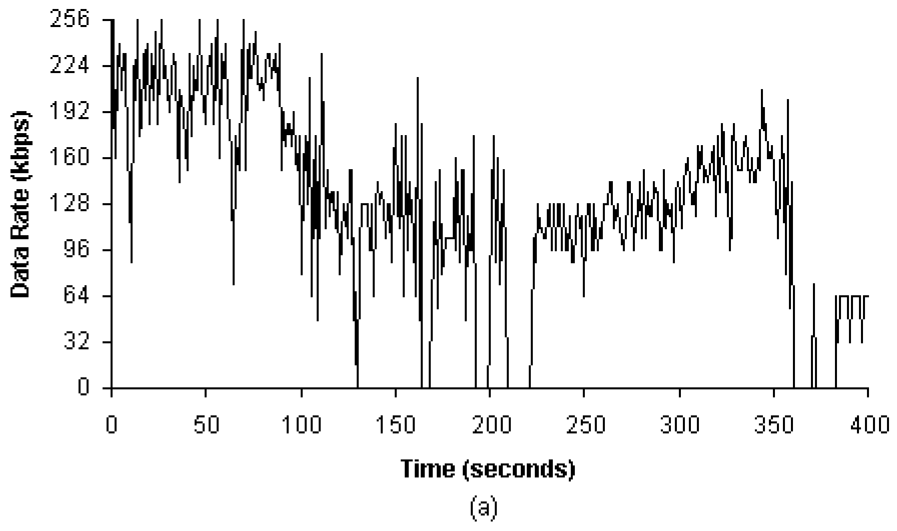
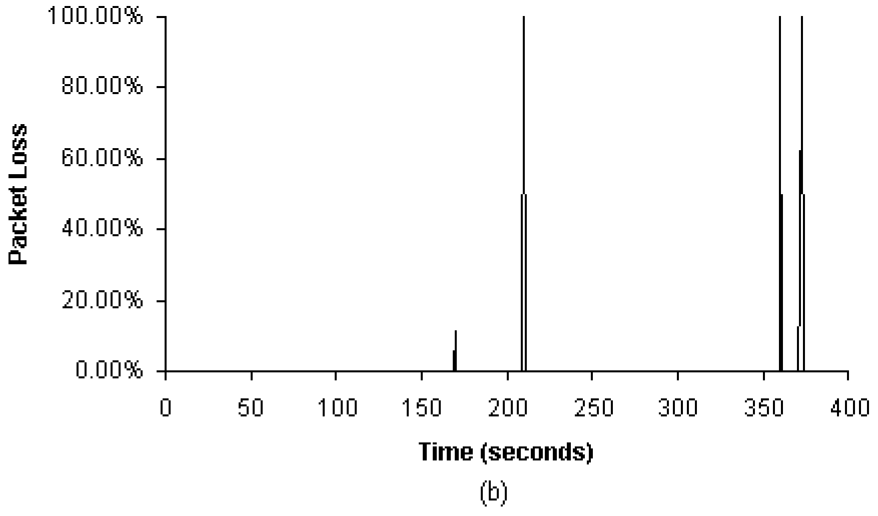


Figure 50: SDTFP test case 7 results. (a) Throughput. (b) Overall packet loss rate. (c) Wired network congestion window changes.





