# Improving Quality of Service in Wireless Multimedia Communication with Smooth TCP

Michael Bauer
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada, N6A5B7
Email: bauer@csd.uwo.ca

Md. Ashfaqul Islam
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada, N6A5B7
Email: mislam59@uwo.ca

***Abstract*—Wireless multimedia communications are becoming mainstays of many applications and these applications are likely to continue to grow and become more demanding. TCP, though not designed for this kind of communication, is still commonly used. Smooth TCP (STCP) has been introduced in previous research as a variant of TCP with properties that can enable it to work better in environments in which TCP was not originally designed, such as wireless multimedia communications. While STCP has been compared to TCP is some situations, it has not been compared to TCP in wireless multimedia environments. In this paper, we briefly describe STCP and report on initial experiments that compare TCP and STCP through simulation. The the results suggest that STCP can provide better support than TCP for wireless multimedia communications.**

***Keywords-wireless communication, multimedia, TCP, performance.***

## I. Introduction

The growth over the past decade in the computational power and wireless capabilities of consumer handheld devices has ushered in an era of new and exciting mobile applications and services. Many of these applications and services have created a demand for multi-modal, dynamic data. This data is delivered using TCP or UDP over the Internet, neither of which was designed with multi-modal, time sensitive data in mind. In particular, TCP was not designed to operate with time-sensitive data nor in wireless environments, though it remains a core protocol in use for wireless data traffic. A substantial challenge for TCP is when multi-modal traffic must be delivered to an application over a wireless connection.

Unlike general data communication, multimedia communication and time sensitive streaming applications rely on timely data delivery and are somewhat less concerned about the guaranteed end-to-end data delivery. In this regard, TCP is not suitable as because it chooses reliability over timeliness. A number of researchers have looked at ways to enhance TCP's timeliness properties. For example, some researchers [1] have proposed new protocols which ensures a minimum rate for multimedia traffic; this is, however, not feasible for wireless environments where bandwidth availability is a great concern.

In this paper, we examine the performance of Smooth TCP (STCP). STCP, introduced and studied in previous research [2] [3] [4]; is a variation of TCP which behaves much like TCP in general data transfer. STCP differs, however, in that it is first of all parameterized and with a suitable set of parameters can be set to have properties more suitable for communications in environments for which TCP was not originally designed. The performance of STCP has been compared to TCP in several scenarios, but there has been no comparison to TCP in wireless, multimedia communication. In this paper, we describe a parameterized version of STCP for wireless multimedia environments and compare it to TCP.

The paper is organized as follows. The following section reviews some previous work on addressing modifications to TCP to handle multi-modal data and to operate in wireless environments. In Section III, we briefly introduce STCP, describe its properties and introduce a parameterized version for wireless multimedia environments. Section IV outlines the simulation approach and tools used. Section V presents experiments and results. Finally, we draw conclusions on the potential use of STCP and outline some future work.

## II. Related Work

Researchers exploring variations in TCP have typically looked to address either the use of TCP in wireless environments or its use with multi-modal data, but not both. We briefly look at some of the previous approaches in addressing these challenges.

Some of the challenges with TCP in wireless multimedia communication environments have been identified by previous research [5]. The wireless medium is very susceptible to path loss or link failure, is based on shared bandwidth, requires hand-off of mobile devices between access points, etc. TCP was developed focusing on wired networks and was not designed to support Quality of Service in an unreliable wireless medium. Previous research has identified that unmodified standard TCP performs poorly in a wireless environment, as TCP can not distinguish packet losses caused by network congestion from those attributed to transmission errors. TCP misinterprets this loss as congestion and invokes congestion control. This leads to unnecessary retransmissions and loss of throughput [6]. The congestion control mechanism of TCP reacts adversely to packet losses due to temporarily broken routes in wireless networks [7], [8].

TCP has been designed to ensure that data arrives at its destination regardless of timing dependencies and this can

 ISBN: 978-1-61208-186-1 

be challenging for TCP for the transmission of time-sensitive data [9]. To address the transfer of multimedia, a number of alternative strategies have proposed changes to TCP. SCTP (Stream Control Transmission Protocol) is a relatively new transport layer protocol which aims to transport telecommunication signaling messages over an IP based network [10]. SCTP can be said to have a blend of features of TCP and UDP, for example, it inherits TCP's congestion control scheme and connection oriented communication [11]. SCTP also offers two other distinct features: multi-homing and multi-streaming [2] [10] [11]. A simulation study of the performance [12] showed TCP outperformed SCTP in some cases because of extra overhead present in SCTP.

