A Fuzzy based TCP Congestion Control for Wired Networks

Zainab T. Alisa
Assist. Prof, Ph.D.
Baghdad University, Iraq

Sara Raad Qasim
Electrical Eng. Department,
University Of Baghdad, Iraq

ABSTRACT
Since TCP cannot recognize bit error loss event from congestion loss event, it fails to work well in wired networks with large random error rate. In this paper, a modification in TCP Westwood congestion control algorithm is proposed by using a fuzzy controller to enhance its performance in wired networks with high error rate. The number of timeout events and the number of triple duplicate acknowledgement (also called 3dupacks) is counted to measure their ratio as the first input to a fuzzy system so that to differentiate congestion loss from bit error loss. Also the time difference between the last two timeout events is taken as the second input to the fuzzy system to check whether the timeout events are due to congestion or non-congestion event. The delay or RTT (Round Trip Time which is the time from transmitting a segment till receive an acknowledgement) also considered in the fuzzy system as the third input. The proposed TCP Fuzzy is tested using OMNET++ IDE simulator and found that it gives better performance than TCP standards in wired networks when error rate is increased.

1. INTRODUCTION
Popular protocols, such as TCP, are designed and implemented for wired networks but they fail to perform well in networks with high error rates since it treats all loss events due to congestion such that it decreases its transmission rate when any loss occur. The congestion control algorithm used in the TCP/IP protocol suite [1], [2], [3] is a sliding window mechanism that uses segment loss as a sign to congestion. The TCP sender probes the state of the network by gradually increasing the window of segments that are outstanding in the network until the network drops segments and become congested. Initially, the increase is exponential and this phase is called "Slow-start". This phase is intended to quickly take all the available bandwidth. When the window size reaches a slowstart threshold (called ssthresh), TCP starts the second phase called "Congestion Avoidance" [4], where the increase is linear. Clearly, it is desirable to set the threshold to a value that approximates the connection's "fair share". The optimal value for the slowstart threshold is the one that corresponds to the number of segments in flight in a pipe when TCP transmission rate is equal to the available bandwidth [5], i.e. when its transmission window is equal to the available bandwidth-delay product.

The rest of the paper is organized as follows. A brief description of TCP congestion control is presented in section 2. Related research works are discussed in Section 3. A brief overview of fuzzy controller is presented in section 4. The proposed new TCP Westwood congestion control algorithm is presented in Section 5. Detailed performance analysis of the proposed algorithm with the help of OMNET++ IDE simulator is presented in Section 6. The conclusions of the paper with some hints on the future research in this direction are discussed in section 7.

2. TCP CONGESTION CONTROL
When a loss occurs either through triple duplicate acknowledgements, or through the expiration of the retransmission timer, the connection backs off by shrinking its congestion window.

In TCP Tahoe, an RTO (Retransmission TimeOut) is an indication of congestion and enters congestion avoidance phase by setting congestion window (cwnd) to 1 and slow start threshold (ssthresh) to half of cwnd. Cwnd is increased additively till it reaches ssthresh, then it is increased linearly until a packet loss is encountered.

TCP Reno [1] retains the basic principle of Tahoe, but uses the logic of triple duplicate acknowledgements (3dupacks) to trigger Fast Retransmit. After 3 dupacks, TCP Reno takes it as a sign of segment lost and retransmits the packet immediately and enters Fast Recovery. In Fast Recovery, ssthresh and cwnd is set to half the value of the current cwnd. For each subsequent dupack, the cwnd is increased by one and a new segment is transmitted if the new value permits it. TCP Reno cannot detect multiple packet loss within the same window.

TCP NewReno [6] is able to detect multiple losses within the same window with small modification to Reno. TCP NewReno does not exit fast recovery mode until all the data that was outstanding at the time it entered Fast Recovery is acknowledged. Thus it does not reduce cwnd multiple times.

The Fast Recovery phase proceeds as in Reno. When a fresh ACK is received then there are 2 cases:

- If it ACKs all the segments which were outstanding when entered Fast Recovery, then it exits Fast Recovery and sets cwnd and ssthresh and continues congestion avoidance as in Tahoe.
- If the ACK is partial then it deduces that the next segment in line was lost and retransmits that segment and sets dupacks to 0. It exits Fast Recovery when all the data segments in the window are acknowledged.

