A Fuzzy Controller for TCP Improvement over Mobile Networks

Sara Raad Qasim
Computer and Control- Electrical Department
University Of Baghdad

Zainab T. Alisa, phD
Baghdad University
Iraq, IEEE member

ABSTRACT
The Transmission Control Protocol (TCP) is used for reliable delivery of data over unreliable networks. Practically, most TCP mechanisms have been carefully designed for wired networks. Neglecting the characteristics of wireless environments can lead to TCP implementations with poor performance. In order to use TCP in mobile networks, improvements have been proposed in this paper to enhance TCP algorithm to distinguish between the different types of loss events. In mobile or static wireless environments, losses are not always due to network congestion, as in the case of wired networks. In this paper, a modified algorithm is presented using fuzzy controller to differentiate the loss events (error loss from congestion loss) that intend at adapting TCP to mobile and static wireless environments with better performance. Simulation results were performed using OMNET simulator and have showed that the new proposal has better throughput than other TCP schemes.

Keywords
ssthresh, cwnd, westwood, FLC, RTT, fwestwood

1. INTRODUCTION
Wireless Networks are complex systems that may contain mobile or static nodes that can organize themselves freely and dynamically. In this way they form randomly, and temporary wireless networks topologies, allowing devices to seamlessly interconnect in areas with no pre-existing infrastructure. Recently, many research efforts have been put on the new challenging wireless environment.

TCP (Transmission Control Protocol) [1] was designed to provide reliable end-to-end data delivery over unreliable networks. In theory, TCP should be independent of the technology of the underlying environment infrastructure. In particular, TCP should not care whether the Internet Protocol (IP) is running over wired or wireless connections. In practice, it does matter because most TCP algorithms have been designed based on assumptions that are in the context of wired networks. Neglecting the properties of wireless transmission lead to poor performance of TCP implementations.

In wireless networks, the principal problem of TCP lies in performing congestion control in case of losses that are not induced by network congestion. Since bit error rates are very low in wired networks, nearly all TCP versions nowadays (Tahoe, Reno, NewReno, and Vegas) assume that packets losses are due to congestion. Consequently, when a packet is found to be lost, either by timeout or by multiple duplicated ACKs, TCP slows down the sending rate by adjusting its congestion window. Unfortunately, wireless networks suffer from several types of losses that are not related to congestion, making TCP not adapted to this environment. Numerous improvements and optimizations have been proposed over the last few years to enhance TCP performance over wireless networks. TCP versions: Tahoe, Reno, NewReno, and Vegas perform differently in wireless networks [2]. However, all these versions suffer from the same problem of inability to distinguish between packet losses due to congestion from losses due to the specific features of Ad hoc networks. While TCP Westwood was basically designed to perform in wireless environments but it fails to achieve high performance at high error rate.

2. TRANSMISSION CONTROL PROTOCOL
TCP is a window-based acknowledgement-flow control protocol [3]. It uses additive-increase multiplicative decrease strategy for changing its window as a function of network conditions. Packets of a TCP connection are sent with increasing consecutive sequence numbers. In the simplest operation of TCP, at each arrival of a packet at the destination, an ACK is sent back to the source with the information of the next sequence number that is expected. Thus if all packets up to packet n-1 have reached the destination, then the last arrival will trigger an ACK with sequence number n. If a packet n is lost in the network and packet n + i, i = 1; 2; 3; arrives at the destination, then each of these packets will trigger an Acknowledgement indicating that the destination is expecting packet n. These are called duplicated ACKs. In the absence of losses, starting from one packet or from a larger value, the window is increased exponentially by one packet every non-duplicate ACK until the source estimate of network capacity is reached. This is the Slow Start (SS) phase, and the capacity estimate is called the slow start threshold (ssthresh). In most versions of TCP (Tahoe, Reno and New Reno) once ssthresh is reached, the source switches to a slower increase in the window by one packet for every window's worth of ACKs. This phase is called Congestion Avoidance (CA) phase [4]. The window increase is interrupted when a loss is detected. Two mechanisms are available for the detection of losses: the expiration of a retransmission timer (timeout), or the receipt of three duplicate ACKs (the latter is called the fast retransmit (FRXT) phase. The source then sets its estimation of the capacity to half the current window; this action is due to the fact that when these TCP versions have been developed, losses were indication of congestion as TCP was then deployed only over wireline networks.

