

IFTP-W: A TCP-Friendly Protocol for Multimedia Applications over Wireless Networks

Hala ElAarag Andrew Moeding
Department of Mathematics and Computer Science
Stetson University
DeLand, Florida, U.S.A.
Email: {helaarag, amoeding}@stetson.edu

ABSTRACT

Recent growth in the use of multimedia applications on wireless networks is calling for the development of a protocol that is both TCP-friendly and capable of conforming to the constraints of wireless networks. This paper presents IFTP-W, an end-to-end congestion control protocol we designed to meet both requirements. 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.

Categories and Subject Descriptors

C.2.2 [Computer Systems Organization]: Computer Communication Networks – *Network Protocols*.

General Terms

Algorithms, Design, Performance

Keywords

TCP-Friendly, congestion control, media streaming over wireless, network simulation, wireless congestion control, IFTP, IFTP-W

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), developed by ElAarag and Bassiouni [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 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 a transport protocol 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 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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

43rd ACM Southeast Conference, March 18-20, 2005, Kennesaw, GA, USA. Copyright 2005 ACM 1-59593-059-0/05/0003...\$5.00.

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.

The rest of the paper is organized as follows. Section 2 presents previous research. In section 3, we briefly explain the IFTP protocol and discuss its performance on wireless networks. Section 4 discusses our proposal for improving the performance of IFTP in networks with wireless links. In this section, we present the IFTP-W protocol. Performance evaluation of the IFTP-W protocol using simulation tests is presented in section 5. Finally, section 6 gives conclusions of the paper.

2. 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 [3]. One example protocol, TCP Friendly Rate Control (TFRC) [4] 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 [5]. 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 [6].

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 [7]. 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. [8] 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 allocation 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 have proposed modifying only the routers at the edge of a network [9]. Router-based flow management can protect congestion conscious flows from being deprived of equal network resources by selfish streams.

3. IFTP OVER WIRELESS NETWORKS

In this section, we first briefly explain the IFTP protocol. More details can be found in [2]. IFTP regulates packet flow by manipulation of the inter-packet gap in response to network congestion. The appropriate inter-packet 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.

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. Both IFTP and 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.

The use of selective ACKs (SACK TCP) is one way to improve TCP's throughput when multiple packets are dropped in one window [10]. While IFTP does not retransmit lost packets, it could nonetheless emulate SACK TCP as its use becomes more widespread.

We studied the performance of IFTP on wireless networks through simulation. There are UDP, TCP and IFTP sources sharing one bottleneck router and transferring 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 milliseconds and each wireless link has a delay of 10 milliseconds. For IFTP and TCP, 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. UDP packets are also 1024 bytes in length and are sent at an average rate of 1 megabit per second with an exponentially distributed inter-packet gap. The links between the sources and the router are wired and have a bandwidth of ten megabits per second, while the links between the router and the corresponding receivers are wireless with a bandwidth of one megabit per second. All presented results are averaged from ten runs of simulated connections running for approximately two minutes for each value of BER.

We focused on two important performance measures: throughput and goodput. We define the throughput as number of bits per second received at each receiver. The goodput is an indication of the utilization of the network. We define it as the ratio between the number of packets received at the receiver to the number of packets sent. Figures 1-2 show our simulation results for the two mentioned performance measures.

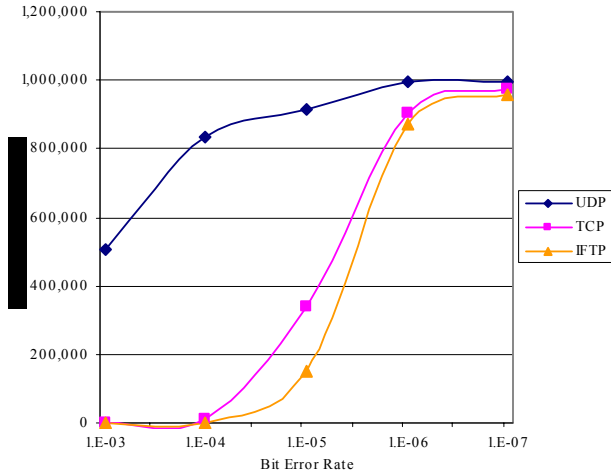


Figure 1. Throughput as a function of BER.