Other approaches have looked to provide protocols at the application layer in order to compensate for TCP's deficiencies as well as to provide other time-based services. The Real-time Transport Protocol (RTP) is used extensively in communication and entertainment systems that involve streaming media such as audio and/or video tele-conference, Internet telephony, Internet TV etc. [13]. It provides identification and sequential orderings of data bits. It can also monitor the delivery of multimedia content. RTP does not address resource reservation and does not guarantee quality-of-service for real-time services. RTP provides data delivery monitoring services and stream control with the help of another supporting protocol - the Real-time Transport Control Protocol (RTCP) [13]. At the level of functionality, RTP and RTCP use two consecutive ports and carry data and control information side by side. This facilitates the option of using pause and play in audio /video streams.

RTSP (Real TIme Streaming Protocol) [14] functions similarly to HTTP but differs from it in that it requires a permanent connection. It uses a message identifier to monitor each data connection. The protocol is used to establish and control media sessions between end points. RTSP works with sessions rather than connections with the server. There is no notion of an RTSP connection; instead, a server maintains a session labeled by an identifier. While using RTSP as an application level protocol, the client can open and close several connections to the server and can use RTSP requests. RTSP also uses the underlying (transport) layer.

Alternatives to TCP have focused on the application layer or involve substantive changes to TCP. Moreover, most of the research efforts have focused on how to deal with multimedia applications over TCP and do address the challenges that TCP has in wireless environments. In practice, both need to be addressed.

## III. SMOOTHTCP (STCP)

SmoothTCP (STCP) [2], [3], [4] is a very recent addition to the collection of TCP variations. Vieira [3] introduced STCP and compared it to standard TCP through various experiments. The results demonstrated that STCP can outperform TCP in some circumstances and is as good otherwise. One of the main advantages os STCP is that it works very naturally in a "normal" TCP environment.

STCP mainly differs from standard TCP in the way it works with the congestion control mechanism. STCP introduces the notion of *smoothness* within congestion control [4]. In order to create a smooth congestion control mechanism, additional parameters are used in conjunction with the basic parameters of TCP in order to smooth out the way congestion is managed.

Four algorithms [15] define TCP's congestion control scheme: 1) the *slow start* phase ensures a moderate increase in the size of sending window (named **cwnd**); 2) the *congestion avoidance* phase starts when **cwnd** exceeds the threshold and continues unless a congestion is detected; 3) the *fast retransmit* phase; and the *fast recovery* **cwnd** and the threshold are both halved. STCP differs from TCP in its approach to congestion control in that it introduces the notion of *smoothness* within congestion control. Though it uses the same slow start algorithm, it manages the size of **cwnd** differently. The change in the window size is done more *smoothly* and results in fewer frequent radical changes in the value of **cwnd**. It introduces smoothness to the congestion control of TCP. There are three key properties required for *smoothness*.

*1. Smooth Curve Property*: TCP's congestion avoidance scheme uses the Additive Increase Multiplicative Decrease (AIMD) algorithm [15]. The AIMD algorithm is generalized by the family of Binomial Congestion Control (BCC) Algorithms [16]. As per the definition of BCC, the functions belonging to this family introduce discontinuities in the graph of the congestion window which can affect the performance of TCP in certain circumstances. To achieve smoothness, the graph needs to be smoothed [3].

*2. Vertical Smoothness Property*: TCP uses retransmission timeouts (RTOs) and duplicate acknowledgements (dupACKs) to detect packet drops or congestion. However, there might be other reasons for a temporary timeout, e.g., a packet may have traveled on a redundant path or there might be a sudden increase in link latency in wireless network. Both could cause a timeout which could make TCP believe that there is congestion. These events do not always mean congestion or packet drop, yet TCP might take action to control the congestion, which would be unnecessary. These types of spurious events are called *Vertical Bursts* [3] and cause spikes in the average time for acknowledgements. To avoid TCP reacting too quickly, the occurrence of the "vertical bursts" needs to be smoothed.

