

TCP friendly protocols for media streams over heterogeneous wired–wireless networks

Hala ElAarag*, Andrew Moeding, Chris Hogg

Mathematics and Computer Science Department, Stetson University, 421 N. Woodland Boulevard, Unit 8332, DeLand, FL 32720, United States

Received 10 February 2008; accepted 11 February 2008

Available online 26 February 2008

Abstract

The growing need for Internet friendly streaming protocols prompted us to develop IFTP (Internet Friendly Transport Protocol). IFTP is a protocol with an inherent rate-based flow control mechanism designed to emulate TCP's window-based flow control mechanism. In this paper, we present two proposals to improve the performance of IFTP in networks with wireless links. The first is IFTP-W, which specifies an error-sensitive section. Bit errors not occurring in this section are ignored. The second is IFTPV, which distinguishes between losses due to congestion and those due to the wireless link. In this paper, we present the performance of IFTP-W with varying error-sensitive section lengths on networks with varying link qualities. We also analyze the performance of IFTPV in extensive scenarios with various network conditions. The results show that both proposals maintain the TCP-friendliness and fairness properties, but perform better than IFTP on wireless networks.

© 2008 Elsevier B.V. All rights reserved.

Keywords: Transport protocols; TCP-friendly; Fairness; Wireless networks; Multimedia applications

1. Introduction

The growth in the popularity of multimedia applications and real-time games is leading to an increased usage of continuous media protocols which lack end-to-end congestion control mechanisms. The lack of flow control inherent to these protocols can lead to a monopolization of the available bandwidth as these protocols are not concerned with reliable data delivery. This problem, called the Internet friendly problem or the TCP-friendly problem has not fully manifested itself, but it poses as a potentially substantial problem that should be addressed before requiring immediate attention. Not only could this problem cause inequitable bandwidth distribution, but it also threatens congestion collapse [1].

There are several solutions to counteracting the effect of flows unresponsive to path congestion, including active queue management and flow-based packet scheduling at the router level. However, if each flow is aware of network conditions, such monitoring and management of each flow at the router level is not required. Several methods of informing flows of imminent packet loss such as Explicit Congestion Notification (ECN) have been developed, but they are only effective for flows that already contain some form of congestion control. The Internet Friendly Transport-level Protocol (IFTP) [2], is one solution to the Internet friendly problem. IFTP combines the streaming qualities of UDP and the congestion control properties of TCP to create a continuous media protocol capable of responding appropriately to congestion.

As the usage of wireless networks increases, the necessity of developing a streaming protocol capable of functioning over a wireless medium also increases. Wireless networks have the eminent problems of high bit error rate (BER) and small bandwidth. These issues adversely affect the performance of protocols that do not account for them. For

* Corresponding author. Tel.: +1 386 822 7554.

E-mail addresses: helaarag@stetson.edu (H. ElAarag), amoeding@stetson.edu (A. Moeding), chogg@stetson.edu (C. Hogg).

example, as most versions of TCP interpret packet drops as a signal of congestion, a packet drop caused by bit corruption is mistakenly identified as an indication of network congestion. On networks with high BER, this can easily cripple the connection.

In this paper, we propose transport protocols for media streaming that is aware of the TCP-friendly problem and the limitations of wireless networks. We modified the TCP-friendly protocol (IFTP) that we previously developed [2] to enhance its performance on wireless networks. Like TCP, IFTP interprets packet loss as network congestion and reacts to it by decreasing its sending rate. This negatively and unnecessarily affects the performance of IFTP as packet loss in wireless networks is mainly related to high BER rather than congestion. To improve the performance of IFTP, we first propose IFTP-Wireless or IFTP-W – a protocol that distinguishes between error-sensitive and error-insensitive data. If an IFTP-W packet is corrupted due to the high BER of the wireless medium, it is dropped only if the error occurred in error-sensitive data. IFTP-W adjusts its sending rate according to IFTP's congestion avoidance protocol if error-sensitive data is corrupted. Our simulation results show that IFTP-W performs better than IFTP on wireless networks and maintains IFTP's TCP-friendliness property. We also study the effect of error-sensitive length of IFTP-W. Second, we present another transport protocol, IFTPV, to solve the TCP-friendly problem on wireless networks. This protocol attempts to differentiate between errors due to congestions and those due to the wireless link. We study the performance of IFTPV through extensive simulations in various network scenarios.

The rest of the paper is organized as follows: Section 2 presents the background for the paper, while Section 3 presents previous research in this area. In Section 4, we briefly explain IFTP protocol. Section 5 explains our discrete event network simulator, network topology and performance metrics. Section 6 shows the performance of IFTP on wireless networks which illustrates the need for protocols that are designed with wireless problems in mind. Section 7, introduces the IFTP-W protocol, while Section 8 demonstrates its performance on wireless networks through simulation using several performance metrics. Section 9 studies the effect of varying the length of IFTP-W's error-sensitive part. Section 10, introduces IFTPV and its performance evaluation. Finally Section 11 concludes the paper and provides suggestions for future work.

2. Background

In this section we give a brief overview of UDP and TCP. UDP is a connectionless unreliable protocol designed for multimedia and other streaming applications. UDP transmits packets at a set rate. The rate is realized by calculating an appropriate interpacket gap. The interpacket gap is generally simulated via an exponential distribution

to model the actual performance of UDP. Thus, our UDP server transmits packets at an average rate with an exponentially distributed interpacket gap and does not process any incoming packets as it does not utilize any form of ACK (acknowledgement packet) scheme [3].

In contrast, TCP is a reliable protocol, ensuring delivery of packets. For each transmitted packet, the server awaits an ACK, signifying reception at the client endpoint. The TCP sending scheme is based on a form of sliding window algorithm [4]. TCP's scheduling scheme is somewhat more complicated than that of the relatively simple sliding window algorithm, however. TCP uses three phases of transmission: slow start, congestion avoidance, and maximum window. During slow start, the transmission window is doubled in every window; during congestion avoidance, the window is increased linearly; and in the maximum window phase, the window size remains constant. There are several implementations of TCP. For example, TCP Reno is a TCP standard that contains enhancements such as fast retransmit and fast recovery. Fast retransmit and fast recovery are described in detail in [5].

3. Previous research

Past research has included suggestions for both TCP-friendly congestion control mechanisms and improved performance over wireless networks as independent topics. A good survey of TCP-friendly protocols can be found in [6]. One example protocol, TCP Friendly Rate Control (TFRC) [7] attempts to approximate the bandwidth usage of TCP, though it strives for a smoother transmission rate, and is thus less responsive to changes in network conditions than IFTP.

Several suggestions have been proposed to improve the performance of TCP on wireless networks and are described in [8]. Another recent proposal for an improved reliable data transport protocol over wireless networks is the TCP VenO protocol, which monitors network congestion levels in order to attempt to distinguish between packets dropped due to congestion and packets dropped due to random bit error [9].

ECN is a promising router-based protocol that delivers unambiguous signals of congestion by algorithmically marking packets when the buffer size exceeds a certain threshold [10]. ECN requires a modified router implementation and a modification at both endpoints and it is not applicable to protocols that do not respond to congestion. Larzon et al. [11] proposed an improvement for UDP to boost performance over links with high BER. This improvement for UDP is designed to counteract the negative effects of wireless networks, but it does not introduce any form of congestion control to UDP.

Another area of research concerned with network fairness and congestion control is focused on developing router based algorithms to enforce congestion control. Various schemes, such as Active Queue Management (AQM) have been designed to enforce fair resource alloca-

tion policies during periods of congestion. The adoption of these techniques requires a substantial investment for the wide-scale deployment that is required to achieve effectiveness. Chandrayana and Kalyanaraman [12] have proposed modifying only the routers at the edge of a network. Router-based flow management can protect congestion conscious flows from being deprived of equal network resources by selfish streams.

4. IFTP

In this section, we briefly explain the IFTP protocol as it is the base of our proposal protocols. More details can be found in [2]. IFTP is an Internet protocol that was designed to provide an Internet friendly transmission protocol for multimedia applications. By Internet friendly we mean that it scales its transmission rate in a fashion similar to TCP as network congestion is introduced. Such a protocol was needed because most multimedia applications are transmitted using UDP, which is considered a non-TCP-friendly protocol. In addition to its TCP-friendliness property, IFTP has a very useful feature which is its ability to keep a low jitter and hence improves the performance of multimedia applications. IFTP regulates packet flow by manipulation of the interpacket gap in response to network congestion. The appropriate interpacket gap is determined by acknowledgements (ACKs) which are sent back to the sender for the sole purpose of estimating the round trip time over the link. The estimated times are then used to emulate the network usage of Reno TCP, the most common TCP implementation, in each of its three phases. By emulating the slow start, congestion avoidance, and maximum window phases, as well as responding appropriately to packet timeout and duplicate ACKs, IFTP performs similarly to TCP under various network conditions.