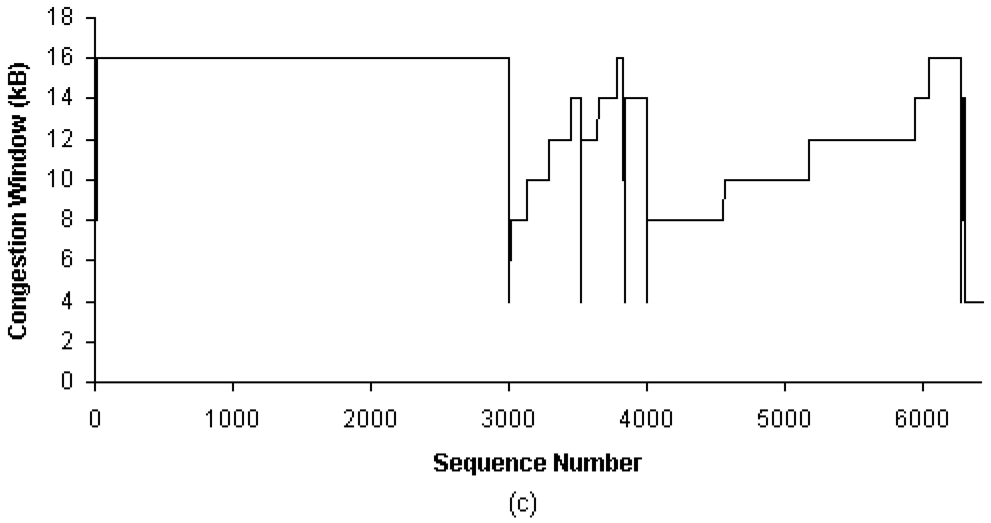
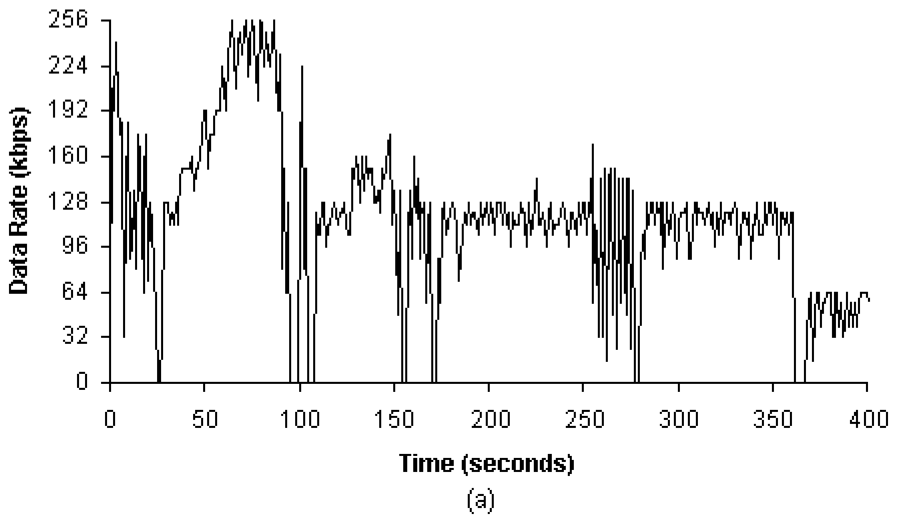
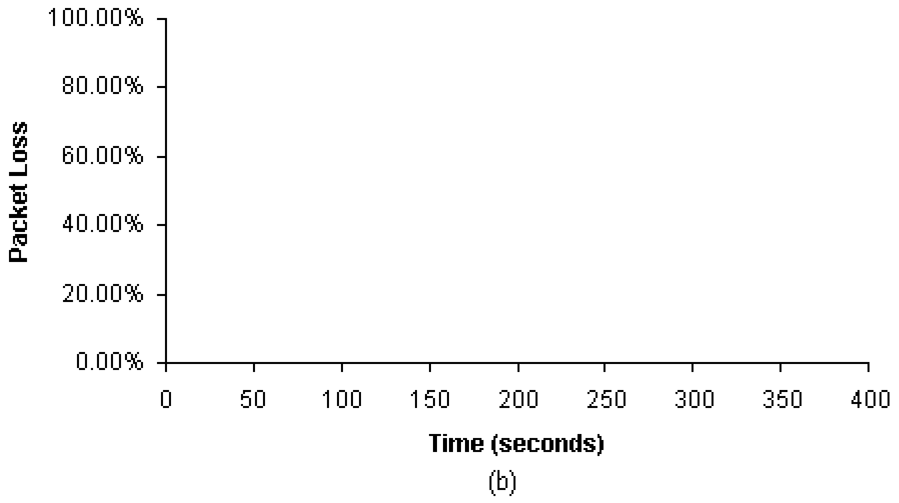


Figure 51: SDTFP test case 8 results. (a) Throughput. (b) Overall packet loss rate. (c) Wired network congestion window changes.





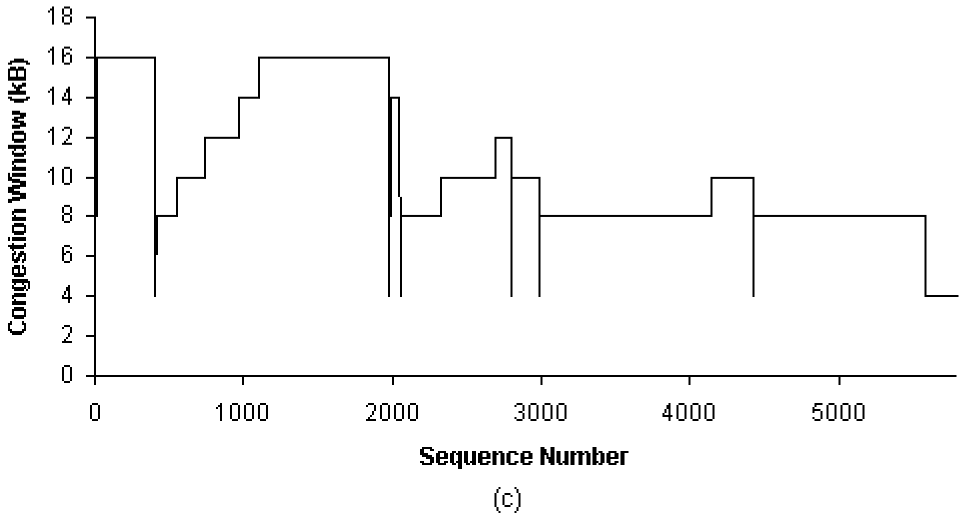


Figure 52: SDTFP test case 9 results. (a) Throughput. (b) Overall packet loss rate. (c) Wired network congestion window changes.