3. OVERVIEW OF TCP VERSIONS
Tahoe [5], the first version of TCP to implement congestion control, at this point sets the window to one packet and enter the slow start phase to reach the new sstresh. Slow starting after every loss detection deteriorates the performance given the low bandwidth utilization during SS. When the loss is detected via timeout a more drastic reaction is taken as a more drastic congestion is understood to occur, since the ACK stream has stopped. In the FRXT case, ACKs still arrive at the source, and losses are recovered without SS. This is the behavior of the newer versions of TCP (Reno, NewReno, SACK, etc.) that call a Fast Recovery (FRCV) algorithm to

retransmit the losses while maintaining enough packets in the newtork to preserve the ACK clock. Once the losses are recovered, this algorithm ends and normal CA is called. If FRCV fails (to recover the losses), the ACK stream stops, a timeout occurs, and the source resorts to SS as with Tahoe. Among the TCP versions that use the FRCV, the difference is in the estimation of the number of packets in the flight during FRCV.

Reno version [6] considers every duplicate ACK a signal that a packet has left the network. The problem of Reno is that it leaves FRCV when an ACK for the first loss window is received. This prohibits the source from detecting the other losses with FRXT. A long timeout is required to detect the other losses.

NewReno version [7] has been proposed to overcome Reno's problem. The idea is to stay in FRCV until all the losses in the same window are recovered. Another problem of Reno and NewReno is that they rely on ACKs to estimate the number of packets in flight. ACKs can be lost on the return path, which results in an underestimation of the number of packets that have left the network. More information is needed to estimate more precisely the number of packets in the pipe. This information is provided by the selective ACK (SACK) [68], a TCP option containing the three blocks of contiguous data most recently received at the destination.

Vegas version [8] aims to decouple congestion detection from losses. In TCP Vegas, the RTT of the connection and the window size are used to compute the number of packets in the network buffers. The window is decreased when this number exceeds a certain threshold and is increased when it is below some threshold. In other words, Vegas also use delay as a congestion indication and then reacts to reduce its throughput.

Westwood version [9] improves the behavior of TCP Reno in wired and wireless networks. The enhancement is significantly occurred in wireless networks with lossy links. In fact, TCP Westwood performance is not very aggressive to random errors, while TCP Reno is equally sensitive to error loss and congestion loss and cannot differentiate them. Hence; the tendency of TCP Reno is to overreact to errors. The key innovative idea is to keep measure at the TCP sender side the bandwidth used by the connection by checking the returning ACKs rates. The estimate is then used to update congestion window (cwnd) and slow start threshold (ssthresh) after congestion detection, that is, after three duplicate acknowledgments or after a timeout. This mechanism avoids the blind halving of the sending rate of TCP Reno after packet losses and enables TCP Westwood to select ssthresh and cwnd which is consistent with the estimated bandwidth used at the time congestion is experienced. This mechanism is called faster recovery.

4. RELATED WORKS

FLC (Fuzzy Logic Controller) has many applications to control network congestion since 1990. In early stage, it was used to do rate control in ATM network, e.g., [10], [11], to guarantee the QoS (Quality of Service). These control algorithms are explicit in nature, and they depend on absolute queue length (the maximum buffer size) to adjust the allowed sending rate. Nevertheless, these early designs have various shortcomings including cell loss (even though cell loss is used as a congestion signal to compute the rate factor, e.g., [12]), queue size fluctuations, poor network latency, stability and low utilization. Later, FLC was used in Random Early Detection (RED) algorithm in TCP/IP networks, e.g., [13], [14], to reduce packet loss rate and improve utilization. However, they are still providing implicit or imprecise congestion signaling, and therefore cannot overcome the throughput fluctuations and conservative behavior of TCP sources.

5. THE PROPOSED TCP CONGESTION CONTROL ALGORITHM

In this paper, a brief description on the proposed functional fuzzy system is presented. More details on it can be shown in our previous work [16]. In [16]; a fuzzy controller for Westwood version is implemented and used for wire network. This proposal is tested in this work for mobile network.

It consists of three inputs, two outputs and nine rules that aggregated in a disjunctive manner.

5.1 Inputs of fuzzy system

The inputs of fuzzy controller are:

i. *Delay or RTT (Round Trip Time)*

This is the first input, it is the time required for a packet to go from a source to a destination and then back again in the form of acknowledgement. It is compared with RTT_MIN (minimum value of RTT) and RTT_MAX (maximum value of RTT) to get the state of the network overloaded or has error. The value of RTT_MIN and RTT_MAX are read during the execution of the program. The membership function for this input is shown in Figure 2.