Assume that IFTP sends n packets in the i^{th} period. It then calculates the interpacket gap for $i+1^{\text{th}}$ period G_{i+1} , which is the time to wait in between sending the last bit of one packet and the first bit of the next packet as follows:

Slow start phase:

$$G_{i+1} = \frac{R}{2n} - \tau \quad (1)$$

Congestion avoidance phase:

$$G_{i+1} = \frac{R}{n+1} - \tau \quad (2)$$

Packet loss:

$$G_{i+1} = \frac{2R}{n} - \tau \quad (3)$$

where,

R is the smoothed round trip time and

τ is the time it takes to send a single packet in seconds.

Note that IFTP assumes a WAN Internet environment and hence $R \gg$ TCP window.

TCP was not designed with wireless problems in mind. In [13], we study the performance of Tahoe, Reno, New-Reno and Sack TCP in wireless networks. TCP interpret the cause of packet drops – as indicated by either retransmission timer expiration or the receipt of multiple duplicate ACKs – as network congestion, and thus reduce their transmission flow. However, due to the relatively high BER of most wireless mediums, packet drops often are not the result of network congestion. Thus, the flow of data is unnecessarily restricted when packet loss occurs due to bit corruption during transmission. This misinterpretation of packet loss unnecessarily transmogrifies the data stream into a data trickle. Because IFTP emulates TCP, it also experiences some of the same performance flaws as TCP when one of the links in the transmission path is a wireless medium.

5. Simulator, topology and metrics

In the next section, we present the performance of IFTP over wireless and follow it with IFTP-W and IFTPV. Since all the performance measures reported in this paper are done through simulations, we first present our network simulator, the network topology and the performance metrics used in the rest of the paper.

5.1. The network simulator

We developed a discrete event network simulator in C#. Using C#, it was natural to adopt an object-oriented design. For each node in the network topology, there exists a corresponding node object. All nodes are instances of a subclass of the node abstract class. The node superclass is extended by the Router, Server, and Client classes. Server instances are designed to simulate a server, sending packets as specified by the protocol they are running. The main functionality provided by the client is the transmission of an ACK if a received packet is from a protocol expecting an ACK. Routers simply forward packets from the server to the client and ACKs from the client to the server. Upon configuration setup, all routers are given perfect knowledge of nodes which are connected directly or transitively. Each connected node is stored as an entry in a routing table that also holds a number for the interface number to which packets should be forwarded to reach the correct destination.

Nodes are connected by physical connections, which are responsible for managing everything that a physical interface card would manage. Physical connections also calculate and simulate delay for packets as they are transmitted. Virtual connections on the transport layer are managed by instances of the virtual connection class. Each virtual connection has a reference to the physical connection through which it operates and a protocol instance to handle the transmission and reception of packets. The protocol class is also abstract, and is extended by classes to simulate protocols such as UDP, TCP, IFTP, etc. Each

protocol creates and determines when to transmit each packet. This can be time based, or in response to ACK reception.

The message is also encapsulated in an object. It manages its type, size, headers, and corrupted bits. The message object is used for both packets and ACKs. The header is changed to indicate the protocol and whether the message is a data or ACK packet.

The scheduler stores all events that have yet to be invoked, calling each in order based on its specified execution time. The event scheduler is implemented as a singleton – a design pattern for objects with only one instance per process. When a singleton is created in C#, a static member variable is set, pointing to the one instance of the class. The use of a public singleton allows for all objects to have access to the scheduler without passing it as an object to each function called.

An additional feature aiding in program debugging and verification was a customized output interface. We used simple C# GUI controls to enable or disable the output generated by various types of objects such as nodes or protocols. We also implemented a method of filtering output based on the name of the node originating it.

5.2. The network topology

We chose the typical router bottleneck dumbbell topology. In this topology, there are multiple servers with their corresponding clients sharing the bottleneck router. Fig. 1 illustrates UDP, TCP and IFTP sources sharing one bottleneck router. All sources transfer simultaneously. The router has an infinite buffer to ensure that all packet drops are caused by the wireless link. The router uses FCFS scheduling and drop-tail queuing. Each wired link has a delay of 100 ms and each wireless link has a delay of 10 ms, unless otherwise indicated. For TCP, IFTP, IFTP-W and IFTPV, data packets are 1024 bytes, acknowledgement packets are 40 bytes, the slow start threshold is 32, maximum window size is 50, and three duplicate ACKs initiate fast recovery. These parameters are typical for a TCP connection and thus chosen to be the same for IFTP, IFTP-W, and IFTPV to allow for fair comparisons. UDP packets are also 1024 bytes in length and are sent at an average rate of 1 Mbps with an exponentially distributed interpacket gap. The links between the sources and the router are wired and have a bandwidth of 10 Mbps, while the links between the router and the cor-

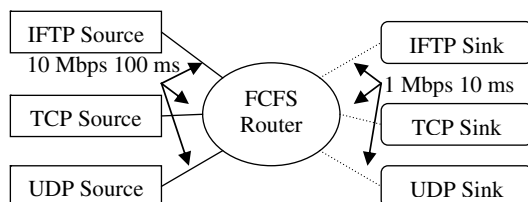


Fig. 1. A visualization of our network topology.

responding receivers are wireless with a bandwidth of 1 Mbit per second. The BER value represents the ratio of bits experiencing corruption. Thus, a BER of 10^{-3} is equivalent to 1 bit corruption out of 1000 bits, and 10^{-7} specifies 1 bit error out of 10,000,000 bits. Our simulator models the occurrence of bit errors with an exponential distribution.

5.3. Performance metrics

Several performance metrics were used, namely, throughput, goodput, delay, and delay jitter. They are defined as follows:

- *Throughput* is defined as number of bits per second received at each receiver.
- *Goodput* is defined as the ratio between the number of packets received at the receiver to the number of packets sent. Higher goodput values coincide with more efficient network resource utilization. A goodput value of 1 indicates that no packets were lost during a connection.
- *Delay* is defined as the length of time a packet is in transit from the time it originates at the server to the time it is received at the data sink.
- *Delay jitter* is the variance of the packet delay.

6. IFTP over wireless networks

We studied the performance of IFTP on wireless networks through simulation. Figs. 2–5 show our simulation results for the four mentioned performance measures.

As expected, UDP reacts less severely to the change of BER than either IFTP or TCP as shown in Fig. 2. This follows expectations because UDP is devoid of flow control, and thus any throughput reduction when encountering dropped packets is caused solely by the loss of the segments from the network. The sending rate is maintained, even after packet drops occur. This is not to suggest that the dropped packets will not significantly affect the experience of a UDP stream on the application layer, but that the average rate of data reception at the UDP sink over the

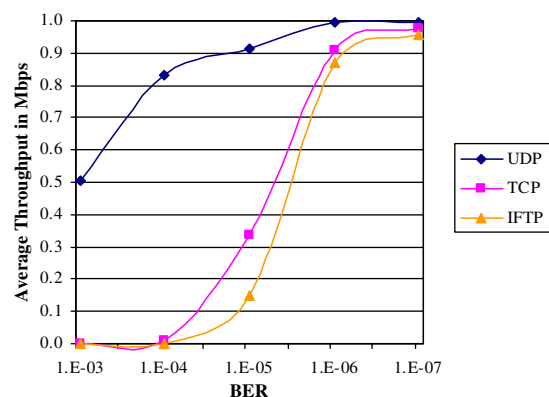


Fig. 2. Throughput as a function of BER.

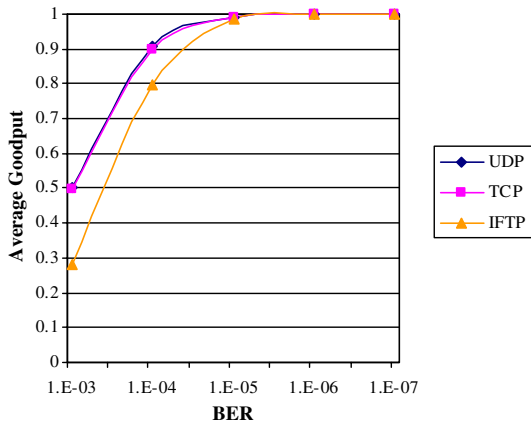


Fig. 3. Goodput as a function of BER.

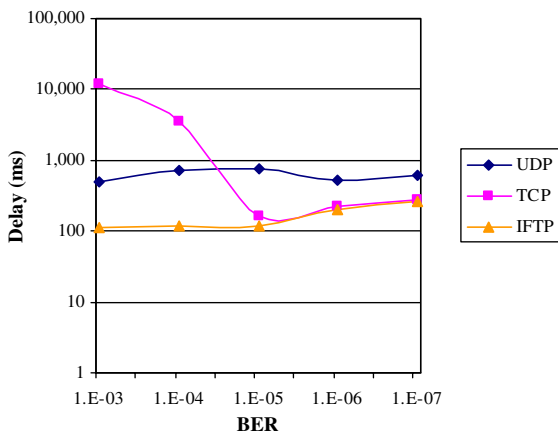


Fig. 4. Delay as a function of BER.