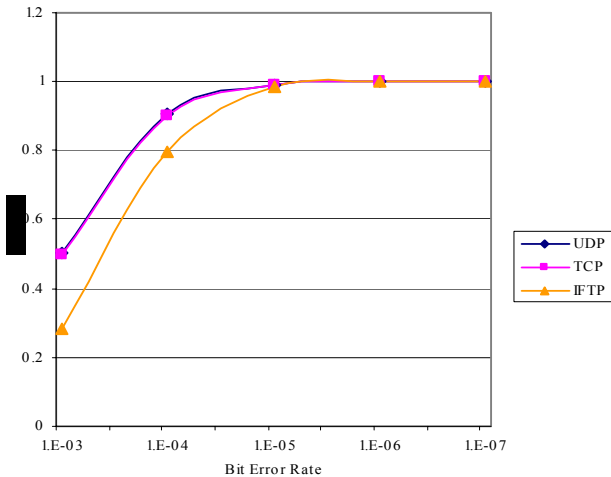


Figure 2. Goodput as a function of BER.

As expected, UDP reacts less severely to the change of BER than either IFTP or TCP as shown in Figure 1. 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 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 imitation 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 Figure 2. As goodput is a percentage of packets received, it maintains approximately the same relation to BER for all protocols.

4. IFTP-W

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 [8]. 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.

Figure 3 shows the IFTP packet header format. The packet header format for an IFTP-W packet, shown in Figure 4, is 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. 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 inter-packet gap has elapsed if $cwnd + una < next$, where *cwnd* is the size of the congestion window, *una* is the sequence number of

Source Port	Destination Port
Packet Num	Checksum
Timestamp	Length
Payload	

Figure 3. IFTP packet format.

Source Port	Destination Port
Packet Num	Checksum
Timestamp	Coverage
Payload	

Figure 4. IFTP-W packet format.

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 Figure 5 demonstrate that the TCP-friendliness is preserved. As the sending rate of UDP is increased, both IFTP-W 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.

5. 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 3, 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.

5.1 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 two minutes, 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

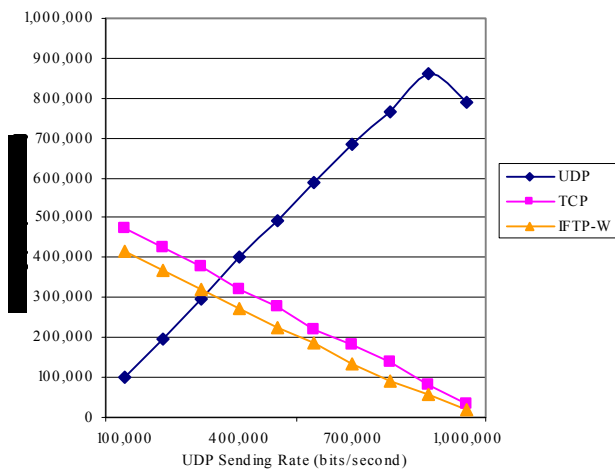


Figure 5. IFTP-W's TCP-friendliness.

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 Figure 6. 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 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.

5.2 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 Figure 7. 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.

6. CONCLUSIONS

We have introduced a 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-

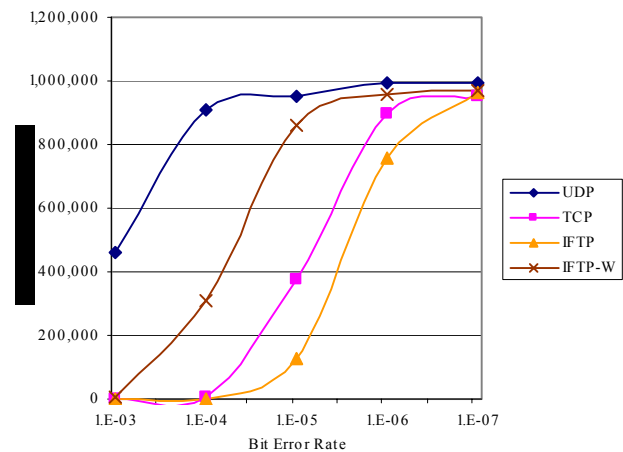


Figure 6. Throughput as a function of BER.