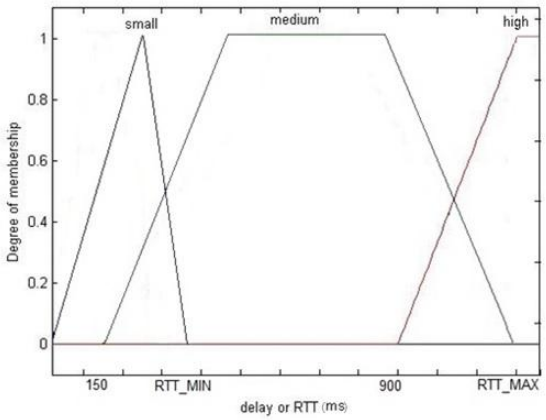

Fig. 2: Fuzzy set for fuzzy variable delay or RTT

ii. The ratio of the number of timeouts to the number of 3dupacks (ratio)

This is the second input, if the ratio is very small (in between 0.01 to 0.2), the observation shows that this event has been caused by a bit error event, not by congestion. If the ratio is high (e.g. greater than 0.5) then the event is more likely due to congestion. The membership function for this input is shown in Figure 3.

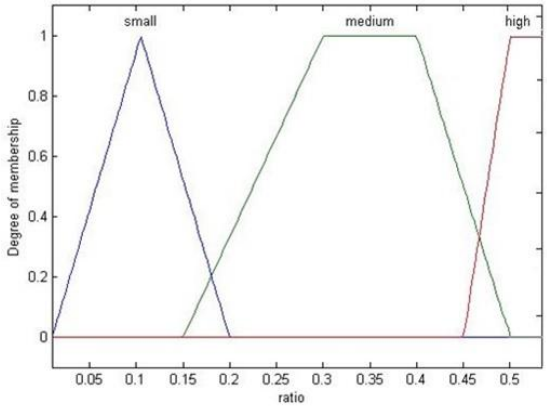

Fig. 3: Fuzzy set for fuzzy variable ratio

5.2 Rules of Fuzzy System

The proposed system includes 9 rules that are aggregated in a disjunctive manner, as in the Table 1. These rules were obtained after tuning the two inputs (RTT and ratio) to get the best results.

Table 1 Rules of the Fuzzy System [16]

Delay	Ratio	Outputs Phase
Small	Small	Same State
Small	Medium	Same State
Small	High	Congestion Avoidance
Medium	Small	Same State
Medium	Medium	Congestion Avoidance
Medium	High	Slow Start
High	Small	Same State
High	Medium	Congestion Avoidance
High	High	Slow Start

5.3 Outputs of Fuzzy System

Takagi-Sugeno Fuzzy System is used in this paper, which used two output functions that do not have an associated membership function. The outputs are ssthresh and cwnd which specify the new phase of TCP to trigger after the packet loss event. These outputs are computed by the defuzzification method "center average" and the Eq. (1) for two outputs will be:

$$y_r = \frac{\sum_{r=1}^{2} \sum_{i=1}^{R} b_i \mu_i}{\sum_{i=1}^{R} \mu_i} \quad (1)$$

Where r represent the output number (r=1 for ssthresh and r=2 for cwnd), i represents the on rule (the rule that represents the current situation), R is the number of rules, b_{ir} represent the output equations (explained in [16]) and μ_i represents the certainty of a premise of a rule and thereby represents the degree of the membership function for the on rule.

6. PERFORMANCE EVALUATION OF TCP FWESTWOOD

OMNET++ simulation IDE [17], [18], [19] is used to evaluate the performance of the proposed system in terms of the number of unique segments transmitted by the sender and the throughput of a connection. Two scenarios were built to prove TCP proposal and compare the results with various kinds of TCP as TCP Reno, TCP Tahoe and TCP Westwood.

6.1 Scenario 1/Wireless Network (with static nodes)

Figure 5 shows the network used for the simulation. The link has 10Mbps data rate and 45ms delay.