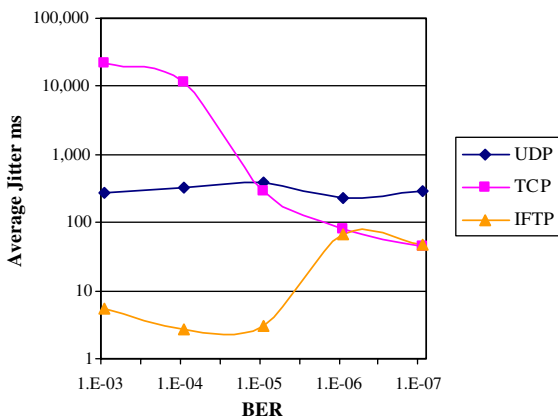


Fig. 5. Delay jitter as a function of BER.

lifespan of the connection will generally not be crippled until BER nears 10^{-4} .

Unlike UDP, TCP and IFTP react severely to higher BER. Not only do the lost packets detract from throughput, but they also result in reduced transmission rates due to interpretation of the lost packets as network congestion. Both TCP and IFTP have similar performance under varying BER, as would be expected because of IFTP's imi-

tation of TCP. In cases of congestion, this imitation is the essential property of IFTP. However, when the packets are dropped due to bit error, throughput reduction is unnecessary.

All three protocols experience similar goodput as illustrated in Fig. 3. As goodput is a percentage of packets received, it maintains approximately the same relation to BER for all protocols.

The delay and delay jitter values presented in Figs. 4 and 5 illustrate the large delay of TCP over wireless links with high BER. The delay is calculated from the time the packet is originally scheduled; thus, packets that are retransmitted several times have a very high delay. The delay jitter is also very high as some packets are delivered on the first transmission. This high delay and delay jitter makes TCP unusable for transmission of real-time and streaming data, especially over wireless links. IFTP and UDP have a more stable delay and delay jitter because they do not retransmit packets. UDP's higher delay is caused partially by sending bursts that cause packets to back up in the pipe. IFTP's rate is more regular as it is controlled by a programmatically modified interpacket gap.

7. IFTP-W

IFTP-W [14,15] is an end-to-end congestion control protocol we designed to solve IFTP's problems on wireless networks illustrated in Section 6. IFTP-W is a TCP-friendly protocol for media streams that allows for applications to choose a section of a packet to be verified by a checksum, with the remaining portion deemed error-insensitive and not checked for corrupted bits. One corrupted bit in a video or audio stream may cause a discolored pixel or distorted millisecond of audio; however, in many codecs, receiving the partially damaged packet results in better overall performance than dropping the packet. IFTP-W allows a greater percentage of packets to be transmitted, resulting in a higher goodput and throughput and hence improved performance over wireless networks suffering high bit error rates.

To improve IFTP's performance during high-loss conditions, we divide IFTP packets into error-sensitive and error-insensitive sections. When a bit error occurs in an error-insensitive section of the packet, the packet is not dropped. This technique has also proved useful for UDP [11]. However, the benefits experienced by UDP are magnified for a protocol such as IFTP that uses packet drops as a signal of congestion.

The preservation of these packets can be implemented by modifying the device driver at the receiver to ignore CRC errors of packets carrying IFTP-W traffic. Packets can then be dropped at the transport level if there is an error in the error-sensitive section of the packet. Otherwise, potentially valid IFTP-packets could be dropped on the link layer, before arriving at the transport layer.

Fig. 6 shows the IFTP packet header format. The packet header format for an IFTP-W packet, shown in Fig. 7, is

Source Port	Destination Port
Packet Num	Checksum
Timestamp	Length
Payload	

Fig. 6. IFTP packet format.

Source Port	Destination Port
Packet Num	Checksum
Timestamp	Coverage
Payload	

Fig. 7. IFTP-W packet format.

similar to the format of the IFTP header. The difference with IFTP-W is that the length field is replaced with a coverage field, which designates the length of the error-sensitive section of the packet. Also, the checksum is only used for the error-sensitive section of the packet. The packet number and timestamp fields are copied into the ACK at the receiver for calculation of RTT by the sender. The ACK format is identical to the data packet format, except that it does not have a payload. Also, the IFTP-W ACK does not have a coverage field as the header is always considered error-sensitive.

We also modified the sending algorithm for IFTP from the definition in our original paper to enhance its throughput. The *transmit_window* function now sends a packet after the interpacket gap has elapsed if $cwnd + una < next$, where *cwnd* is the size of the congestion window, *una* is the sequence number of first unacknowledged packet, and *next* is the number of the next packet to be transmitted. In the previous implementation, the IFTP server waited for all ACKs of one window to arrive before beginning transmission of the next window.

To ensure that IFTP-W maintains the TCP-friendliness present in IFTP, we ran a simulation of simultaneous UDP, TCP, and IFTP-W flows running through a bottleneck router with a finite buffer. The results shown in Fig. 8 demonstrate that the TCP-friendliness is preserved. As the sending rate of UDP is increased, both IFTP-W

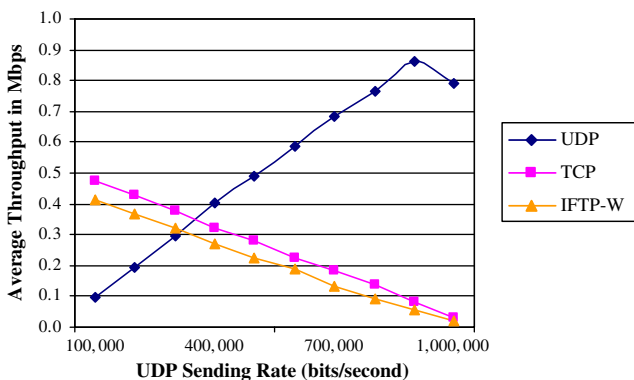


Fig. 8. IFTP-W's TCP-friendliness.

and TCP detect the increasing congestion and decrease their sending rate. Once the UDP stream reaches a sending rate of 1 Mbps, its measured throughput is reduced because the router's buffer becomes full and the router is forced to drop packets.

8. IFTP-W over wireless networks

We simulated the performance of IFTP-W in comparison to UDP, TCP and IFTP using the same simulation configuration as in Section 5, this time adding an IFTP-W source operating simultaneously with the other three sources. All IFTP-W packets had an error-sensitive header of 40 bytes and an error-insensitive payload. Results are averaged from 10 runs of simulated connections running for approximately 2 min for each value of BER.

8.1. Throughput

The results for the throughput of the protocol with congestion control mechanisms are in some cases affected by one simulated connection that experienced an early packet drop. If one of these connections experiences a packet drop during the slow start phase, the slow start phase is prematurely exited as a result of the lowered slow start threshold. An early exit causes a postponed arrival at the maximum window phase. As these results are from simulated connections running for 2 min, an early packet loss has a substantial effect on the throughput of the entire period which may not be reflected in the steady state throughput. A longer simulation period could potentially counteract the effect of an early packet drop, but the slow start threshold would never recover. Regardless of these effects upon the results, the results appear accurate in indicating the approximate effects of IFTP-W.

The throughput of our scenario testing IFTP-W is shown in Fig. 9. These results show that IFTP-W does offer a significant throughput improvement over IFTP. With the compound benefit of additional delivered packets and thereby preserved throughput, IFTP experiences less unnecessary reduction of throughput from high BER. Even

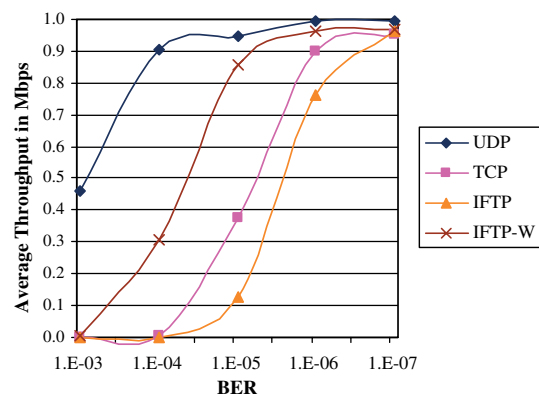


Fig. 9. Throughput as a function of BER.

when BER is 10^{-5} – representing 1 bit error for every 100,000 bits – IFTP-W retains an average throughput of more than 0.8 Mbps, instead of the throughput of IFTP of less than 0.2 Mbps.

8.2. Goodput

As UDP, TCP, and IFTP have the same packet size in our simulations – and as all packet drops are caused by the wireless link and are not the result of congestion – it is expected that they would have the same goodput. These three protocols do have approximately equal goodput for each value of BER. IFTP-W, however, has a greater goodput because a smaller percentage of IFTP-W packets are dropped. The results are illustrated in Fig. 10. A greater goodput indicates a better utilization of the network; as a greater percentage of transmitted packets are delivered, fewer network resources are wasted by transmitting packets that are eventually dropped before reaching the destination.

So far we have introduced the first modification of our TCP-friendly protocol IFTP to provide improved performance over links prone to high BER. This modification, designated as IFTP-W, partitions packets into error-sensitive and error-insensitive sections. Any bit errors in the error-insensitive section are acceptable. By only discarding packets with errors in the error-sensitive section, both goodput and throughput are increased. Throughput is the primary beneficiary of these changes as there are fewer packet drops falsely interpreted as signals of congestion. The results of our simulations indicate that these modifications do provide the expected improvements. A similar division of packets into error-sensitive and error-insensitive sections by other TCP-friendly protocols should lend similar results in boosting performance where high BER is involved. The length of the error-sensitive section is an important parameter in IFTP-W. In the next section, the effect of varying the error-sensitive section length is carefully studied in order to provide guidance in the selection of error-sensitive section lengths.