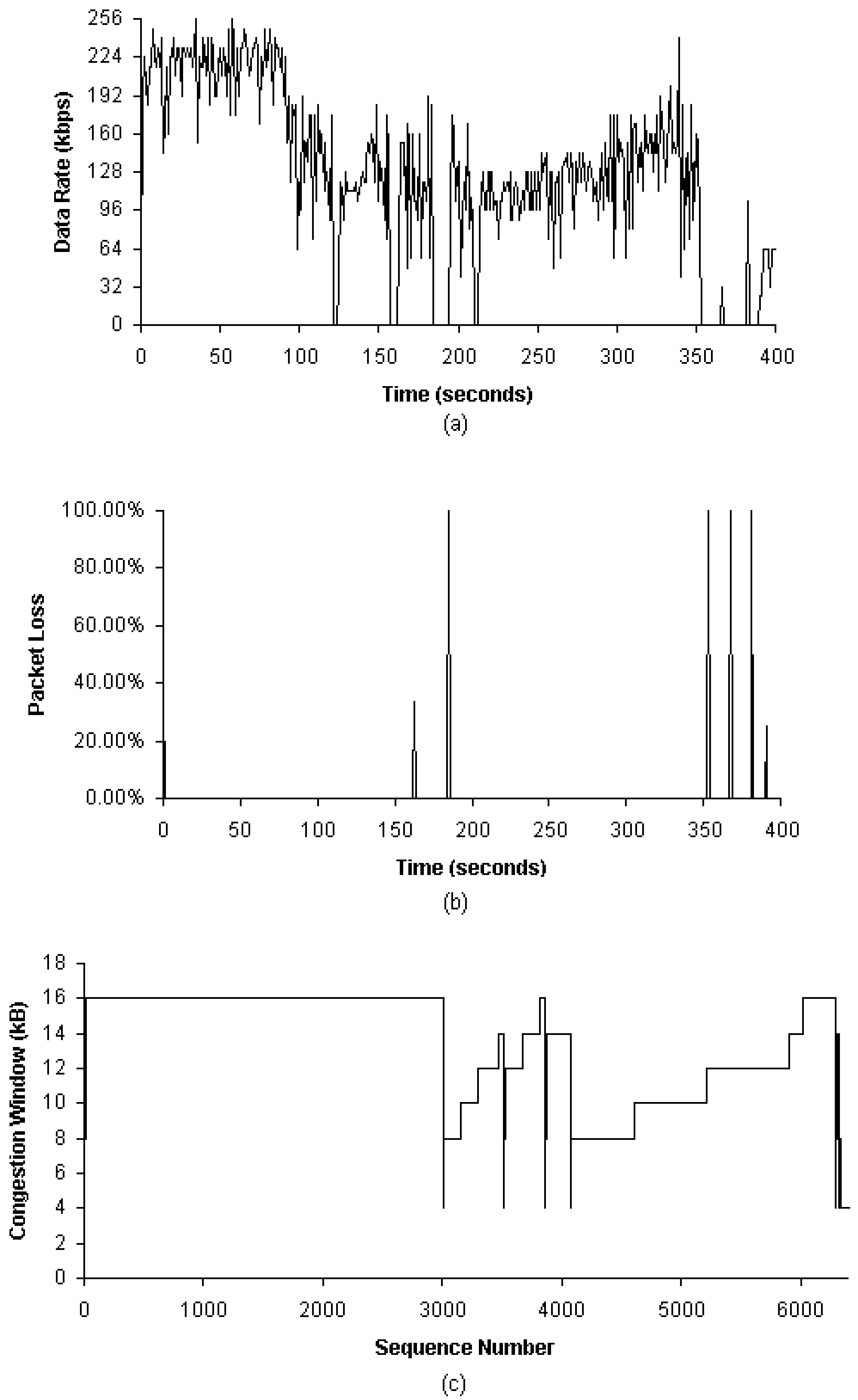
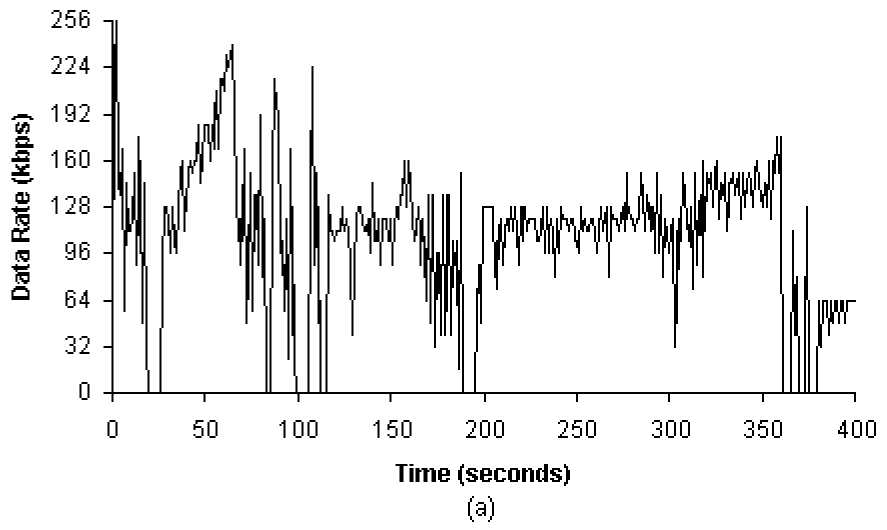
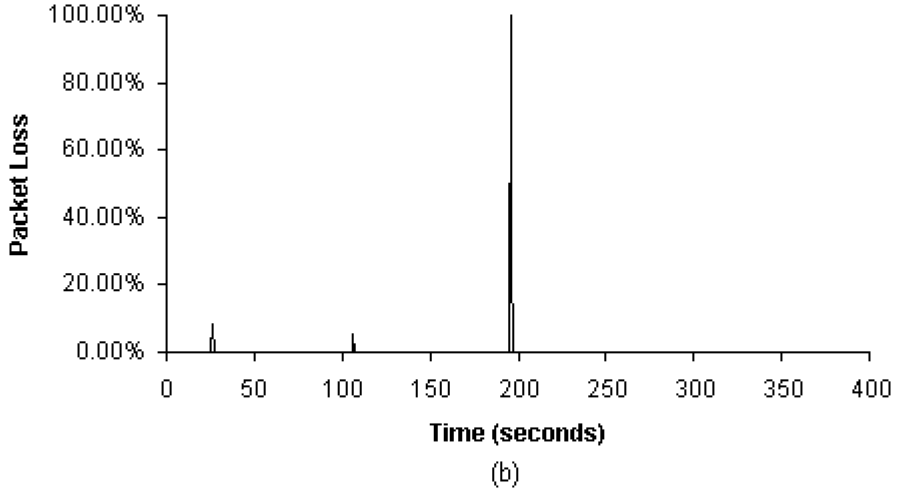