*3. Proactive Control Property*: We know that TCP detects a congestion or a packet drop, but it takes almost 1 RTT plus 3 dupACK arrival times to inform the sender that a congestion has occurred. This is the time that TCP waits before taking any action against congestion. This relatively long time is known as a *temporal gap* and within this gap a sender may send other packets. The proactive control property of STCP reduces the temporal gap [3]. Three different methods were considered for reducing the temporal gap using different metrics: Variation in Round Trip Time (RTTVar), Number of Retransmission Time Outs (RTOs), Number of Retransmitted Packets. Our work here focuses on retransmission time outs.

 ISBN: 978-1-61208-186-1 

### A. The STCP Function

SmoothTCP is defined as a subset of functions of the Smooth Congestion Control Algorithms [3]. The general form of the STCP function is defined as follows:

$$D = \frac{\delta w}{\delta p_1} dp_1 + ... + \frac{\delta w}{\delta p_i} dp_i + ... + \frac{\delta w}{\delta p_n} dp_n \quad (1)$$

Here, $p_1$, $p_2 ... p_n$ are the set of *state variables* which describe the network's status. For example, the variation in Round Trip Time (RTTVar), the number of Retransmission Time Outs (RTO), or the number of retransmitted packets are some of the variables that could be used in equation 1. $D$ is the change in the congestion window **cwnd**. A positive value indicates an increment of $D$ to the current **cwnd**, while a negative value indicates a decrease. The basic idea is that the change in the congestion window, namely $D$, is determined by changes in the state variables, hence the partial derivatives. In turn, the change in each parameter is modeled using a sigmoid function $f(u)$ to accommodate bursts, spikes and to adjust the window size to adapt the change; the proposed function is: $f(u) = A + C * tanh(B(u + M))$ [3]. Here, $u$ is one of the state variables and $A$, $B$, $C$ and $M$ are coefficients providing weighting of each such state variable. Thus, each one of the partial derivatives are of the form:

$$f(u) = \frac{\delta w}{\delta p_i} = A_{p_i} + C_{p_i} tanh(B_{p_i}(p_i + M_{p_i})) \quad (2)$$

$A$ is the lower asymptote and it determines the smallest value of $f(u)$. $B$ controls the growth rate of $f(u)$, $C$ is the width of the range set of $f(u)$ which determines the maximum amount of variation in the window size, and $M$ can be used for controlling the position of the curve on the $u$-axis.

The coefficients $A$, $B$, $C$ and $M$ can be determined using environmental conditions and could be set as constants, set by an application (e.g. by a multimedia server prior to sending content) or even set dynamically based on operating conditions. For this study, we have chosen to use three *state variables*: Round Trip Time (*rtt*), the number of Retransmission Time-Outs (*rto*) and the number of retransmitted packets (*resend_pack* or *rsnd*) as the variables $(p_i)$ of equation 1. These variables were initially selected since they are readily available within most implementations of TCP; a brief explanation of these variables and the coefficient values selected follows.

*rtt*: *rtt* refers to the time difference between the time a packet is sent from the sender and the time sender receives the associated acknowledgement of that packet. TCP keeps track of round trip time and uses it to detect any possible congestion. In a steady network, the round trip time remains steady in terms of variation. The larger the difference between a measured round trip time and the average as measured by TCP, the higher the possibilities of congestion in the network. Round trip time estimation was proposed by Van Jacobson in [17] and is used in our simulation. The form of the equation 2 for *rtt* is defined as:

$$\frac{\delta w}{\delta rtt} = A_{rtt} + C_{rtt} tanh(B_{rtt}(rtt + M_{rtt})) \quad (3)$$

$$\frac{\delta w}{\delta rtt} = 1452 \times tanh(-(rtt - 1)) \quad (4)$$

Thus, the values for these coefficient are:

$A_{rtt} = 0$
$C_{rtt} = maximumsegmentsize(mss) = 1452$
$B_{rtt} = -1$
$M_{rtt} = -1$

$A_{rtt}$ is set to 0 to get a smallest value from overall calculation. $C_{rtt}$, being the width of the range set of the function, is set to the Maximum Segment Size (mss). It is logical to have a full mss as the multiplicative factor to the $tanh$ function so that in the best case it will achieve a complete mss increase. In other cases, mss will contribute to the outcome of $tanh$ function. Having $B_{rtt}$ and $M_{rtt}$ equal to $-1$ ensures that the *rtt* value will contribute towards window size increment while it has a value less than 1. When it reaches at 1, it will contribute 0 and afterward, it will contribute negatively. Previous research done to identify variability in TCP's round trip time [4] has found out that 90% of all the round trip time samples lie between $0.1s$ to $1s$. This is the reason we choose $M_{rtt}$ equal to $-1$. $B_{rtt}$ is used to make the result of $(rtt + M_{rtt})$ a negative value when $rtt$ exceeds the value of 1 (second).