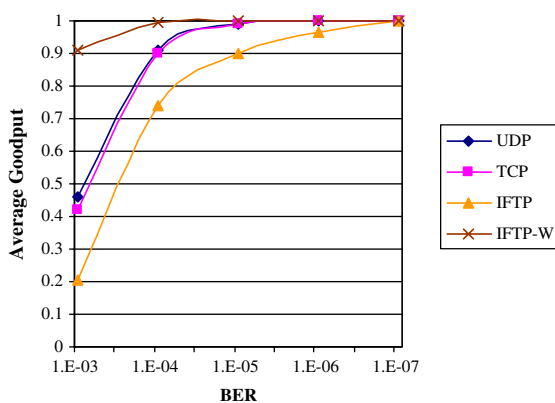


Fig. 10. Goodput as a function of BER.

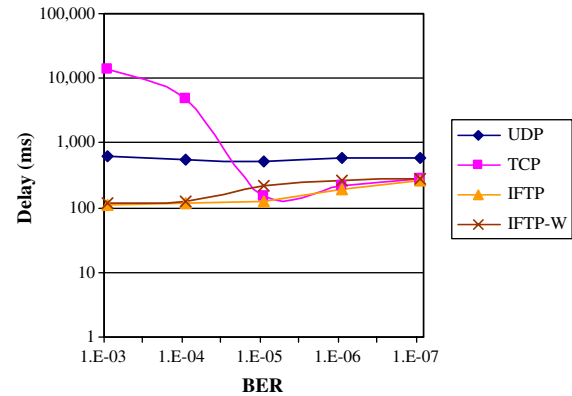


Fig. 11. Delay as a function of BER.

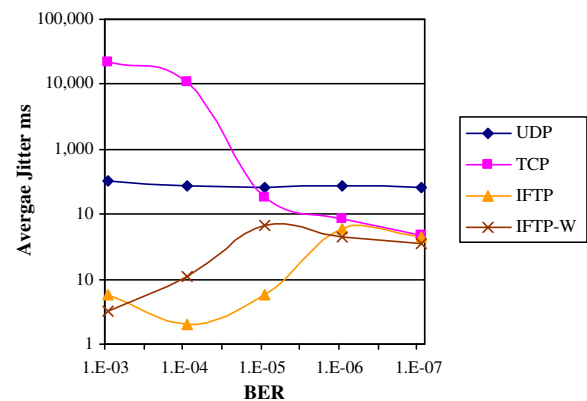


Fig. 12. Delay Jitter as a function of BER.

8.3. Delay and delay jitter

TCP, IFTP, and UDP have similar delay and delay jitter in this simulation, illustrated in Figs. 11 and 12, as in the simulation shown in Section 6, Figs. 4 and 5, respectively. Explanation of their behavior is provided in Section 6. IFTP-W has similar delay and delay jitter to IFTP, though in some cases both values are slightly higher, as a result of IFTP-W's greater throughput.

9. Effect of error-sensitive length of IFTP-W

To determine the performance of video and audio applications using IFTP-W on wireless links we ran several simulations. Our network topology consisted of three data sources and three data sinks connected through a FCFS bottleneck router as illustrated in Fig. 1. Each source operated on a different protocol – one simulated a UDP stream, one simulated a TCP connection, and the third simulated an IFTP-W stream. The default packet size for packets containing streaming video data over EDCF on 802.11e networks is 1464 bytes. For audio packets, the default size is 92 bytes [16].

We modified our routers from the standard implementation that drops packets if the checksum does not compute

correctly. Dropping all packets with a checksum error could potentially inhibit the delivery of packets that only have an error in the error-insensitive section by dropping them before they reach the network layer. Thus, we only drop packets if the error occurs in the error-sensitive section. As a general guideline, video codecs require a packet loss rate of less than 2% for acceptable performance and audio codecs require a packet loss rate of approximately 1% [17,18].

For BER values of 10^{-3} – 10^{-7} , we ran simulations with packet sizes for both video and audio packets with a varying error-sensitive section. The video packets had a size of 1464 bytes with an error-sensitive section varying from 40 to 1464 bytes, and the audio packets has a size of 92 bytes with an error-sensitive section ranging from 40 to 92 bytes. The error-sensitive section in this case is header-inclusive. Each run is a simulation of a 120-s connection. All results represent the average of multiple runs.

9.1. Throughput

The throughput results in Figs. 13 and 17 show the increased performance with IFTP-W over standard IFTP for both video and audio packets, respectively. A simulation with IFTP is equivalent to an IFTP-W simulation run with an error-sensitive section length equal to the packet size. With IFTP-W, the client will receive some corrupted bits, but the codecs should be able to better handle these errors than if the entire packet were lost.

Simulations with BER values of 10^{-4} or more, while improved by IFTP-W, indicate that links with this BER are still too error prone to be practical for streaming transmission with IFTP-W.

The throughput for the audio stream is significantly less than the stream of video data. This is due to the fact that the audio packets consist of less data than video packets. There is still a noticeable increase in throughput as the size of the error-sensitive section is decreased.

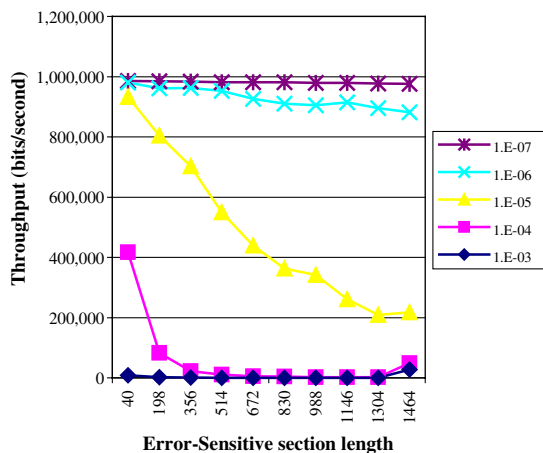


Fig. 13. Throughput of video packets as a function of BER and error-sensitive section length.

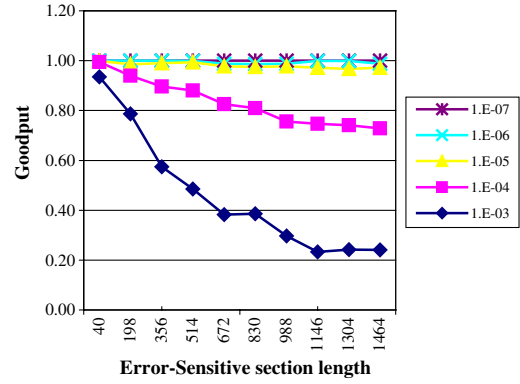


Fig. 14. Goodput of video packets as a function of BER and error-sensitive section length.

9.2. Goodput

The goodput results displayed in Figs. 14 and 18 are also indicative of increased performance using IFTP-W. The increased goodput allows streaming transmissions to be of sufficient quality. It is essential, however for the throughput to also be sufficiently high in order for the goodput to be relevant. In the case of video packets with a BER of 10^{-4} , the goodput appears high enough, for some values of the error-sensitive section length, to facilitate usability. However, the throughput is too low reduced for usability to be realized.

9.3. Delay and delay jitter

The delay and delay jitter values of video packets increase as the BER decreases in Figs. 15 and 16 because packets tend to back up at the router as more are sent in close proximity. When BER is higher, packets are dropped more frequently, reducing the rate of transmission. The rate reduction results in packets experiencing zero delay at routers and thus achieving low delay and delay jitter. As BER decreases, packets are sent in closer succession

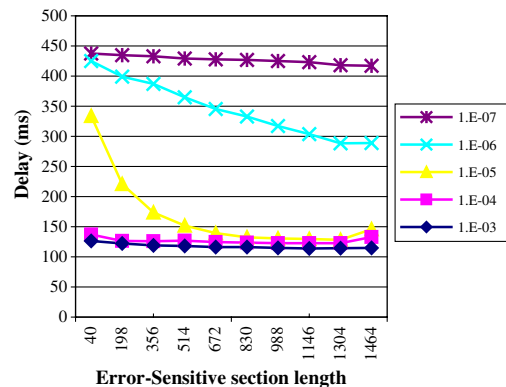


Fig. 15. Delay of video packets as a function of BER and error-sensitive section length.

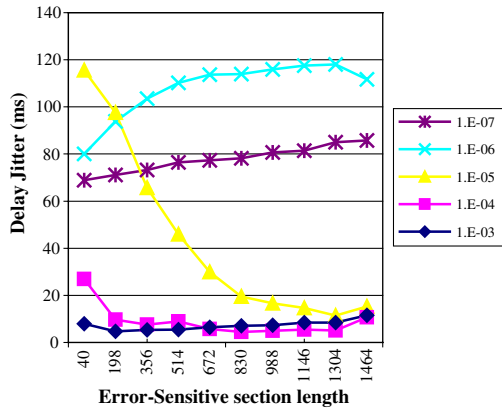


Fig. 16. Delay jitter of video packets as a function of BER and error-sensitive section length.