Figure 53: SDTFP test case 10 results. (a) Throughput. (b) Overall packet loss rate. (c) Wired network congestion window changes.





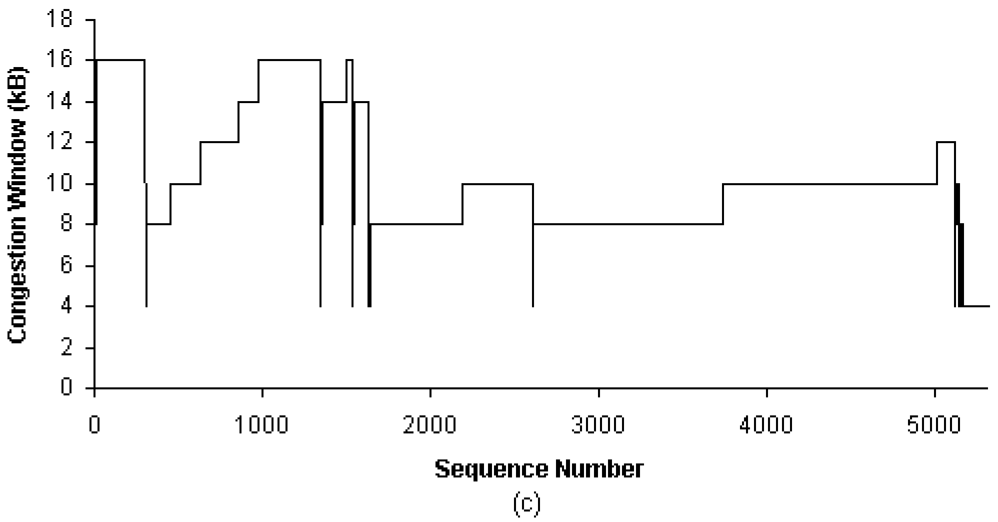
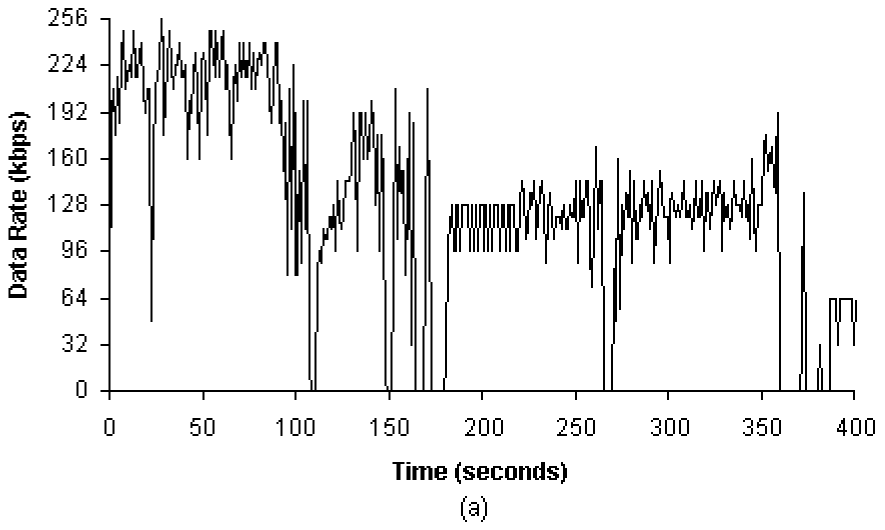
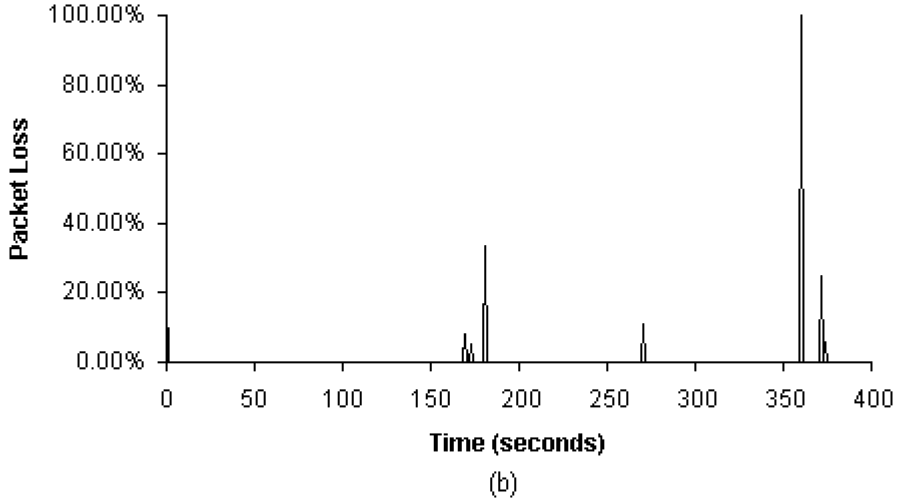