*rto*: TCP maintains a timer to trigger any retransmission of packets. Whenever the timer expires, TCP retransmits the packet from the top of retransmission queue. The *rto* value depends on *rtt*. Initially the timer is set to a low value which is closer to the average round trip time. Setting the time to a very low value would cause unnecessary retransmission. Whenever there is a retransmission time out, TCP resends the packet from the queue and at the same time the value of *rto* is increased. Karn's [18] algorithm suggests a doubling of RTO each time the retransmission timer expires. With a higher value of *rto*, it is understandable that the TCP sending rate should be decreased as keeping the same sending window size which would create further congestion. The coefficients and the form of equation 2 used with *rto* value is the same as the equation for *rtt*, namely:

$$\frac{\delta w}{\delta rto} = A_{rto} + C_{rto} tanh(B_{rto}(rto + M_{rto})) \quad (5)$$

$$\frac{\delta w}{\delta rto} = 1452 \times tanh(-(rto - 1)) \quad (6)$$

$$(7)$$

$A_{rto}$ is set to 0 following the same reason as $A_{rtt}$ and the same explanation goes for setting $C_{rto}$ equal to $snd\_mss$. $M_{rto}$ is set so that it can contribute positively whenever the *rto* value is less than 1 and otherwise, whenever the *rto* value exceeds 1, $tanh$ will return a negative value which in turn will

 ISBN: 978-1-61208-186-1 

decrease the sending window size by contributing a negative value.

*rsnd*: The number of retransmitted packets plays an important role in TCP's congestion control mechanism. There is a maximum value set for this parameter in each TCP connection, where exceeding that value will cause the connection to terminate; we use the default value (12). Whenever there is a retransmission, the connection needs to be slowed to avoid further retransmission. The equation form for this parameter is defined as:

$$\frac{\delta w}{\delta rsnd} = A_{rsnd} + C_{rsnd} tanh(B_{rsnd}(rsnd + M_{rsnd})) \quad (8)$$

$$\frac{\delta w}{\delta rto} = 1452 \times tanh(-(rsnd - 0.5)) \quad (9)$$

The coefficients of this function are set similarly to the previous variables. $A_{rsnd} and C_{rsnd}$ are set to 0 and $mss$ as before. Here, $M_{rsnd}$ is set to $-0.5$ to have a minimum effect when there is no retransmission. But whenever there is a retransmission, it will start decreasing the window size immediately. The higher the number of retransmissions the greater the decrement will be. This ensures a lower sending rate to try to bring the connection into a steady mode.

Thus, the new value for *cwnd* is:

$$cwnd + \frac{\delta w}{\delta rtt} + \frac{\delta w}{\delta rto} + \frac{\delta w}{\delta rsnd} \quad (10)$$

There is a scope for fine tuning the coefficients and selecting appropriate parameters to get the maximum efficiency from this algorithm, but we do not explore that in this paper. The parameterized characteristics of STCP makes it more flexible and potentially more efficient in handling different types of scenarios, such as scenarios involving multimedia contents. The nature of the traffic and the condition of the network can be used to tune the coefficients. Events related to wireless environment congestion can be managed by tuning parameters which are solely related to wireless environment. This aspect of STCP means that it has the potential of improving the overall performance of TCP by adapting to network types and traffic contents. In particular, we are interested in understanding how the parameters can be selected and what are the best ways to choose the coefficients' initial values and how to tune them further.

## IV. Simulation Approach

Our specific interest is in understanding STCP in wireless networks and, in particular, traffic involving multimedia contents. As this is a comparative study of two different protocols (standard TCP and SmoothTCP), the research requires a platform where protocols can be implemented and observed in different scenarios. We use an available network simulator, OMNeT++ [19], [20]. OMNeT++ has been used for a variety of network research [21], [22], [23]. It also has available a well developed wireless library, INET [19], [20]. For our simulation, we needed to add several components and elements to OMNet++.