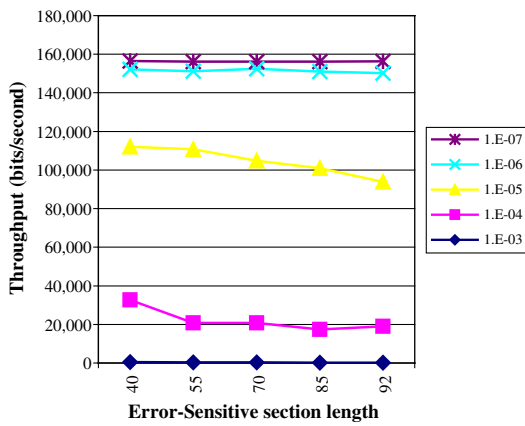


Fig. 17. Throughput of audio packets as a function of BER and error-sensitive section length.

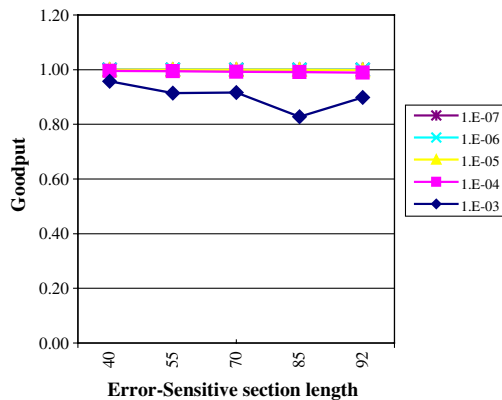


Fig. 18. Goodput of audio packets as a function of BER and error-sensitive section length.

and begin to experience some delay imposed by router queuing. This only occurs for video packets as the audio packets require less time for transmission. Audio packets on an IFTP-W stream with this network topology and con-

figuration do not experience any queuing delay as the stream reaches maximum window phase without achieving a small enough interpacket gap to cause a backup. Thus, the delay and delay jitter values for our audio stream in , Figs. 19 and 20 are essentially constant for all BER values.

The results of our simulation provide approximations of some of the performance statistics to be expected from running an IFTP-W stream over wireless mediums in varying conditions. The results show that IFTP-W does remedy some of the issues experienced by protocols with inherent congestion control mechanisms on bit error prone links. IFTP-W delivers more packets, resulting in increased throughput and goodput. Delivered packets have the potential for embedded corruption and thus this method would not be appropriate for reliable transport protocols, such as TCP. However, results indicate a large measure of improvement for streaming protocols capable of tolerating moderate bit corruption. This is especially applicable for streaming protocols, such as IFTP, that implement a congestion control mechanism.

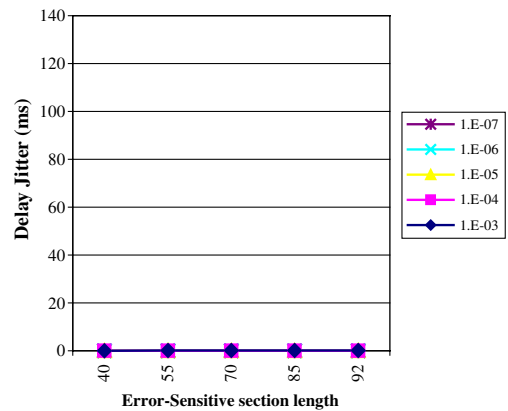


Fig. 19. Delay of audio packets as a function of BER and error-sensitive section length.

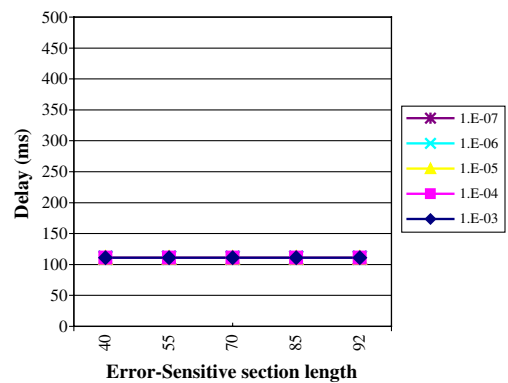


Fig. 20. Delay jitter of audio packets as a function of BER and error-sensitive section length.

10. IFTPV

10.1. Rationale

Recently, there are four proposed TCP protocols optimized for wireless networks and based on the end-to-end paradigm. They are JTCP [19] and TCP-Jersey [20], TCP Westwood [21] and TCP-Veno [9]. JTCP uses the inter-arrival jitter and the jitter ratio to infer network congestion. While TCP-Jersey estimates the congestion window after the loss has occurred using the available bandwidth estimator algorithm. TCP Westwood frees TCP from the traditional AIMD algorithm and rather statistically estimates the bandwidth with each packet sent. TCP-Veno is an improvement on TCP-Reno. It calculates the rate as the ratio of the congestion window to the RTT. It uses two main parameters. One is the expected rate which uses the best RTT, and the other is the actual rate which uses the last measured RTT. Todorovic and Benitez [22] compare these four proposals and conclude that there is no one solution that showed superiority in all scenarios. Their results are not conclusive as they mention that the choice of the different parameters could provide different results.

In Section 8, we showed that IFTP-W performs better than IFTP on wireless networks. However, it was still, like IFTP, faithful to TCP-Reno to maintain its TCP-friendliness property. TCP-Reno is inherently inefficient on wireless networks as it interprets every packet loss as congestion. The recent proposals of TCP protocols optimized for wireless networks, as explained above, seemed encouraging. Although the techniques are different, they all attempt to differentiate between errors due to congestions and those due to the wireless link. It appears to be logical to have IFTP faithful to one of these proposals instead of TCP-Reno to improve its performance on wireless networks. However, we do not want to sacrifice the IFTP's fairness and TCP friendliness properties. Kaneko and Kato [23] proved that TCP Westwood's friendliness with TCP Reno is deteriorated according to network situations such as the buffer size of a bottleneck link router. TCP Jersey requires ECN capable routers in the network [20]. The authors of JTCP [19] mention that its fairness is not the best. However, TCP Veno [9] is the most compatible to TCP Reno which the original IFTP protocol is based on. Also Zhou and Fu [24] demonstrated that TCP Veno is fair to TCP Reno.

10.2. The IFTPV protocol

In this section, we propose IFTPV which enhances the performance of IFTP protocol by being faithful to TCP Enhanced Veno [24] which is a modification of TCP-Veno [9]. IFTPV maintains two variables; the backlog, B , and the congestion rate, con_r . The first variable, B , estimates the numbers of packets in a connection until a packet loss occurs i.e. tracks the number of packets suspected to wait in a queue. IFTPV determines this by keeping track of the round trip times and how long a packet would take

to be acknowledged. When it seems that a given window took longer to transmit than its size indicated then a packet must have spent time in a router's queue. Thus B is calculated according to Eq. (4), where RTT_{base} is the minimum of measured round trip times, RTT_{actual} is the measured smoothed RTT and n is the number of packets transmitted in a round trip.

$$B = \left(\frac{n}{RTT_{base}} - \frac{n}{RTT_{actual}} \right) \times RTT_{base} \quad (4)$$

If the estimated backlog is greater than or equal to a certain threshold, β , then a packet loss indicated by a duplicate ACK is due to congestion and hence the number of packets transmitted in the next period, n , is halved. Otherwise the network is assumed to be in a random loss state rather than a congestion state. Fu et al.'s [9] experiments indicate that a good value for β is 3. The second maintained variable by IFTPV is the congestion loss rate, con_lr , which determines if during the random loss state the network is congestive. The congestion loss rate is calculated by keeping track of the number of packets lost to congestion (C) in between three consecutive packet losses due to link error, and dividing it by the time between their associated link error losses. For example, consider that the three consecutive random losses due to link error occurred at times T_1 , T_2 and T_3 , respectively. Now assume that between T_1 and T_2 , the number of packet lost due to congestion is C_{prev} and that between T_2 and T_3 , this number is equal to $C_{current}$. con_lr_{prev} and $con_lr_{current}$ are defined as shown in Eqs. (5) and (6).

$$con_lr_{prev} = \frac{C_{prev}}{T_2 - T_1} \quad (5)$$

$$con_lr_{current} = \frac{C_{current}}{T_3 - T_2} \quad (6)$$

When a random loss occurs these values are updated.

IFTPV is interested in determining whether during the random loss state ($B < \beta$) the congestion loss rate is increasing or decreasing. If it is increasing IFTPV protocol decreases its number of transmitted packets by one-fifth in an attempt to avoid congestion, however if the rate is not increasing this number would stay unchanged. IFTPV still treated a timeout as a sign of extreme network condition and so acted the same way as IFTP, which is the same as TCP, by halving the slow start threshold and entering the slow start phase with a window equal to 1.