Figure 54: SDTFP test case 11 results. (a) Throughput. (b) Overall packet loss rate. (c) Wired network congestion window changes.





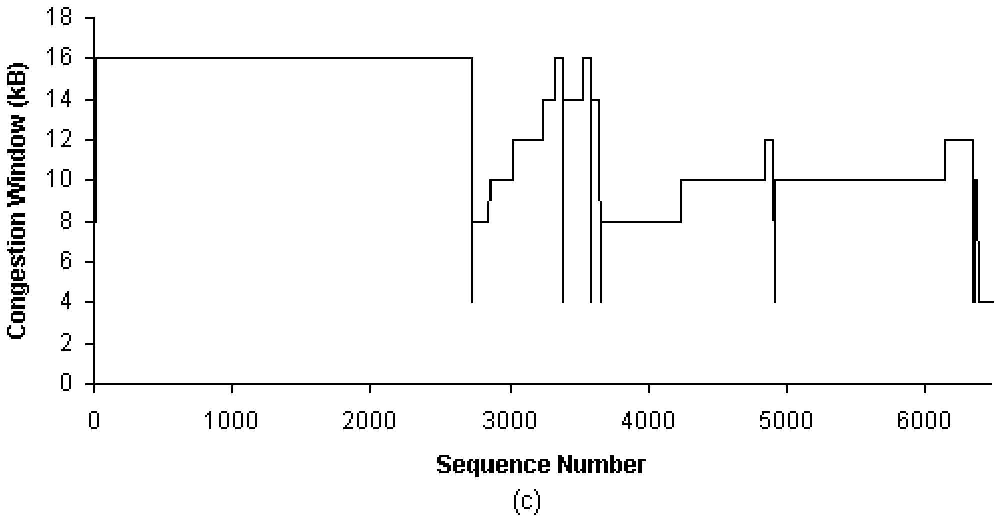


Figure 55: SDTFP test case 12 results. (a) Throughput. (b) Overall packet loss rate. (c) Wired network congestion window changes.

6.4 Simulation Results Analysis

Even a quick glance over the graphically organized simulation results from Figure 26 to Figure 55 would reveal the tremendous extend of performance discrepancy of the three protocols under similar testing conditions. Despite these apparent differences, there exist certain behavioural patterns and weaknesses exhibited by each protocol in their respective test runs. Upon thorough examination, operational merits and performance comparisons of the three protocols are summarized categorically in the following sections.

6.4.1 UDP/RTP

In the first three test cases, the video encoding rate is set at 64 kbps, which is below the minimum traffic threshold of the wired network congestion profile (see Figure 23). As a result, receiver throughput is generally able to maintain at 64 kbps for the majority of the playback duration, and the video QoS is therefore presumably preserved. Still, the UDP/RTP connection fails to fully maximize bandwidth utilization because of the lack of a network traffic probing instrument, whereas the absence of a packet loss recovery mechanism 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}. It leads to frequent 100% packet loss instances that are detrimental to video quality, even with the assistance of MPEG-4 error resilience tools.

When the video encoding rate at the sender is increased to 128 kbps in test cases 4 to 6, the wired network emulator begins to drop packets when the allowed throughput is reduced to 64 kbps in the last 40 seconds of video streaming. During this period, the average packet loss rate, apart from those inflicted by wireless channel errors, ballooned to over 50% with the effective receiver throughput reduced by more than half. Elsewhere, the throughput profile behaves in a similar manner as test cases 1 to 3 with the random bit errors in the wireless channel producing the same disturbing effects and the bandwidth utilization not maximized.

The rise of the video encoding rate to 256 kbps in the last three test cases further magnifies the problems experienced above. Additional packet losses are recorded amidst the wired network congestion profile modifications from 90 to 360 seconds. Although the packet loss rate may seem prominent, in reality the actual throughput may not suffer as much because the persistent UDP/RTP stream of video packets will gradually wrench extra bandwidth from other TCP-compliant connections that are sharing the same path, as illustrated in Figure 12, thereby infringing upon other Internet connections' right for a fair share of transmission bandwidth.