TCP New Reno suffers from the fact that it takes one RTT to detect each packet loss. When the ACK for the first retransmitted segment is received, only then it is possible to deduce which other segment was lost.

TCP Westwood (TCPW) [7] improves the performance of TCP Reno in wired as well as wireless networks. The improvement is most significant in wireless networks with lossy links. In fact, TCPW performance is not very sensitive to random errors, while TCP Reno is equally sensitive to random loss and congestion loss and cannot distinguish between them. Hence, the tendency of TCP Reno to overreact to errors. The key innovative idea is to continuously measure at the TCP sender side the bandwidth used by the connection

via monitoring the rate of returning ACKs. The estimate is then used to compute congestion window and slow start threshold 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 a slow start threshold and a congestion window which is consistent with the effective bandwidth used at the time congestion is experienced. This mechanism is called faster recovery.

TCP, which was originally designed for wired network, does not work efficiently with the increase of error rates. TCP assumes packet losses are due to congestion in the path and reacts accordingly. However, in erroneous networks the packet loss is not always due to congestion. This problem is become worse when sporadic losses occur. Random losses are losses not caused by congestion at the links. In this case, a lost segment is misinterpreted by a TCP sender as a sign of congestion, and dealt with by decreasing the sender's window. Such action, clearly, does not handle the random loss condition nicely and it merely results in reduced throughput. The larger the bandwidth-delay product, the larger the performance degradation caused by such action.

In networks with random error rates, a good percentage of timeouts and reception of out of order segments have been happened due to the bit error rather than congestion. In such cases, reducing transmission rate does not help as this action results in under-utilization of network bandwidth. The basic TCP congestion control algorithm cannot distinguish between congestion and bit error timeouts; as a result it fails to give good performance in such networks.

In this paper, a new TCP Westwood congestion control algorithm is proposed, TCP FWestwood, by using a fuzzy controller in order to get better performance in wired networks. Several counters at the sending host were used in order to record the frequencies on retransmission timeout and duplicate acknowledgement. This statistics with the help of fuzzy controller used to differentiate non-congestion event from congestion event and to take appropriate action at a specific event. Experimental result shows that the algorithm ensures good throughput for connections that incorporate erroneous links.

3. RELATED WORKS

In [8], Balakrishnan *et al.* proposed the design and implementation of a simple protocol called "Snoop" for the scenario where a fixed host is communicating with a mobile host with the help of a base station. The network layer code at the base station is changed to implement the Snoop protocol. The packets sent from the fixed host are buffered at the base station before delivering them to the mobile host. When the Snoop agent residing at the base station receives a duplicate acknowledgement against a lost packet at the mobile host, it retransmits the missing packet locally to the mobile host and conceals the packet loss events from the sender and hence prevents it from reducing its congestion window to maintain a good throughput.

In [9] Cen et al. proposed a hybrid loss discriminator ZBS which uses three loss discriminators: ZigZag [9], Biaz [10], and Spike [11]. ZBS (the name is taken from the first letters of Zigzag, Biaz and Spike) is used at the receiver. ZigZag and Biaz are based on the interarrival of the packets at the receiver and the number of losses detected. Spike is based on the relative one way trip time. ZBS dynamically switches between the three losses discriminators according to observed network conditions. The accuracy of ZBS discriminator depends on the number of flows sharing the bottleneck and yields high congestion misclassifications.

Indirect-TCP (I-TCP) [12] uses TCP splitting approach with different flow control and congestion control mechanism on the wireless and wired parts of a network. The TCP connection is splitted into two connections at the point where the wired networks meet, i.e. at the base station. The base station keeps one TCP connection with the fixed host, while it uses another connection with a protocol specially designed for better performance over wireless links for the mobile host. The base station acknowledges the segments as soon as it receives them. I-TCP provides faster adaptation to mobility and wireless link breaks. However, I-TCP breaks end-to-end TCP ACKs and puts extra load on the base station. It is not a scalable solution.

El Khayat et al. [13] worked with applying machine learning (ML) towards improving TCP over wireless connections. The idea is to use data available inside of the state machine as variables to feed into a ML algorithm that powers a packet loss classifier. By lowering the number of registered LE (denote a loss due to a link error) on wireless networks (and simultaneously maintaining the correct C (denote a loss due to a congestion) for TCP- friendliness on wired networks.