The pseudo code of how IFTPV alters the number of packets transmitted in a given period when a duplicate ACK is received is shown below:

```

if ( $B < \beta$  &&  $con\_lr_{current} \leq con\_lr_{prev}$ ) {
     $n = n$ ;
} else if ( $B < \beta$  &&  $con\_lr_{current} > con\_lr_{prev}$ ) {
     $n = n * (4/5)$ ;
} else if ( $B \geq \beta$ ) {
     $n = n * (1/2)$ 
}

```

where,

- B is the estimated number of backlogged packets
- β is a threshold set to be equal to 3
- $con_lr_{current}$ is the number of packets lost due to congestion per unit time between the last two packets lost due to link error
- con_lr_{prev} is the number of packets lost due to congestion per unit time between the two previous losses due to link error.
- n is the number of packets transmitted in the current window.

We studied the performance of IFTPV in 6 different scenarios. In a network with:

1. Low BER
2. Low BER and congestion due to competing TCP, UDP and IFTP sources
3. Low BER and UDP based congestion. This scenario tests IFTPV's friendliness problem.
4. Low BER with TCP based congestion. This scenario tests fairness
5. Various BER and a competing IFTP-W
6. Low BER and competing IFTP-W with various lengths of error sensitive part.

The network topology used in this section is similar to the dumbbell topology of Fig. 1, but has two bottleneck routers connected with a wired network of 50 Mbps and 50 ms delay. The delay of the wired links is 10 ms and that of the wireless links is 100 ms. Below are the results of our simulations. The results are average of 60 runs of 10 s for each value of BER.

10.3. Scenario 1

In this scenario we study the performance of each protocol flow, UDP, TCP, IFTP, and IFTPV, as it exists alone in the network. We focus on the wireless problems and we study the effect of the BER on the performance of IFTPV and compare it to the other protocols. The network topology in this case has a single source and its corresponding client sharing the bottleneck routers. In this scenario, the routers have an infinite buffer to ensure that all packet drops are caused by the wireless link and no packet is lost due to the network congestion.

We varied the BER values from 10^{-3} to 10^{-7} . The throughput of UDP was much higher than the other three protocols. They were not plotted on the graph in order to zoom in and distinguish between the other three protocols. The results of Figs. 21–23 can be compared to those of Table 1.

It can be seen in Fig. 21 that IFTPV manages to maintain a better throughput than IFTP. It can also be seen that

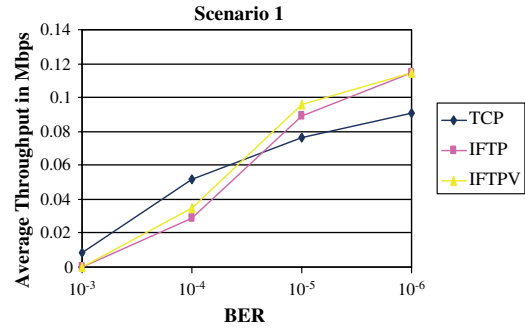


Fig. 21. Average throughput of protocols as they stand alone in a wireless network with various BER.

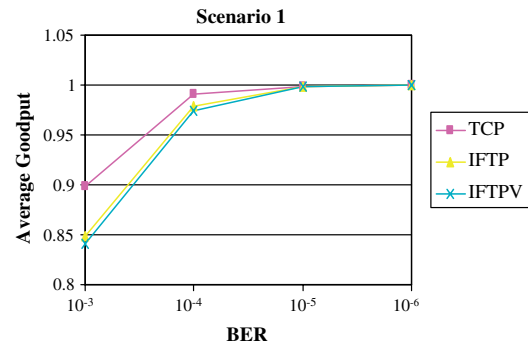


Fig. 22. Average goodput of protocols vs. as they stand alone in a wireless network with various BER.

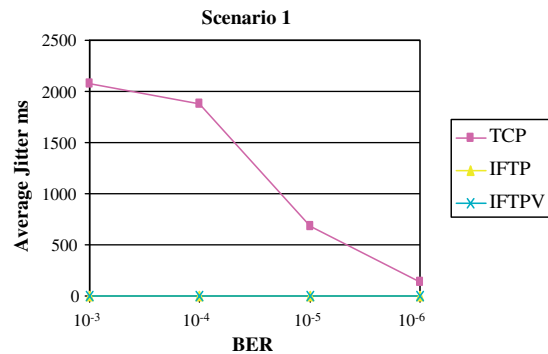


Fig. 23. Average jitter of protocols as they stand alone in a wireless network with various BER.

BER	Throughput (Mbps)	Goodput	Jitter (ms)
10^{-3}	0.929384	0.91563774	35.1558921
10^{-4}	0.891845547	0.87655887	34.3698445
10^{-5}	0.899109867	0.88616871	37.0670941
10^{-6}	1.014740373	0.99990701	32.2957187

IFTP and IFTPV can outperform TCP, in networks with BER lower than 10^{-5} .

From Fig. 22 one can notice that IFTPV's goodput is within 0.1% of that of TCP. Since the focus here is on wireless losses and there are no congestion losses, IFTPV's

goodput is very similar to that of IFTP. Comparing IFTPV's goodput in Fig. 22 with UDP's goodput in Table 1, one can notice that on average IFTPV's goodput is better than UDP's. Since IFTPV, as UDP, does not retransmit lost packets, minimizing the number of lost packets is a great advantage as it increases the QoS. From Fig. 23 it is clear that both IFTP and IFTPV maintain a low jitter which is a great feature of any connectionless transport protocol used for the transfer of multimedia applications.

10.4. Scenario 2

Scenario 1 gives us insight on the performance of IFTPV when run alone in comparison to UDP, TCP, and IFTP in reaction to various BER. However, the main objective of IFTPV is to improve the performance of IFTP as it distinguishes between losses due to congestion and those due to the wireless link errors. In this scenario we ran experiments to study the effect of both congestion and wireless problems. In this scenario IFTPV had to fight for bandwidth with competing UDP, TCP and IFTP flows. To illustrate the effect of congestion the buffer size was limited to 50K. The average throughput of UDP was not plotted in the graph for the same reason mentioned earlier. The performance metrics of UDP for the corresponding BER are illustrated in Table 2.

It is clear from Fig. 24 IFTPV manages to outperform IFTP. It has a better throughput and this is the result that it distinguishes between wireless and congestion losses and hence does not throttle its transmission rate in case of wireless losses, resulting in higher throughputs. From Fig. 25, we can see that IFTPV still maintains a goodput within 0.1% of TCP. Fig. 26 shows that TCP has very high delay jitter and hence is not suitable for the transfer of multimedia applications, while IFTPV's jitter is still extremely low even if it is competing with multiple other flows.

10.5. Scenario 3

In this scenario IFTPV suffers from both congestion and wireless problems as well. The buffer size was limited to 50K as in scenario 2. However, this scenario represents UDP-based congestion. The BER was set to 10^{-5} . We studied the performance of IFTPV as the sending rate of UDP is varied. The UDP sending rate was varied from 10 to 1000 kbps. We compared IFTPV's performance to that of TCP's. Fig. 27 shows the effect it had on the throughput of TCP and IFTPV. The figure demonstrates

Table 2
Average values for UDP under scenario 2 for various BER

BER	Throughput (Mbps)	Goodput	Jitter (ms)
10^{-3}	0.9286112	0.915126467	34.4345609
10^{-4}	1.005395627	0.991002197	32.53960098
10^{-5}	1.015148853	0.999127881	36.63492211
10^{-6}	1.01577936	0.999904609	36.25059913

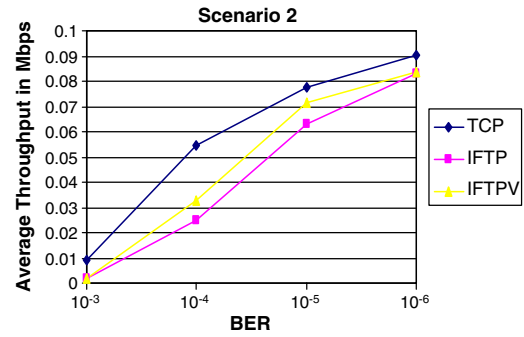


Fig. 24. Average throughput of protocols in a congested wireless network with various BER.

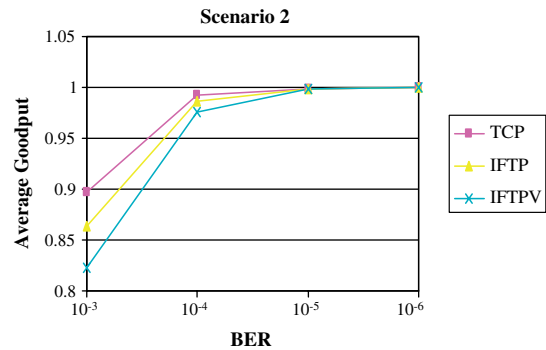


Fig. 25. Average goodput of protocols in a congested wireless network with various BER.