In summary, an end-to-end UDP/RTP connection is able to maintain video QoS only in the absence of wired network 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 TCP-friendliness and the minimization of MPEG-4 error resilience tools deployment at the mobile receiver cannot be guaranteed by UDP/RTP. Overall, the UDP/RTP simulation results provided by test cases 1 to 9 vindicate the protocol's weaknesses pertinent to video streaming as outlined earlier in Section 3.2.

6.4.2 TCP

Despite the disparities in input system parameters, most of the TCP test runs are able to produce a receiver throughput that can, to a certain extent, follow the contours of the wired network 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 AIMD congestion window pattern previously described in Section 3.1.4. Also, bursts of wireless channel errors that corrupt both video packets and SACKs are mishandled by the TCP sender as a flood of timeouts limits the growth of the congestion window to stifle data transmission rate.

Although TCP has often been relied upon to provide data transmission integrity in normal circumstances, unusually high packet losses, which are defined as packets that do not arrive in time for playback processing at the receiver, have been detected in most of the test case results. Closer examination on the figures exposes the fact that undeterred congestion window growth caused the instantaneous data rate to go beyond the 256 kbps threshold, thus leading to unnecessary packet loss in the wired network emulator. Packet stream integrity is also compromised during high BER periods in test cases 1, 4 and 7 as the TCP sender fails to retransmit lost packets in time after long error bursts. In reality, the video encoder would only transmit at a maximum rate of 256 kbps regardless of the instantaneous TCP congestion window dimensions, so the assumed packet losses will be less likely to occur. However, the simulated results still show the futility of unabated TCP congestion window growth for video streaming applications.

Compared to others, test cases 1 to 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 TCP test cases 1 to 3 because of the shorter

RTT at 125 ms and larger buffer to RTT ratio (see Figure 3) that accommodates more retransmission attempts. Test cases 7 to 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.

6.4.3 SDTFP

Under similar testing conditions, SDTFP comprehensively outperforms the other two tested protocols in maintaining a much more stable data throughput at the receiver that closely adheres to the wired network congestion profile via a highly structured and steady congestion control algorithm, while remarkably suffering from virtually no packet loss in all SDTFP 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. Again, the simulations results splendidly demonstrate the advantages of SDTFP as outlined in Section 5.6.1.

Still, the simulation results expose a few areas of minor concern. In test cases 1 to 3 with RTT at 125 ms, SDTFP fails to achieve the intended maximum streaming rate because of the higher ratio of processing time to RTT. The increase in processing time could be due to the heavier computation load instigated by the short delay SACK frequency of 1. While this situation could be remedied by dynamic adjustment of RTT estimation in relation to processing delay, it shows the vulnerability of SDTFP against an inaccurate RTT value.

Even with the luxury of a predetermined constant RTT value, there exist some 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 origin, such slight changes would still be enough to inflict picture smoothness inconsistencies. In practical applications, SDTFP should more closely cooperate with the adaptive rate video encoder to ensure a smoother encoding rate in response to dynamically changing RTT values and SDTFP congestion window dimensions.

During the first 90 seconds of video streaming with the wired network congestion profile at its maximum value of 256 kbps, some test cases experience unexpected timeouts that render the congestion window to undergo 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 to the wired network 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 network 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.

Besides unnecessary reduction in video streaming rate, early timeouts also affect wait time calculations in congestion window management. As indicated in Section 5.2, the wait time is increased by 2 times when a timeout occurs. Therefore early unnecessary timeouts will have a considerable effect on the congestion window recovery efficiency in some later time. As a side note, the wired network congestion profile modifications from 90 to 360 seconds inflicts some rather fluctuating changes on data throughput in some of the test cases. These are mainly due to a short default wait time value of 15 seconds for the purpose of expediting simulations. An easy solution is to set a longer wait time, say 1 minute, during practical video streaming sessions.

In SDTFP test cases with RTT = 500 ms, as expected, a smaller delay SACK frequency of 2 causes the sender to commit unnecessary retransmits, thus inundating the wired network 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 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 SDTFP should be able to dynamically determine the optimal delay SACK frequency to balance the tradeoff between data throughput maximization and packet loss rate reduction.

7 Conclusions and Future Work

Because of the discrepancies in the intrinsic physical characteristics between both wired and wireless domains, there exist a number of design challenges in implementing cross-domain video streaming connections. Current protocols such as TCP and UDP often fail to provide a comprehensive solution to overcome the many cross-domain video streaming hurdles. Devising a single optimized end-to-end protocol for cross-domain video streaming applications remains one of the prime challenges for the research community.

This thesis proposes the Split-Domain TCP-Friendly Protocol for MPEG-4 adaptive rate video streaming over 3G networks that aims to maintain video QoS amidst wired network congestion, achieve fairness (or TCP-friendliness) in wired network resource allocation, cope with high bit error rate in wireless links, minimize the deployment of MPEG-4 error resilience tools at the mobile receiver, and exploit split wired-wireless domain architecture. The principal features of SDTFP 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.