Wozniak [16] classifies packet losses using Fuzzy system as due to congestion or random collision and react only to those losses perceived as being caused by network congestion and seeks to use environmental variables available to TCP implementations to feed a fuzzy inference system(such as Delay and InterArrival). Fuzzy inference systems also allow problems to be defined in an intuitive way using a propositional IF-THEN rule base.

Liu [17] proposes a distributed traffic management framework, in which routers are deployed with intelligent Fuzzy controllers to tackle the traffic mass. This fuzzy-logic-based controller can measure the router queue size directly; hence it avoids various potential performance problems arising from parameter estimations while reducing much consumption of computation and memory resources in routers.

4. FUZZY SYSTEM

While traditional logic contains only two truth values (true and false), fuzzy logic may contain an infinite number of truth values on the continuous range [0,1].

The fuzzy controller [14] is composed of the following four elements (fig.1):

1. A rule-base (a set of If-Then rules), which contains a fuzzy logic quantification of the expert's linguistic description of how to achieve good control.

2. An inference mechanism (also called an "inference engine" or "fuzzy inference" module), which emulates the expert's decision making in interpreting and applying knowledge about how best to control the plant.

3. A fuzzification interface, which converts controller inputs into information that the inference mechanism can easily use to activate and apply rules.

4. A defuzzification interface, which converts the conclusions of the inference mechanism into actual inputs for the process.

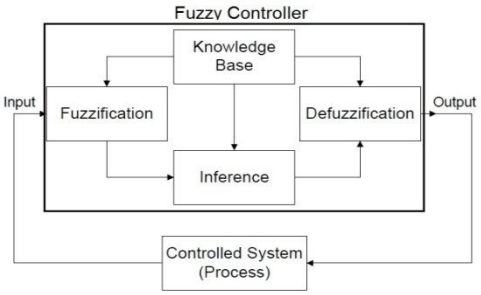

Fig 1: The Fuzzy Logic Controller structure

The fuzzy system is divided into two main types, Standard Fuzzy Systems and Takagi-Sugeno Fuzzy Systems (also called functional fuzzy system).

For the functional fuzzy system, it uses singleton fuzzification, and the ith rule has the form :

If $\tilde{\alpha}_1$ is $\check{A}^j_1$ and $\tilde{\alpha}_2$ is $\check{A}^k_2$ and, . . . , and $\tilde{\alpha}_r$ is $\check{A}^l_r$ Then $b_i = g_i(\bullet)$

Where $\tilde{\alpha}_1$ to $\tilde{\alpha}_r$ are the linguistic variable for the input, $\check{A}^j_1$ to $\check{A}^l_r$ are the fuzzy set, "•" simply represents the argument of the function g_i and the b_i are not output membership function centers.

The consequents of the rules are different, however. Instead of a linguistic term with an associated membership function, in the consequent a function $b_i = g_i(\bullet)$ is used (hence the name "functional fuzzy system") which does not have an associated membership function. Notice that often the argument of g_i contains the terms u_i, i = 1, 2, . . . , n, but other variables may also be used. The choice of the function depends on the application being considered. For instance, it may have the form:

$b_i = g_i(\bullet) = a_{i,1} + a_{i,2}(u_1)^2 + \bullet\bullet\bullet + a_{i,r}(u_r)^2$ …….. (1)

More information on fuzzy system can be shown in [15].

Fuzzy control systems have been utilized in many areas such as the industrial and scientific Projects, in parallel processor systems and in networks applications.

5. THE PROPOSED TCP CONGESTION CONTROL ALGORITHM

In this paper, the proposed functional fuzzy system consists of three inputs that are represented using triangular membership function (Fig. 2), two outputs and nine rules that aggregated in a disjunctive manner.