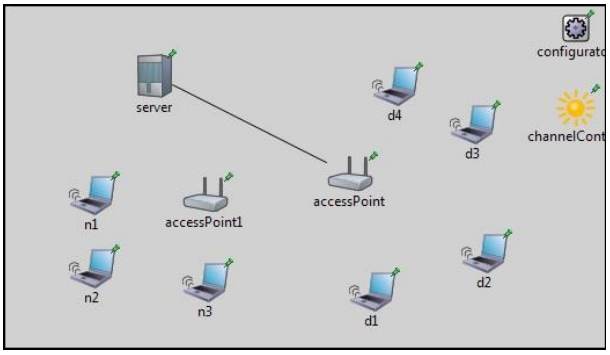

Fig.5: Network setup of scenario 1

The following applications are generated:

- TCP application between server and d2.
- UDP application between d1 and server.
- UDP application between d3 and d2.
- TCP application between n1 and server.
- TCP application between n3and d3.

The simulation has run separately for FWestwood and other TCP scheme (Tahoe, Reno and Westwood) for 100s for different error rates (packet error rate are changed between 1%-10%). Figure 6 show the number of transmitted segments of FWestwood, Westwood, Reno and Tahoe while Figure 7 present the throughput comparisons among all TCPs.

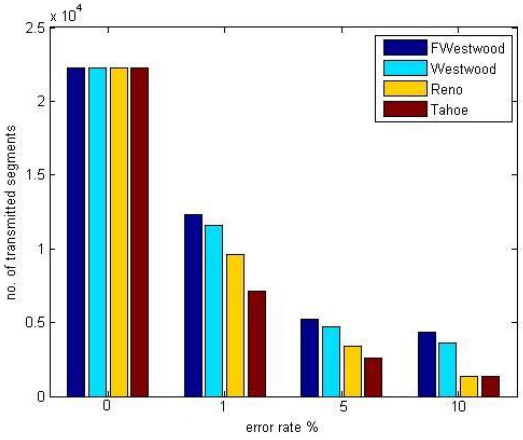

Fig. 6: The number of transmitted segments of scenario 1

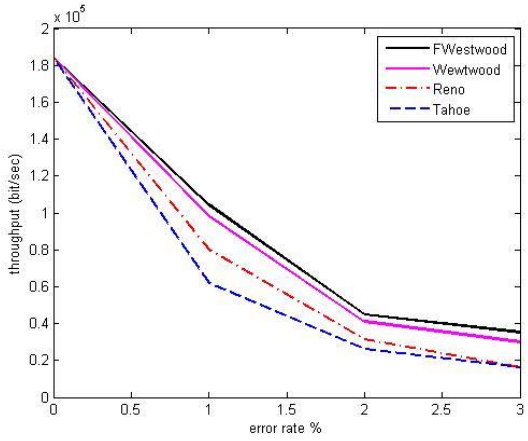

Fig. 7: Throughput comparison of scenario 1

6.2 Scenario 2/Mobile Network

In this scenario, FWestwood is used as the main algorithm in all related nodes so that to control the congestion and packet loss. Also the node's movements are chosen to be random with random speed. Figure 8 show the connected network.

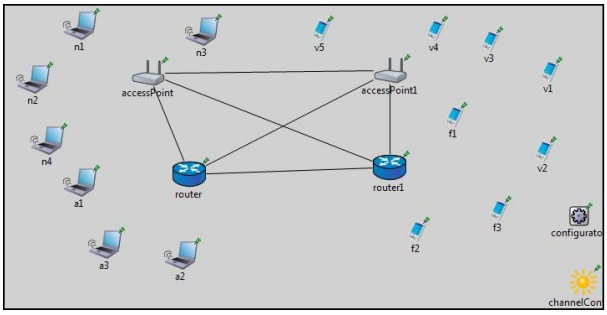

Fig. 8: Network of Scenario 2

All the links have data rate=10Mbps. The simulation is run for 100s for FWestwood, Westwood, Reno and Tahoe separately and the transmitted segments with the throughput are measured and for each TCP. The applications used in this scenario are:

- TCP Application between n1 and a1.
- TCP Application between n2 and v4.
- TCP Application between n3 and a3.
- TCP Application between n4 and f3.
- TCP Application between f3 and n4.
- TCP Application between a1 and v5.
- TCP Application between a3 and f1.
- UDP Application between n1 and a3.
- UDP Application between n3 and v1.
- UDP Application between f1 and a2.

The speed of the mobile nodes are chosen randomly between (20mps, 50mps) where mps=meter per second and the moving environment is (400mx600m). The results are shown in Figure 9 and Figure 10.