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

During simulations, SDTFP demonstrates a more stable data throughput at the receiver with virtually no packet loss recorded in all test scenarios. In comparison with end-to-end TCP and UDP/RTP emulated video streaming connections under similar testing conditions, SDTFP shows convincing superiority in all performance criteria. However, there exist some minor concerns within the SDTFP protocol design that slightly hinder the overall performance. Further protocol improvements will address these subtle issues.

As the current set of simulations is based on a number of underlying assumptions and network model simplifications, future testing should stride for a closer portrayal of practical usage scenarios. Specifically, custom adaptive rate MPEG-4 codecs should be constructed and incorporated into the system model to produce genuine video streaming data. Also, a more sophisticated cross-domain traffic profile that includes realistic Internet congestion episodes and extensions on the two-state Markov wireless link BER model should be considered. In addition, specific considerations such as connection management procedures, inter-cell handoffs, interactive user features (video seeking, security), and multicast capabilities should be explored in further research projects in order to refine the rudimentary SDTFP specifications into a truly integrated cross-domain video streaming protocol.

8 References

1. "Streaming Methods: Web Server vs. Streaming Media Server", Microsoft Corporation, October 2001. (http://www.microsoft.com/windows/windowsmedia/compare/webservvstreamserv.asp)
2. J. Postel, "Transmission Control Protocol", RFC 0793, IETF, September 1981.
3. A. Bakre, and B. Badrinath, "Implementation of Performance Evaluation of Indirect TCP", *IEEE Trans. on Computers*, vol. 46, no.3, March 1997, pp. 260-278.
4. H. Balakrishnan, V.N. Padmanabhan, S. Seshan, and R.H. Katz, "A Comparison of Mechanisms for improving TCP Performance over Wireless Links", *IEEE/ACM Trans. on Networking*, vol. 5, no. 6, December 1997, pp. 756-769.
5. J. Postel, "User Datagram Protocol", RFC 0789, IETF, August 1980.
6. S. Floyd and K. Fall, "Promoting the Use of End-to-end Congestion Control in the Internet", *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, August 1999, pp. 458-472.
7. R. Koenen, ed., "Overview of the MPEG-4 Standard", ISO/IEC JTC1/SC29/WG11, March 2001. (http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm)
8. R. Talluri, "Error-resilient video coding in the ISO MPEG-4 standard", *IEEE Communications Magazine*, vol. 36, no. 6, June 1998, pp.112-119.
9. J. Widmer, R. Denda, and M. Mauve, "A Survey on TCP-Friendly Congestion Control", *IEEE Network Magazine*, vol. 15, no. 3, May/June 2001, pp. 28-37.
10. C. Hsu, A. Ortega, and M. Khansari, "Rate Control for Robust Video Transmission over Burst-Error Wireless Channels", *IEEE Journal Selected Areas in Communications*, vol. 17, no. 5, May 1999, pp. 756-773.

11. K. Yu, D. Onishi, and R. Van Dyck, “Modification of ETFTP for MPEG-4 Wireless Video Transport”, *IEEE MILCOM Proceedings*, vol. 2, 1999, pp.1221-1225.

12. “WAP Architecture”, WAP Forum, July 2001.

13. V. Jacobson, R. Braden, and D. Borman, "TCP Extensions for High Performance", RFC 1323, IETF, May 1992.

14. M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control", RFC 2581, IETF, April 1999.

15. M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, “TCP Selective Acknowledgement Options”, RFC 2018, IETF, October 1996.

16. K. Y. Wang and S. K. Tripathi, “Mobile-End Transport Protocol: An Alternative to TCP/IP Over Wireless Links”, *IEEE INFOCOM Proceedings*, Vol. 3, 1998, pp. 1046-1053.

17. A. Tanenbaum, “Computer Networks”, 3rd Edition, Prentice Hall, US, 1996.

18. K. Gerd, “Optical Fiber Communications”, 3rd Edition, McGraw-Hill, US, 2000.

19. P. Rysavy, “The Evolution of Cellular Data: On the Road to 3G”, Rysavy Research, 1999. (http://www.rysavy.com/Articles/3G/3g.htm)

20. T. S. Rappaport, “Wireless Communications: Principles & Practice”, Prentice Hall, US, 1996.

21. V. K. Garg and J. E. Wilkes, “Wireless and Personal Communications Systems”, Prentice Hall, US, 1996.

22. M. Degermark, H. Hannu, L. Jonsson and K. Svanbro, “Evaluation of CRTP Performance Over Cellular Radio Links”, *IEEE Personal Communications*, vol. 7 no. 4, August 2000, pp. 20-25.