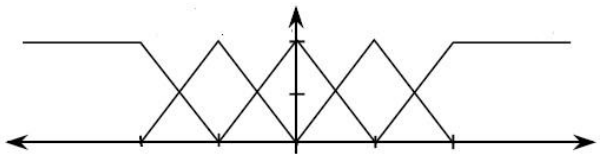

Fig 2: triangular membership function

The "center-average" method is used to obtain the representative value of the fuzzy set which is:

$$y = \frac{\sum_{i=1}^{R} b_i \mu_i}{\sum_{i=1}^{R} \mu_i} \quad \text{…… (2)[14]}$$

Where b_i is the output function and μi is defined as the Height of a fuzzy set or membership function:

$\mu_i(u_1,u_2,..,u_r) = \mu A^j_1(u_1) * \mu A^k_2(u_2) *..* \mu A^l_r(u_r)$……. (3)[14]

Where $\check{A}^j_1$ to $\check{A}^l_r$ are the fuzzy set, u_1 to u_r are the universe of discourse, j to l are the linguistic value of the linguistic variables u_1 to u_r respectively, 1 to r represent the linguistic variables and * represent AND operation.

The characteristics of TCP operations were examined in wired networks with various error rates and observed two important phenomenons that can be used to detect false congestion alarm with good precision. The input's variables are chosen from the parameters of the network and they are measured with respect to error rates from some experiences that are performed separately on random networks and topologies.

5.1 The inputs of fuzzy system

The proposed system structure is shown in Figure 3.

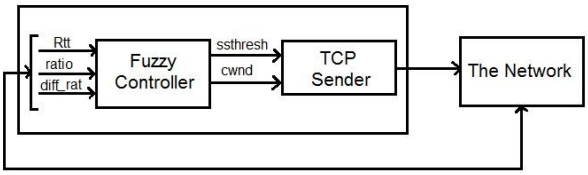

Fig 3: the proposed system structure

The inputs of the fuzzy controller are:

5.1.1. 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.

5.1.2 The ratio of the number of timeouts to the number of dupacks (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. This ratio is concluded according to the experiences first with respect to error rate and then with respect to UDP traffic with error rate =0 as in Table 1 and Table 2 respectively.

Table 1 Ratio with respect to different error rates

Error Rate (%)	Timeout Count (x)	3Dupack Count (y)	Ratio (x/y)
1	1	14	0.0714
5	1	13	0.0769
10	1	12	0.0833

Table 2 Ratio with respect to different UDP traffics

UDP traffic (Mbps)	Timeout Count (x)	3Dupack Count (y)	Ratio (x/y)
0.75	3600	1215	2.963
1	3643	1214	3
2	3643	1214	3

5.1.3 The ratio of the time difference between two consecutive timeouts to the current estimate of retransmission timer's timeout interval (diff.ratio)

This is the last input of the Fuzzy controller. It is measured in the same experiences for the above two inputs.

If the ratio is small (in between 0.01 to 0.1), the observation shows that this event has been caused by a bit error event, not by the congestion. If the ratio is high (e.g. greater than 0.25) then the event is more likely due to congestion.

5.2 Rules of Fuzzy System

The proposal algorithm includes 9 rules that are aggregated in a disjunctive manner, as in the table below.

Table 3 Rules of the Fuzzy System

Delay	Ratio	Diff_Ratio
Small	Small	Small
Small	Medium	Medium
Small	High	High
Medium	Small	Small
Medium	Medium	Medium
Medium	High	High
High	Small	Small
High	Medium	Medium
High	High	High

5.3 Outputs of the Fuzzy System

As mentioned before, Takagi-Sugeno Fuzzy System is used which its output represented by a function Instead of a linguistic variable. In this paper, there are two outputs ssthresh (slow start threshold) and cwnd (congestion window) which specify the new phase of TCP to trigger after the packet loss event.

5.4 Algorithm Steps

The proposed algorithm is shown in Figure 4. TCP with Fuzzy algorithm begins working just when a packet loss occurs and as follow:

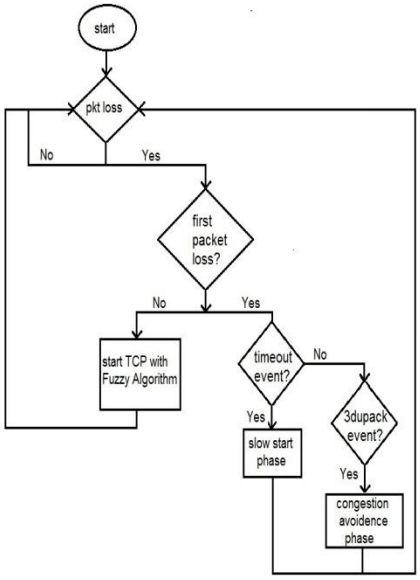

Fig 4: TCP Fuzzy System Algorithm

1. When a packet loss occurs, the algorithm start with checking whether this loss is the first loss or not, if it is the first loss then Start checking the cause of this loss(due to timeout or 3dupack)

   a. If it a timeout event, then start slow start phase which performed by set ssthresh to half (or 2 segments size whichever is the minimum) of the current congestion window and cwnd to one maximum segment size (MSS).

   b. If it a 3dupack event, then start congestion avoidance phase which performed by set ssthresh to one-half of the current window size and cwnd to ssthresh.