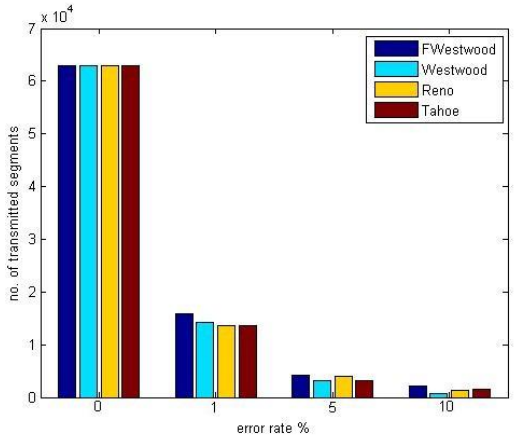

Fig. 9: The number of transmitted Segments of Scenario 2

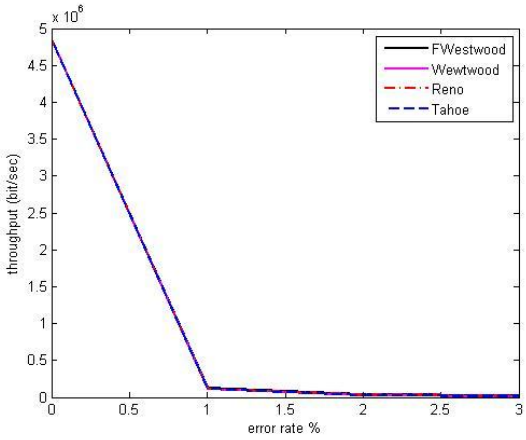

Fig. 10: Throughput Comparison of Scenario 2

6.3 Simulation Results and Analysis

Figures 6 presents the number of transmitted segments for scenario 1. As the error rate increases, TCP FWestwood transmits significantly more segments than Reno, Tahoe, and Westwood. This behavior of FWestwood showed that it is less aggressive than other TCP schemes when the random wireless link error rate is increased. The numerical results of the transmitted segments are listed in Table 2. In Figure 7, the throughput among different TCP schemes is presented. Since FWestwood is based on a precise and stable reaction upon loss events then it will make use of resources more than other TCPs and thus transmit more data and achieve higher throughput.

Table 2 Numerical values of the transmitted segments number of scenario 1

Error Rate (%)	Reno	Tahoe	Westwood	FWestwood
0	22286	22286	22286	22286
1	9589	7111	11589	12339
5	3363	2622	4713	5225
10	1326	1351	3602	4316

Values obtained in Table 2 shows the priority of FWestwood over other TCP variants. This conclusion have confirmed that the proposed schemes have resulted in significant performance improvement over other TCP standards since it retain congestion window as near as possible to the value when loss event occurs.

In mobile network, there is possible causes of packet losses, Other than congestion, include wireless link errors, channel contention, and link breakages due to node mobility. For all these reason there was no expansion in this field other than this scenario since it needs to take many parameters into account, for instance network partition, multipath routing and all MAC layer problems.

The results in Figure 9 and Figure 10 show that all TCP scheme perform closely to each other at small error rate but the difference in behavior among FWestwood and the other standards is increased as error rate become 1% and upper. This result is due to FWestwood ability to take the best decision (by the help of fuzzy controller) and check the reason of the loss event and then react. As a result FWestwood sends more data and achieve higher throughput as shown clearly in Table 3.

Table 3 Numerical values of transmitted segments of scenario 2

Error Rate (%)	Reno	Tahoe	Westwood	FWestwood
1	13670	13670	14139	15930
5	4059	3230	3148	4275
10	1305	1537	670	2222

The results in the table above ensure that FWestwood detects segment losses due to bit error with better precision and keeps a steady flow of segments towards the destination to ensure a good throughput. Again in real congestion erroneous environment, FWestwood does not behave aggressively and hence do not worsen the congestion in the network. This behavior is very significant feature in FWestwood.

7. CONCLUSIONS

In this paper, technique has been adapted and applied to increase TCP performance in mobile networks. Through simulation evaluation it has been demonstrated, under widely differing operating conditions and environments, that the FWestwood algorithm is better than all other TCP variants for mobile network in addition to immobile network.