*Multimedia Content*: "Multimedia contents" will mean a bit stream and entail a continuous data transmission from server to client in response to a request from the client. Our "media files" were of specific sizes for the experiments and we only make use of dummy packets, not actual multimedia data, since we are only interested in the traffic delivery under differing network conditions. We do not need the client to process the data, but only to acknowledge the arrival, so it is enough to work with dummy data packets.

*Multimedia Client Application*: We use a client application which requests multimedia data from the media server (discussed below). It will only acknowledge the receipt of data and will not process it. From a technical point of view, a TCP client of the OMNeT++ simulator always issues a CLOSE command after receiving the response it requested. The client application in our case was modified so that it does not issue a CLOSE command unless the whole stream has been transmitted from server to client. After receiving the notification from server side about the end of stream, client application sends CLOSE command to close the connection.

*Multimedia Server Application*: An application on the server which receives requests and transmit the data. It stops only when it receives a CLOSE connection request from the client. Our server will use TCP and STCP, respectively.

*The Media Server*: A model of a server connected to our network that runs the Multimedia Server Application.

*Wireless environment*: A wireless environment will have wireless nodes and access points; it also has the unique features of a wireless network, including shared bandwidth, use of radio channels, variable transmitter power, etc. Successive experiments with different set of parameters and prior research [24] helped us determine the more significant parameters to study: the variables in the wireless environment we adjusted were the radio transmitter power and bit rate (described below).

More details on the simulation environment, the components modeled and their details, can be found in [25].

## V. Experiments and Results

Our simulation includes both wireless and wired network components. Our topology consisted of a single access point with one Ethernet interface and wireless network interface (802.11). The access point was connected to a router via a 10Mb/s wired connection. A multimedia server was connected to the router via a 100Mb/s wired connection. Two identical laptops with the client application moved within this environment. The simulation environment of OMNeT++ provides a rectangular "playground" which contains the devices and in which the laptops "move". Our laptops start their movement from opposite edges of the playground and travel straight towards the opposite side, then return to their original point and repeat this through the duration of the simulation. They move in a straight line with a speed of 10 meters/second.

 ISBN: 978-1-61208-186-1

The access point is stationary. Three factors are varied for the simulation:

- "Transmitter power": set to either 350mW or 450mW;
- "Bit rate": set to either 24Mbps and 54Mbps;
- "Data size": size of our multimedia file to be transferred: 30Mbytes and 60Mbytes.

Our basis of comparison is the time required to transmit all the data packets from server to the laptops and the number of retransmission time outs that occurs during the transmission. The transmission time results of the experiments are presented in Tables I and III and the results on the number of RTOs are presented in Tables II and IV.

| Bitrate | Power (mW) | Protocol | Avg. Time (sec) |
|---|---|---|---|
| 24Mbps | 350mW | TCP | 129.28 |
| 24Mbps | 350mW | STCP | 126.22 |
| 24Mbps | 450mW | TCP | 125.31 |
| 24Mbps | 450mW | STCP | 120.08 |
| 54Mbps | 350mW | TCP | 122.03 |
| 54Mbps | 350mW | STCP | 117.58 |
| 54Mbps | 450mW | TCP | 117.76 |
| 54Mbps | 450mW | STCP | 114.54 |

TABLE I
TIMING RESULTS FOR 30 MBYTES OF DATA.

| Bitrate | Power (mW) | Protocol | Avg. RTOs |
|---|---|---|---|
| 24Mbps | 350mW | TCP | 2344.5 |
| 24Mbps | 350mW | STCP | 2143.5 |
| 24Mbps | 450mW | TCP | 2348.0 |
| 24Mbps | 450mW | STCP | 2166.0 |
| 54Mbps | 350mW | TCP | 2346.0 |
| 54Mbps | 350mW | STCP | 2143.0 |
| 54Mbps | 450mW | TCP | 2350.0 |
| 54Mbps | 450mW | STCP | 2162.0 |

TABLE II
TIME OUTS FOR 30 MBYTES OF DATA.

| Bitrate | Power (mW) | Protocol | Avg. Time (sec) |
|---|---|---|---|
| 24Mbps | 350mW | TCP | 261.19 |
| 24Mbps | 350mW | STCP | 256.07 |
| 24Mbps | 450mW | TCP | 248.71 |
| 24Mbps | 450mW | STCP | 242.35 |
| 54Mbps | 350mW | TCP | 247.47 |
| 54Mbps | 350mW | STCP | 240.80 |
| 54Mbps | 450mW | TCP | 235.59 |
| 54Mbps | 450mW | STCP | 228.28 |

TABLE III
TIMING RESULTS FOR 60 MBYTES OF DATA.