2. If it is not the first packet loss, then the TCP with Fuzzy algorithm starts. This begins with collecting the values required as inputs for the Fuzzy System (delay, the ratio of the number of timeout to the number of 3dupack and the ratio of the difference between the last 2 consecutive times to the current retransmission timeout) then checking the rules of the Fuzzy system to conclude which rule is on.

3. As soon as the rule that is on is found, the values of ssthresh and cwnd are set according to this rule. Based on the rules mentioned in Table 3, the equation of the two outputs is one of the following:

   a. TCP starts slow start phase which is performed by setting

Ssthresh = max {(BW[k] * RTT_min)/seg_size, 2*mss)}..(4)

Cwnd = 1* mss…….(5)

Where seg_size is the segment size and BW[k] is the estimated bandwidth and can be measured by:

$BW[k] = \beta \bullet BW[k-1] + (1-\beta/2)(sample\_BW[k] + sample\_BW[k-1])$ ……..(6)

Where β=19/21, BW [k − 1] is the previous estimated bandwidth, sample_BW[k] and sample_BW[k − 1]) are the last two bandwidth samples used by a connection [7].

b. TCP starts congestion avoidance phase which is performed by setting

Ssthresh = max {(BW * RTT_min)/seg_size, 2*mss)}….(4)

Cwnd = ssthresh ….(7)

c. TCP remains on same state such that ssthresh and cwnd keep the same values.

6. PERFORMANCE EVALUATION OF TCP FWESTWOOD

OMNET++ simulation IDE [18], [19], [20] has been used to evaluate TCP algorithm on the following network (Figure 5). The scenario has been built to compare TCP proposal with various kinds of TCP as TCP Reno, TCP Tahoe and TCP Westwood.

As shown in figure 5, the nodes starting with "w" connected with router1 through an individual link has delay = 0.02s and data rate = 1Mbps. Router2 is connected with the nodes starting with "n" by a link that has delay = 0.02s and data rate = 1.5Mbps. Finally, the link between router1 and router2 has data rate=10Mbps.

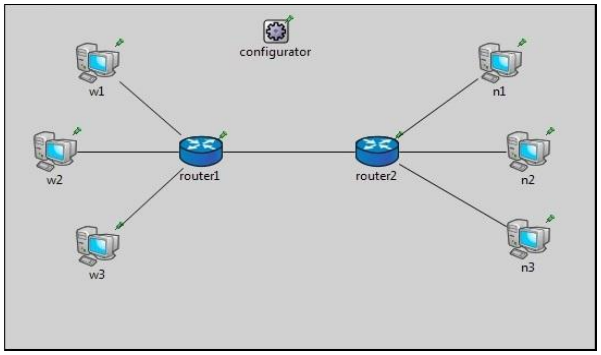

Fig 5: Simulation setup of wired network

The following traffics are generated:

- A TCP connection between *w*1 and *n*2
- A TCP connection between *w*2 and *n*1
- A TCP connection between *w*3 and *n*3
- A UDP connection between *w*1 and *n*2
- A UDP connection between w3 and *n*1

UDP connections have been used to generate constant bit rate traffics and create congestion in the network. In order to evaluate the performance of different TCP implementations the number of unique segments transmitted by the sender is used as a comparison parameter. If these values of the TCP are larger in one TCP implementation than that are in another TCP implementation, the former denotes the superiority over the later.

The simulation is run for TCP Tahoe, TCP Reno, TCP Westwood and TCP FWestwood. All the runs have been continued for 250 seconds. When a single TCP connection is concerned; TCP connection between nodes is used to analyze how TCP FWestwood behaves when it does not have to compete with other concurrent TCP connections.