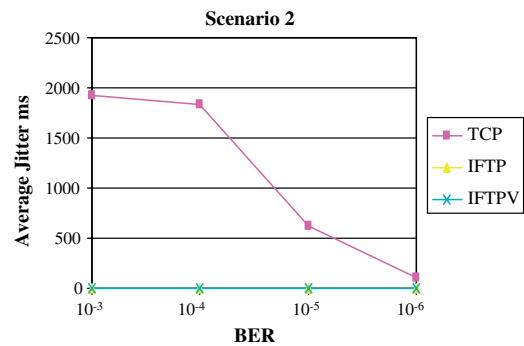


Fig. 26. Average jitter of protocols in a congested wireless network with various BER.

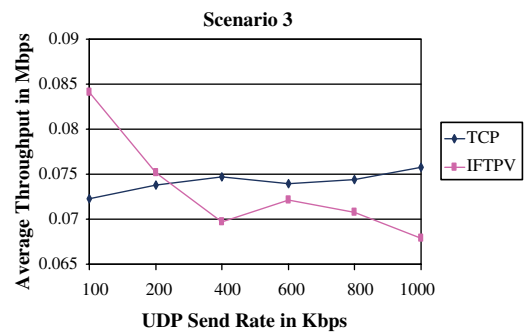


Fig. 27. Average throughput in a wireless network with BER 10^{-5} and UDP-based congestion.

the TCP-friendliness property of IFTPV. From Fig. 27 it can be seen that IFTPV manages to have a transmission rate similar to TCP in highly congested networks. TCP and IFTPV have a similar share of the bandwidth and hence IFTPV is considered TCP-Friendly. However, IFTPV's throughput is mostly less than TCP, but it is possible for IFTPV to achieve greater rates than TCP. As mentioned in [2], IFTP, and hence IFTPV, can operate in an extended mode which gives selected flows a QoS-differentiated service. In Fig. 28, we notice some oscillations. This is due to the oscillatory behavior inherent in TCP [25]. TCP and consequently IFTPV, as it is an emulation of TCP, do not converge to an equilibrium state but oscillates around the optimal state. The variance in the achieved utilization results from the oscillatory behavior of TCP and IFTPV. This is also similar to the results obtained in [2] and [26] that shows that the link utilization of IFTP and DAA (another TCP-friendly protocol), respectively, can range from 60% to 100%. Fig. 29 shows consistent results for delay jitter as in previous scenarios.

10.6. Scenario 4

This scenario is similar to scenario 3, but represents TCP based congestion. In this case IFTP connections competed with no UDP connections. This scenario demonstrates the

fairness property of IFTPV. This is an important feature of IFTPV, as we demonstrate its compatibility with TCP, the dominating transport protocol on the Internet today. In this scenario the network also suffers from congestion and BER.

As can be seen in Fig. 30 IFTPV performs better than IFTP in a TCP-based congested wireless network. IFTPV and TCP have a fair share of the bandwidth. IFTPV has on average a throughput lower than TCP in BER 10^{-3} – 10^{-5} , however it does eventually surpass TCP's throughput as BER lowers. Fig. 31 shows similar results of goodput as in scenario 2. Fig. 32 shows again consistent results for delay jitter. This means that IFTPV suffers almost no jitter in various network scenarios.

10.7. Scenario 5

Scenarios 5 and 6 compare the performance of IFTP-W and IFTPV. In Scenario 5 we study their performance under various BER. Both protocols had a packet size of 1464 which is the default packet size for packets containing streaming video data over EDCF on 802.11e networks. The size of IFTP-W error-sensitive part is fixed to 512.

In Fig. 33, it can be seen that IFTPV outperforms IFTP-W in terms of throughput. Fig. 34 shows that IFTPW has a slightly greater goodput. Our experiments also show in

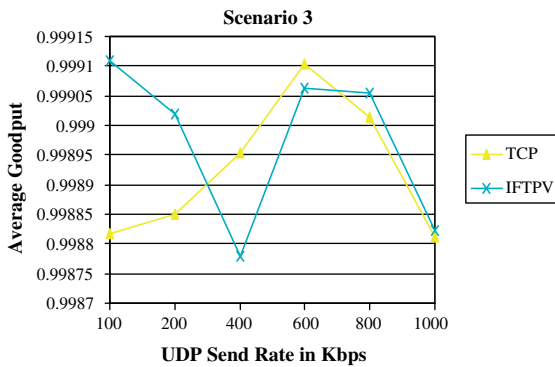


Fig. 28. Average goodput in a wireless network with BER 10^{-5} and UDP based congestion.

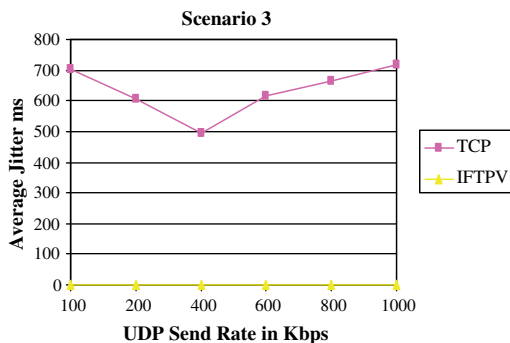


Fig. 29. Average jitter in a wireless network with BER 10^{-5} and UDP-based congestion.

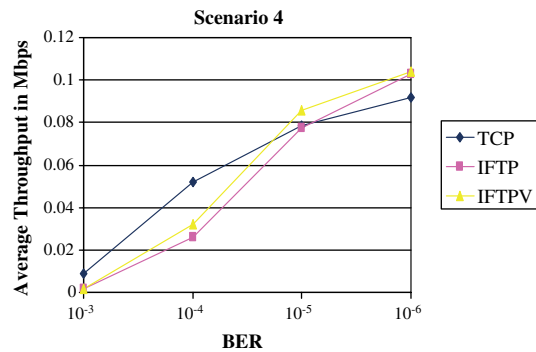


Fig. 30. Average throughput of protocols in wireless TCP-based congested network with various BER.

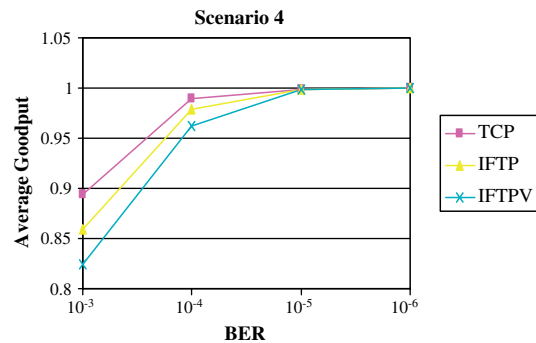


Fig. 31. Average goodput of protocols in wireless TCP-based congested network with various BER.

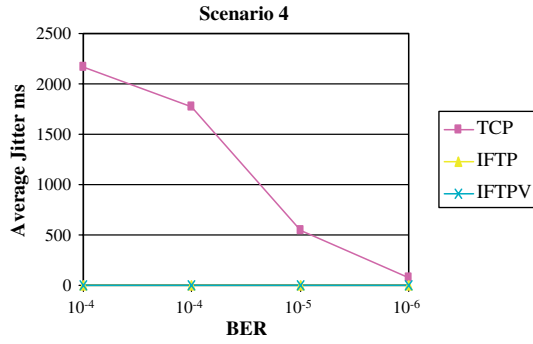


Fig. 32. Average jitter of protocols in wireless TCP-based congested network with various BER.

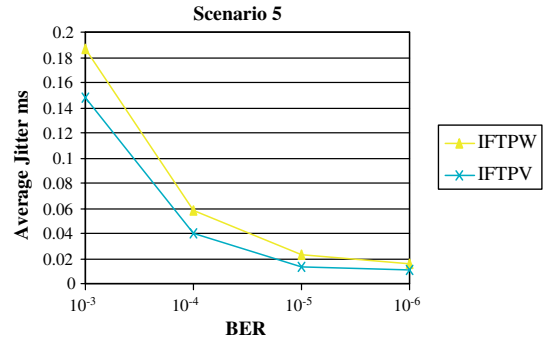


Fig. 35. Comparison of average delay jitter of IFTP-W and IFTPV under various BER.

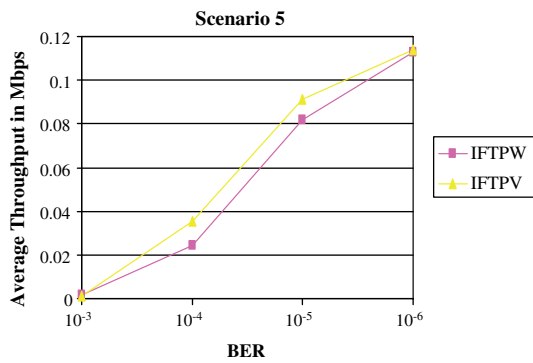


Fig. 33. Comparison of average throughput of IFTP-W and IFTPV under various BER.

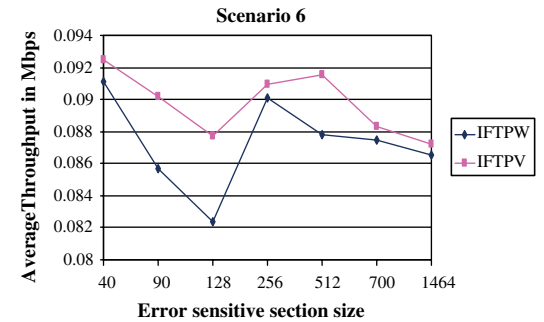


Fig. 36. Comparison of throughput of IFTPV and IFTP-W with different lengths of error-sensitive section.