For data of 30Mbytes and 60Mbytes, STCP consistently outperforms TCP, though by only a little in some cases. STCP also clearly outperforms TCP in the number of RTOs, substantially reducing the number of RTOs.

While looking at the results suggests that the differences in transfer time and in terms of RTOs might be attributable to STCP, it is certainly not clear. To understand the factors

| Bitrate | Power (mW) | Protocol | Avg. RTOs |
|---|---|---|---|
| 24Mbps | 350mW | TCP | 4683.5 |
| 24Mbps | 350mW | STCP | 4324.0 |
| 24Mbps | 450mW | TCP | 4690.5 |
| 24Mbps | 450mW | STCP | 4333.5 |
| 54Mbps | 350mW | TCP | 4694.5 |
| 54Mbps | 350mW | STCP | 4321.5 |
| 54Mbps | 450mW | TCP | 4693.5 |
| 54Mbps | 450mW | STCP | 4324.0 |

TABLE IV
TIME OUTS FOR 60 MBYTES OF DATA.

impacting the measured results, we performed an analysis of variance on the data. We considered the bitrate, power, protocol (TCP, STCP) and data size as factors and considered transmission time and number of timeouts as dependent variables. Thus, it was a four factor, two level analysis. For both the transmission time and the retransmission timeouts, the data size was the primary factor, explaining 98% of the variability. In looking at the raw results, this is clear - the data size dictates the time and number of timeouts.

We then considered the results separately for the different sizes of data, i.e., did separate analyses for the 30MB and 60MB experiments. Thus, each of these was a three factor, two level analysis. For the 30MB data size, the bit rate had the greatest impact on the variability, explaining 59% in the variability of transmission times. The difference in protocols explained about 18% of the variability and the transmission power explained about 22%. For the 60MB data size, the results were similar, with the bit rate explaining 49%, the transmission power 40% and the protocol used explaining 10%. In terms of transmission time, the impact of the protocol used was not the dominant factor. This is not surprising, since a higher bit rate would have a major impact on the transmission times. Thus, these experiments do not show much difference in the total times; larger files for longer durations may have to be examined.

However, when one considers the number of RTOs, the results are different. The analysis of variance for both the 30MB and 60MB data size files shows that the protocol accounts for almost all the variability (99%) in both sets of results. The other factors have no little impact (less than .5%) on the variability of the number of timeouts. There is a clear advantage for STCP over TCP in reducing the number of retransmission timeouts.

## VI. CONCLUSION

While the results presented in this paper are still early, they do show that STCP has some potential use in wireless environments. Even though other aspects of STCP have been studied elsewhere [25], there is still quite a bit to understand about its behavior. One of the advantages of STCP is that, as illustrated in the simulation, it works with TCP. In our simulation, only the multimedia server used STCP, other components, such as the laptops, used the standard TCP. Since

 ISBN: 978-1-61208-186-1

STCP adjusts the congestion window on the sender's side, it does not need to exist everywhere. This is a definite advantage.

The parameterized smoothing functions can also be used in other novel ways. As mentioned, parameters could be set by applications or by a server depending on context, such as settings for a video on demand server. These can then be changed as the environment changes, e.g. more users, etc. How to do this is unexplored. An alternative is to look at a more "dynamic" version of STCP, where the coefficients are dynamically changed depending on the network environment. We are currently exploring how this might be done. There is also a need to compare STCP to other TCP variants, such as RTP and UDP. This is something we are also exploring.

## REFERENCES


[1] I. Kim, Y. Kim, M. Kang, J. Mo, and D. Kwak, "TCP-MR: Achieving end-to-end rate guarantee for real-time multimedia," in *Proceedings of the 2nd International Conference on Communications and Electronics (ICCE)*, 2008, pp. 80–85.

[2] E. Vieira and M. Bauer, "Proactively controlling round-trip time variation and packet drops using smoothTCP-q," in *QShine'06: Proceeding of the 3rd International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks*, 2006, pp. 39–44.