6.1 Simulation Results and Analysis

Figure 6 shows simulation results obtained after running a single TCP connection without any UDP traffic for different TCP variants and TCP FWestwood. From the figure, it is clearly evident that FWestwood outperforms all other TCP variants with increasing error rate, even TCP Westwood which relies on consistent supply of acknowledgement segments from the receiver to estimate the available bandwidth of the network. TCPWestwood's performance depends highly on the precision of the above estimation. In the presence of bit errors, the nodes will fail to successfully transmit acknowledgement segments to their respective peers. So the presence of random bit errors refrain TCP Westwood from having an accurate estimate of available network bandwidth.

The performance improvement of FWestwood can be attributed to its less conservative reaction during segment losses due to random bit errors. Whenever FWestwood detects a possible non-congestion event it does not reduce its transmission rate too much rather continues to transmit at a good rate i.e. delivers more segments. On the other hand, other TCP variants (Tahoe, Reno and Westwood) drastically reduce the congestion window whenever a segment loss is detected, i.e. they fail to achieve a good throughput. In case of Reno, fast retransmission and fast recovery are capable of ensuring a good throughput when one segment is dropped from the window.

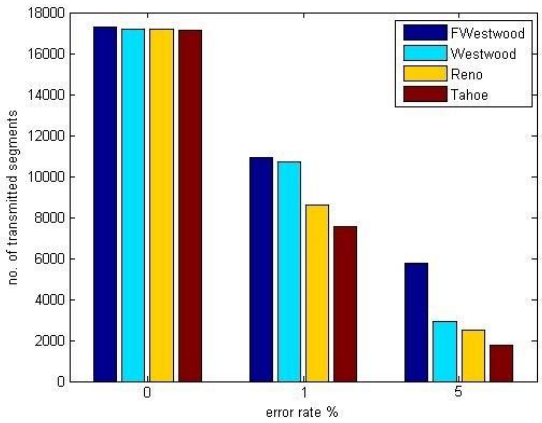

Fig 6: Performance comparison of single TCP connection without UDP traffic

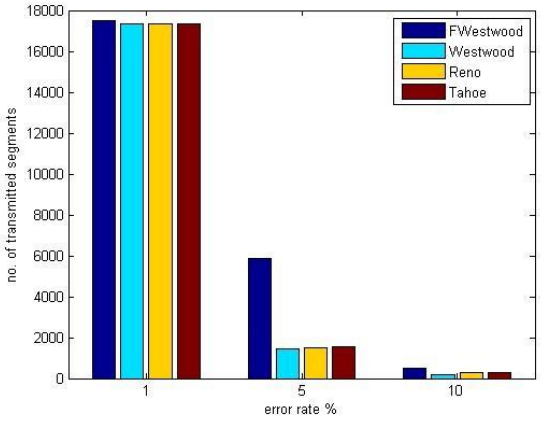

Fig 7: Performance comparison of single TCP connection with UDP traffic

However, if segment drops are sporadic in nature, consecutive reception of 3-dupacks will continue the halving of the congestion window even though the segments are dropped due to bit error. 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, FWestwood does not behave aggressively and hence do not worsen the congestion in the network. This behavior is very much significant where two concurrent TCP connections are used.

Figure 7 shows the performance of single TCP connection in the presence of UDP traffic for different TCP variants. Even in the presence of UDP traffic, FWestwood performs better than both TCP Reno and TCP Westwood.

Table 4 shows the simulation results obtained from two concurrent TCP connections. Here, the average number of unique segments sent by the two connections has been recorded. UDP traffics were kept present during these simulation runs.

Table 4 Performance comparison by the number of transmitted segments with error rate

Error Rate %	Reno	Westwood	FWestwood
0	17468	17469	17480
1	10004	7985	10179
5	1329	1322	5058

From the above data, it is clearly evident that FWestwood successfully injects more unique segments into the network compared to TCP Reno and TCP Westwood. These observations confirm that TCP FWestwood does not suffer with concurrent TCP connections. If TCP FWestwood were too much aggressive then it would have adversely affected the other TCP connections, who are sharing the same bottleneck link. In such case, a large number of segments will be dropped at the congested node during network overload time. Both the moderate and the aggressive connections will experience more timeout events and hence will throttle their rate of transmission. Moreover, TCP FWestwood does not reduce cwnd and/or ssthresh too much until it is sure about the fact that the timeout or 3-dupack event has been occurred due to real congestion related event. All the simulation results presented above have also shown that the gain of the TCP FWestwood increases with the increase in the error rate in the network, which is a desired property of a TCP algorithm that is designed to overcome the bit error problem.