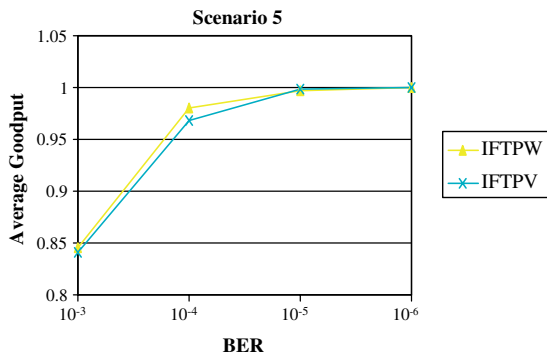


Fig. 34. Comparison of average goodput of IFTP-W and IFTPV under various BER.

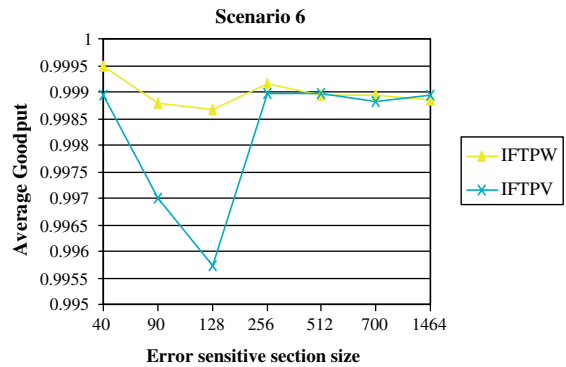


Fig. 37. Comparison of goodput of IFTPV and IFTP-W with different lengths of error-sensitive section.

Fig. 35 that IFTPV maintains a lower delay jitter and so would be the better choice for multimedia applications.

10.8. Scenario 6

In this scenario the BER is fixed to 10^{-4} . Similar packet sizes as scenario 5 were used. We vary the length of the error-sensitive part of IFTP-W from 40 which is the merely the header to 1464 which is the whole packet.

In Fig. 36 it can be seen that IFTPV maintains an average throughput greater than IFTP-W no matter what IFTP-W's sensitive section length is. However, Fig. 37

shows that IFTPW maintained a greater goodput, while IFTPV did maintain a lower jitter, as Fig. 38 shows.

11. Conclusions

In this paper we proposed two new end-to-end protocols for the transmission of multimedia applications on heterogeneous wired-wireless networks. The first is IFTP-W which partitions packets into error-sensitive and error-insensitive sections. By only discarding packets with errors in the error-sensitive section, both goodput and throughput are increased. The length of the error-sensitive section is an

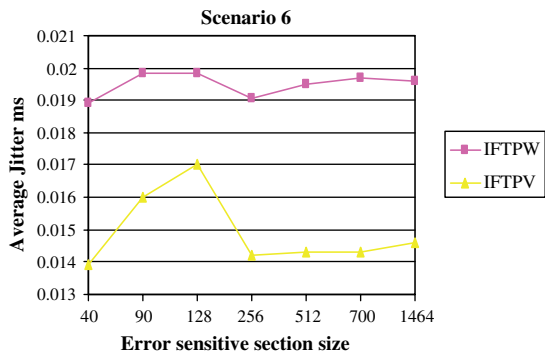


Fig. 38. Comparison of average jitter of IFTPV and IFTP-W with different lengths of error-sensitive section.

important parameter in IFTP-W. We study the effect of its variance for video and audio packets with wireless links of various bit error rates. The second proposal is IFTPV, which improves the performance of IFTP by distinguishing between packet loss due to congestion and wireless losses. We analyzed the performance of IFTPV in extensive scenarios with various network conditions. The results of our simulations indicate that these modifications do provide the expected improvements. We demonstrated that both IFTP-W and IFTPV maintains the fairness and TCP-friendliness properties. They also maintained a low packet delay jitter on all networks and so make for good multimedia transmission protocols. The comparison of IFTP-W and IFTPV shows that IFTPV has higher throughput and lower jitter, while IFTP-W has higher goodput. To combine the benefit of both proposals, IFTPV could take advantage of IFTP-W's ability to specify which section is error-insensitive. This paper also describes the object oriented discrete event networking simulator we wrote in C# to evaluate the performance of our designs.

References

- [1] S. Floyd, K. Fall, Promoting the use of end-to-end congestion control in the Internet, *IEEE/ACM Transactions on Networking* 7 (4) (1999) 458–472.
- [2] H. ElAarag, M. Bassiouni, An internet friendly transport protocol for continuous media over best effort networks, *International Journal of Communication Systems* 15 (2002) 881–898.
- [3] IETF, User Datagram Protocol, RFC 768, August 1980, <<http://www.ietf.org/rfc/rfc0768>>.
- [4] IETF, Transmission Control Protocol, RFC 793, September 1981. <<http://www.ietf.org/rfc/rfc0793>>.
- [5] IETF, TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, RFC 2001, January 1997, <<http://www.ietf.org/rfc/rfc2001>>.
- [6] J. Widmer, R. Denda, M. Mauve, A survey on TCP-friendly congestion control, *IEEE Network* 15 (2001) 28–37.
- [7] M. Handley, S. Floyd, J. Padhye, J. Widmer, TCP Friendly Rate Control (TFRC): Protocol Specification, <<http://www.icir.org/tfrc/rfc3448.txt>>.
- [8] H. ElAarag, Improving TCP performance over mobile networks, *Journal of ACM Computing Surveys* 34 (3) (2002) 357–374.
- [9] C.P. Fu, Soung C. Liew, TCP Veno: TCP enhancements for transmission over wireless access networks, *IEEE Journal on Selected Areas in Communication* 21 (2) (2003) 216–227.
- [10] S. Zabir et al., Ensuring fairness among ECN and non-ECN TCP over the Internet, *International Journal of Network Management* 13 (2003) 337–348.
- [11] L. Larzon, M. Degermark, S. Pink, Efficient Use of Wireless Bandwidth for Multimedia Applications, in: *IEEE MoMUC*, November 1999.
- [12] K. Chandrayana, S. Kalyanaraman, Uncooperative Congestion Control, in: *SIGMETRICS/Performance '04*, vol. 32, No. 1, June 2004, pp. 258–269.
- [13] H. ElAarag, M. Bassiouni, Performance evaluation of TCP connections in ideal and non-ideal network environments, *Computer Communications Journal* 24 (18) (2001) 1769–1779.
- [14] H. ElAarag, A. Moedinger, IFTP-W: a TCP-friendly protocol for multimedia applications over wireless networks, in: *ACM Southeast Conference*, March 2005.
- [15] H. ElAarag, A. Moedinger, Performance evaluation of an internet friendly transport protocol over networks with lossy links, in: *Applied Telecommunication Symposium, Spring Simulation Multiconference*, San Diego, California, April 2005, pp. 21–25.
- [16] S. Choi et al., IEEE 802.11e contention-based channel access (EDCF) performance evaluation, in: *Proceedings of IEEE ICC '03*, Anchorage, AK, May 2003, vol. 2, pp. 1151–1156.
- [17] S.A. Khayam, S. Karande, M. Krappel, H. Radha, Cross-layer protocol design for real-time multimedia applications over 802.11 b networks, in: *Proceedings of International Conference on Multimedia and Expo, 2003 ICME '03*(July 2003), vol. 2(6–9), pp. 425–428.
- [18] Larzon et al., Efficient transport of voice over IP over cellular links, in: *IEEE Global Telecommunications Conference, GLOBECOM 2000*, vol. 3, 2000, pp. 1669–1676.
- [19] E. Wu, M.Z. Chen, JTCP: jitter based TCP for heterogeneous wireless networks, *IEEE Journal on Selected Areas in Communications* 22 (4) (2004) 757–766.
- [20] K. Xu et al., TCP-Jersey for wireless IP communications, *IEEE Journal on Selected Areas in Communications* 22 (4) (2004) 747–756.
- [21] C. Casetti et al., TCP Westwood: end-to-end bandwidth estimation for enhanced transport over wireless links, *Wireless Networks* 8 (2002) 467–479.
- [22] Milan Todorovic, Noé López-Benítez, Efficiency study of TCP protocols in infrastructured wireless networks, *IEEE* 2006.
- [23] K. Kaneko, J. Katto, Reno friendly TCP Westwood based on router buffer estimation, in: *Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS/ICNS)*, 2005.
- [24] Bin Zhou, Cheng Peng Fu, An enhancement of TCP Veno over light-load wireless networks, *IEEE Communications Letters* 10 (6) (2006).
- [25] D. Chiu, R. Jain, Analysis of the increase/decrease algorithms for congestion avoidance in computer networks, *Journal of Computer Networks and ISDN* 17 (1) (1989) 1–14.
- [26] D. Sisalem et al., The Direct Adjustment Algorithm: a TCP-friendly Adaptation Scheme, *Lecture Notes in Computer Science*, 1922, Springer, Berlin, 2000.