[3] E. Vieira, "Smooth congestion control algorithms," PhD Thesis, The University of Western Ontario, 2006.

[4] E. Vieira and M. M. Bauer, "The variation in RTT of smooth TCP," in *Proceedings of the 3rd Consumer Communications and Networking Conference*, 2006, pp. 361 – 365.

[5] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," in *MOBICOM*, 1999, pp. 219–230.

[6] T. Dyer, D. Thomas, R. Boppana, and V. Rajendra, "A comparison of TCP performance over three routing protocols for mobile ad hoc networks," in *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM, 2001, pp. 56–66.

[7] K. Chandran, S. Ragbunathan, S. Venkatesan, and R. Prakash, "A feedback based scheme for improving TCP performance in ad-hoc wireless networks," in *Proceedings of the 18th International Conference on Distributed Computing Systems*, 1998, pp. 472–479.

[8] W.-T. Chen and J.-S. Lee, "Some mechanisms to improve TCP/IP performance over wireless and mobile computing environment," in *Proceedings of the 7th International Conference on Parallel and Distributed Systems*, 2000, pp. 437–444.

[9] A. Chodorek and R. Chodorek, *Applicability of TCP-friendly protocols for real-time multimedia transmission*, Faculty of Electronics and Telecommunicaitons, Poznan University of Technology, 2007.

[10] J. Shi, Y. Jin, H. Huang, and D. Zhang, "Experimental performance studies of SCTP in wireless access networks," in *Proceedings of the International Conference on Communication Technology*, 2003, pp. 392 – 395.

[11] R. Fracchia, C. Casetti, C.-F. Chiasserini, and M. Meo, "A wise extension of SCTP for wireless networks," in *IEEE International Conference on Communications*, 2005, pp. 1448 – 1453.

[12] A. Kumar, L. Jacob, and A. L. Ananda, "SCTP vs TCP: Performance comparison in MANETs," in *Proceedings of the 29th IEEE International Conference on Local Computer Networks*, 2004, pp. 431 – 432.

[13] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RFC 3550: RTP: A Transport Protocol for Real-Time Applications*, IETF Network Working Group, July 2003.

[14] H. Schulzrinne, A. Rao, and R. Lanphier, "Rfc 2326: Real time streaming protocol (rtsp)," IETF Network Working Group, United States, 1998.

[15] M. Allman, V. Paxson, and W. Stevens, *RFC 2581: TCP Congestion Control*, IETF Network Working Group, April 1999.

[16] D. Bansal and H. Balakrishnam, *TCP-Friendly Congestion Control for Real-time Streaming Applications*, May 2000.

[17] V. Jacobson, "Congestion avoidance and control," in *SIGCOMM '88: Symposium Proceedings on Communications Architectures and Protocols*. New York, NY, USA: ACM, 1988, pp. 314–329.

[18] P. Karn and C. Partridge, "Improving round-trip time estimates in reliable transport protocols," in *SIGCOMM '87: Proceedings of the ACM Workshop on Frontiers in Computer Communications Technology*. New York, NY, USA: ACM, 1987, pp. 2–7.

[19] OMNeT++, "http://www.omnetpp.org/index.php."

[20] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1–10.

[21] A. Varga, "Using the OMNeT++ discrete event simulation system in education," *IEEE Transactions on Education*, vol. 42, no. 4, pp. 11–11, November 1999.

[22] J.-C. Maureira, O. Dalle, and D. Dujovne, "Generation of realistic 802.11 interferences in the OMNET++ INET framework based on real traffic measurements," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 1–8.

[23] M. Bredel and M. Bergner, "On the accuracy of IEEE 802.11g wireless LAN simulations using OMNeT++," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST, Brussels, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 1–5.

[24] K. Rmachandran, R. Kokku, H. Zhang, and M. Gruteser, "Symphony: Synchronous two-phase rate and power control in 802.11 WLANs," in *Proceedings of the 6th International Conference on Mobile Systems, Applications and Services (MobiSys'08)*. New York, NY, USA: ACM, 2008, pp. 132–145.

[25] M. Islam, "Smooth TCP: A solution for wireless multimedia communication," MSc Thesis, The University of Western Ontario, 2010.


 ISBN: 978-1-61208-186-1 