The new novel traffic management scheme, FWestwood , has proved its ability to manage the Internet congestion and to assure the use of resources efficiently for different applications. The controller is designed by paying attention to the disadvantages as well as the advantages of the existing congestion control protocols. As a distributed operation in networks, the fuzzy controller uses the available parameters in the network (delay or RTT, the number of timeout and the number of 3dupack) to effectively throttle the source sending rate with better stabilized reaction. Unlike the existing explicit traffic control protocols that potentially suffer from performance problems or random degradation in the transmission rate, FWestwood overcomes those fundamental deficiencies and check the reason of the packet loss before take any action. To verify the effectiveness and superiority of FWestwood, extensive experiments have been conducted in OMNET simulator. In addition to the feature of the FLC being able to intelligently tackle the nonlinearity of the traffic control systems, the success of this controller is also attributed to the careful design of the fuzzy logic elements which occurred obviously in the simulation results.

As a future work two TCP schemes may be taken in cooperation with fuzzy controller in order to improve their algorithm in high speed networks.

8. REFERENCES

[1] Postel, J. 1981.Transmission control protocol, RFC793.

[2] Xu, S. and Saadawi, T. 2002. Performance evaluation of TCP algorithms in multi-hop wireless packet networks, Journal of Wireless Communications and Mobile Computing, Vol. 2, no. 1, pp. 85–100,.

[3] Wikipedia organization, [Online] Available from: http://en.wikipedia.org/wiki/Sliding_window_protocol, [Accessed: April 2014].

[4] TCP Congestion avoidance algorithm [Online] Available from: http://en.wikipedia.org/wiki/TCP_congestion-avoidance-algorithm, [Accessed: April 2014].

[5] Jacobson, V. 1988. Congestion avoidance and control, in Proc. of ACM SIGCOMM, Vancouver, Canada.

[6] Fall, K. and Floyd, S. 1996. Simulation based comparisons of Tahoe, Reno, and Sack TCP, in Computer Communications review.

[7] Floyd, S., Henderson, T and Gurtov, A. 2012. The New Reno Modification to TCP's Fast Recovery Algorithm, RFC 3782.

[8] Brakmo, L. S. & Peterson, L. L., 1995. TCP Vegas: End to End Congestion Avoidance on a Global Internet, IEEE Journal on Selected Areas in Communication, Vol. 13, no. 8.

[9] Casetti, C., Gerla, M. and Mascolo, S. 2002. TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks, Kluwer Academic Publishers, Wireless Networks, pp. 467–479.

[10] Chang, C. and Cheng, R. 1994. Traffic control in an ATM network using fuzzy set theory, in *Proc. IEEE INFOCOM*, vol. 3. pp. 1200–1207.

[11] Harju, J. and Pulakka, K. 1999. Optimization of the performance of a rate based congestion control system by using fuzzy controllers, in *Proc. IEEE IPCCC*, pp. 192–198.

[12] Chang, R. and Cheng, C. 1996. Design of fuzzy traffic controller for ATM networks, *IEEE/ACM Trans. Netw.*, vol. 4, no. 3, pp. 460–469.

[13] Aoul, H,. Nafaa, A., Negru, D. and Mehaoua, A. 2004. FAFC: fast adaptive fuzzy AQM controller for TCP/IP networks, in *Proc. IEEE GLOBECOM*, vol. 3, pp. 1319–1323.

[14] Chrysostomou, C., Pitsillides, A. and Hadjipollas, G. 2003. Fuzzy explicit marking for congestion control in differentiated services networks, in *Proc. IEEE Int. Symp. Computers Commun.*, vol. 1, pp. 312–319.

[15] Passino, K. M. and Yurkovich, S. 1998. Fuzzy Control, Addison Wesley Longman Inc.

[16] Qasim, S. R., and Alisa, Z. T., 2014. A Fuzzy based TCP Congestion Control for Wired Networks, International Journal of Computer Applications (0975 – 8887) Vol. 89, no.4.

[17] Varga, A., and OpenSim Ltd., 2011. OMNeT++, User Manual, Version 4.3.

[18] Varga, A., and OpenSim Ltd., 2011. OMNeT++, User Guide, Version 4.3.

[19] www.omnetpp.org.

9. List of Abbreviation

cwnd	Congestion Window
dupack	Duplicate Acknowledgement
FWestwood	Fuzzy controller with Westwood
Mbps	Megabits per second
mss	Maximum Segment Size
QoS	Quality of Service
RTT	Round Trip Time
RTT_MAX	Round Trip Time Maximum
RTT_MIN	Round Trip Time Minimum
ssthresh	Slow Start Threshold
TCP	Transmission Control Protocol
UDP	User Datagram Protocol