7. CONCLUSIONS

A new congestion control algorithm is proposed that can be incorporated with any existing TCP variant and is capable of performing well in erroneous wired networks. This feature makes it suitable for deploying in real life scenario and does not impose any burden on the internal network. With the help of some newly introduced variables and decision blocks, it is able to determine whether segments are getting dropped in congested routers or are being damaged due to random bit errors. In case of real congestion, it simply behaves as original TCP Reno algorithm. However, after detecting a probable non-congestion event, unlike TCP Reno, it does not throttle its transmission rate too much. It continues to transmit at a good pace so that the network capacity does not remain unutilized at the presence of random bit errors. The proposed algorithm was compared with other major TCP variants using omnet++ and found that it performs better than all of them. Changes in the estimated round-trip time (RTT) of a TCP connection which enters as one of the parameters gives a good overview on the current network load. Therefore, the record keeping of previous RTT values and compare it with RTT_MIN and RTT_MAX and decisions based on those records give a great benefit toward improve TCP's performance in erroneous wired networks.

8. REFERENCES

[1] M. Allman, V. Paxson, W. Stevens, "TCP Congestion Control, RFC2581", April 1999.

[2] V. Jacobson, "Congestion avoidance and control", In ACM SIGCOMM, pages 314–329, Stanford, CA, August, 1988.

[3] J. Postel. Transmission control protocol, RFC 793, September, 1981.

[4] TCP Congestion avoidance algorithm, http://en.wikipedia.org/wiki/TCP-congestion-avoidance- algorithm, Accessed in July, 2012.

[5] J. C. Hoe, "Improving the start-up behavior of a congestion control scheme for tcp", In ACM SIGCOMM, pages 270–280, Stanford, CA, USA, August, 1996.

[6] S. Floyd, T. Henderson, A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 3782.

[7] C. CASETTI, M. GERLA, S. MASCOLO, "TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks", Wireless Networks 8, 467–479, 2002.

[8] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving tcp/ip performance over wireless networks', In 1st ACM International Conference on Mobile Computing and Networking (Mobicom), pages 2–11, November, 1995.

[9] S. Cen, P. C. Cosman, and G. M. Voelker, "End-to-end differentiation of congestion and wireless losses", IEEE/ACM Transactions on Networking, 11(5):703{717, October 2003.

[10] S.Biaz and N. H. Vaidya, "Discriminating congestion losses from wireless losses using inter-arrival times at the receiver", In IEEE Symposium ASSET'99, Richardson, TX, USA, March 1999.

[11] Y. Tobe, Y Tamura, A Molano, S. Ghosh, and H. Tokuda, "Achieving moderate fairness for UDP flows by path-status classication", In 25th LCN'00), Tampa, FL, USA, November 2000.

[12] A.Bakre and B. R. Badrinath. "I-tcp: Indirect tcp for mobile hosts". In 15th International Conference on Distributed Computing Systems, pages 136–143, May, 1995.

[13] P. Geurts, I. El Khayat, G. Leduc, "A Machine Learning Approach to Improve Congestion Control over Wireless Computer Networks", Institute Montefiore - B28 - Sart Tilman Li`ege 4000 – Belgium, 2010.

[14] K. M. Passino, S. Yurkovich, "Fuzzy Control", 1998 Addison Wesley Longman, Inc., 2725 Sand Hill Road, Menlo Park, California 94025, Columbus, Ohio, July 1997.

[15] N. K. Kasabov, "Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering, Chapter 3", 1996 Massachusetts Institute of Technology, JUNE 1995.

[16] M.Wozniak, "Using Fuzzy Inference to Improve TCP Congestion Control over Wireless Networks", College of Arts and Science at Stetson University Deland, Florida 2010.

[17] J.Liu, O.W.Yang, "Using Fuzzy Logic Control to Provide Intelligent Traffic Management Service for High-Speed Networks", IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, VOL.10, NO.2, JUNE 2013.

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

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

[20] OMNeT++ Community available at www.omnetpp.org.