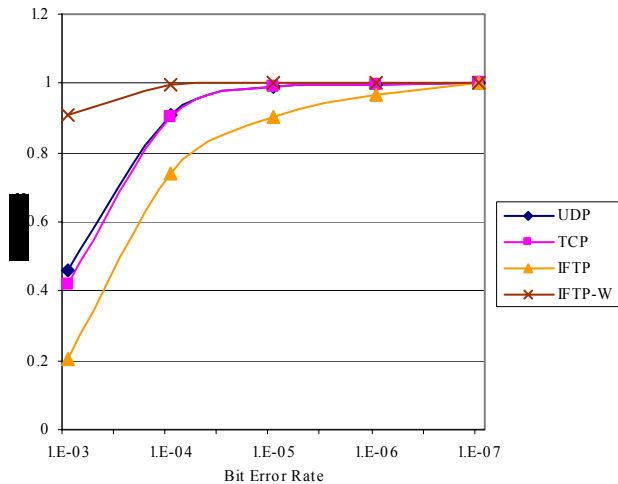


Figure 7. Goodput as a function of BER.

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. The effect of varying the error-sensitive section length should be carefully studied in order to provide guidance in the selection of error-sensitive section lengths.

The inclusion of ECN into IFTP is another modification with great potential for improving performance, especially over wireless networks. If all nodes are ECN-enabled, the use of packet drops as a signal of congestion can be replaced with the explicit notification of network congestion and the erroneous responses to perceived congestion can be avoided.

Simulations are a crucial step in protocol verification and testing, but they only provide an initial perspective on the actual performance of the protocol. A physical implementation of IFTP-W will provide further confirmation of its ability to counteract the negative effects of wireless networks. Real-world environments are less predictable than simulated environments and contain many factors that elude accurate simulation. Thus, testing a full implementation will enable refinement of the protocol as well as a more accurate understanding of its effects.

7. ACKNOWLEDGEMENTS

The authors are grateful to the anonymous reviewers for their comments and suggestions that helped improve the quality of this paper.

This work has been supported by Stetson University's SURE and Faculty Professional Development grants. The views and conclusions herein are those of the authors and do not represent the official policies of Stetson University.

8. REFERENCES

- [1] 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, pp. 458-472, August 1999.
- [2] H. ElAarag and M. Bassiouni, "An Internet Friendly Transport Protocol for continuous media over best effort networks," *Int. J. of Communication Systems*, John Wiley and Sons, 2002, 15:881-898.
- [3] J. Widmer, R. Denda, and M. Mauve., "A survey on TCP-Friendly congestion control," *IEEE Network*, vol. 15, pp. 28-37, May 2001.
- [4] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," <http://www.icir.org/tfrc/rfc3448.txt>.
- [5] H. ElAarag, "Improving TCP performance over mobile networks," *Journal of ACM Computing Surveys*, vol. 34, no. 3, pp. 357-374, September 2002.
- [6] C. Fu, S. Liew, "TCP VenO: TCP Enhancement for Transmission Over Wireless Access Networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 2, pp. 216-228, February 2003.
- [7] S. Zahir, et al. "Ensuring fairness among ECN and non-ECN TCP over the Internet," *Int. J. of Network Mgmt*, John Wiley and Sons, 2003, 13:337-348.
- [8] L. Larzon, M. Degermark, and S. Pink, "Efficient use of wireless bandwidth for multimedia applications," *IEEE MoMUC*, November 1999.
- [9] K. Chandrayana and S. Kalyanaraman, "Uncooperative Congestion Control," *SIGMETRICS/Performance '04*, vol. 32, no. 1, pp. 258-269, June 2004.
- [10] K. Fall and S. Floyd, "Simulation-based comparison of Tahoe, Reno, and SACK TCP," *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 3, pp. 5-21, July 1996.