23. C. Perkins, ed., “IP Mobility Support for IPv4”, RFC 3220, IETF, January 2002.

24. J. Postel, “Internet Protocol”, RFC 0791, IETF, September 1981.
25. G. Montenegro, S. Dawkins, M. Kojo, V. Magret and N. Vaidya, “Long Thin Networks”, RFC 2757, IETF, January 2000.
26. V. Paxson and M. Allman, “Computing TCP’s Retransmission Timer”, RFC 2988, IETF, November 2000.
27. S. Floyd and T. Henderson, “The NewReno Modification to TCP’s Fast Recovery Algorithm”, RFC 2582, IETF, April 1999.
28. R. Braden, Editor, “Requirements for Internet Hosts – Communication Layers”, RFC 1122, IETF, October 1989.
29. S. Floyd and K. Fall, “Simulation-based Comparisons of Tahoe, Reno, and SACK TCP”, *Computer Communication Review*, vol. 26 no. 3, July 1996, pp. 5-21.
30. G. Huston, “TCP Performance”, *Internet Protocol Journal*, vol. 3, no. 2, Cisco Systems, June 2000.
31. G. Huston, “TCP in a Wireless World”, *IEEE Internet Computing*, vol. 5, no. 2, March-April 2001, pp. 82-84.
32. E. Amir, H. Balakrishnan, S. Seshan, and R.H. Katz, “Efficient TCP over Networks with Wireless Links”, Hot Topics in Operating Systems, 1995. (HotOS-V), Proceedings, Fifth Workshop on, 1995, pp. 35-40.
33. E. Ayanoglu, S. Paul, T.F. LaPorta, K.K. Sabnani, and R.D. Gitlin, “AIRMAIL: A Link-Layer Protocol for Wireless Networks”, *ACM/Baltzer Wireless Networks Journal*, February 1995, pp. 47-60.
34. Z.J. Haas and P. Agrawal, “Mobile-TCP: an Asymmetric Transport Protocol Design for Mobile Systems”, *Communications, ICC ’97,* Montreal, Canada, vol. 2, 1997, pp. 1054-1058.

35. K. Ramakrishnan and S. Floyd, “A Proposal to Add Explicit Congestion Notification (ECN) to IP”, RFC 2481, IETF, January 1999.

36. M. Allman, ed., “Ongoing TCP Research Related to Satellites”, RFC 2760, IETF, February 2000.

37. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications”, RFC 1889, IETF, January 1996.

38. H. Schulzrinne, “RTP Profile for Audio and Video Conferences with Minimal Control”, RFC 1890, IETF, January 1996.

39. B. Flinchbaugh, M. Zhou and R. Talluri, “How MPEG-4 Trade-offs Affect Design”, *EE Times*, November 12, 2001. (http://www.eetimes.com/story/OEG20011112S0047)

40. L. Chiariglione, ed., “MPEG-1 - Coding of Moving Pictures and Associated Audio for Digitial Storage Media at up to about 1.5 Mbit/s”, ISO/IEC JTC1/SC29/WG11, June 1996. (http://mpeg.telecomitalialab.com/standards/mpeg-1/mpeg-1.htm)

41. L. Chiariglione, ed., “MPEG-2 - Generic Coding of Moving Pictures and Associated Audio Information”, ISO/IEC JTC1/SC29/WG11, October 2000. (http://mpeg.telecomitalialab.com/standards/mpeg-2/mpeg-2.htm)

42. S.T. Worrall, A.H. Sadka, P. Sweeney, and A.M. Kondoz, “Optimal Packetisation of MPEG-4 Using RTP Over Mobile Networks”, *IEE Proceedings - Communications*, vol. 148, no. 4, August 2001, pp. 197-201.

43. J. Mogul and S. Deering, “Path MTU Discovery”, RFC 1063, IETF, November 1990.

44. J. Postel, “Internet Control Message Protocol”, RFC 792, IETF, September 1981.

45. M. Allman, S. Floyd and C. Partridge, “Increasing TCP’s Initial Window”, RFC 2414, IETF,

September 1998.

46. T. Dierks and C. Allen, “The TLS Protocol Version 1.0”, RFC 2246, IETF, January 1999.

47. M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones, “SOCKS Protocol Version 5”, RFC 1928, IETF, March 1996.

48. T. Zhang, P. Agrawal and J.C. Chen, “IP-Based Base Stations and Soft Handoff in All-IP Wireless Networks”, *IEEE Personal Communications*, vol. 8, no.5, October 2001, pp. 24-30.

49. “Internet Traffic Report”, (http://www.internettrafficreport.com)

A List of Abbreviations

3G	Third Generation
ACK	Acknowledgement
AIMD	Additive Increase Multiplicative Decrease
AVO	Audiovisual Object
BER	Bit Error Rate
CDMA	Code Division Multiple Access
CIF	Common Interchange Format
FEC	Forward Error Correction
FTP	File Transfer Protocol
HEC	Header Extension Code
ICMP	Internet Control Message Protocol
IP	Internet Protocol
ITU	International Telecommunication Union
kb	Kilobit
kB	Kilobyte
kbps	Kilobits per second
MBA	Macroblock Address
MBM	Motion Boundary Marker
MPEG	Moving Picture Experts Group
MTU	Maximum Transfer Unit
QCIF	Quarter Common Interchange Format

QoS	Quality of Service
QP	Quantization Parameter
RTP	Real Time Protocol
RTT	Round Trip Time
RVLC	Reversible Variable Length Codes
SACK	Selective Acknowledgement
SDTFP	Split-Domain TCP-Friendly Protocol
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
W-CDMA	Wideband Code Division Multiple Access
WAN	Wide Area Network
WAP	Wireless Application Protocol