J. Sci. technol. (Dinajpur) Vol. 5: 1-9 (2007)
ISSN 1994-0386

ANALYSIS BETWEEN VARIOUS TCP VERSIONS USING NS2 AND IMPROVING THE PERFORMANCE OVER CELLULAR MOBILE SYSTEM BY FEC USING JAVA

M. A. A. Mamun[1], M. S. Islam[2], M. M. Islam[3] and A. M. Nitu[4]

ABSTRACT

Transmission control protocol (TCP) provides reliable communication. Traditional TCP implementations are tuned to work well over wired networks. A packet loss is occurred in a wired network mainly due to network congestion. On the other hand in a wireless link packet losses are caused mainly due to bit errors resulted from noise, interference, and various kind of fading. TCP has no idea whether a packet loss is caused by congestion or bit error. In this paper, we have investigated TCP performance over cellular mobile system environments. TCP performances in these environments are analyzed for congestion window vs. time, throughput vs. time & throughput vs. delay in ns2 environment. We have also investigated the effects of combining forward error correction (FEC) with TCP in simulated environment using Java. FEC reduces the number of retransmissions and shows different congestion control behavior which can effectively run the network at a higher load.

Keywords: *TCP, FEC, Cellular Mobile System, Throughput, Congestion Window, Delay.*

INTRODUCTION

Due to rapid advances in the area of wireless communications and the popularity of the Internet, provision of packet data services for applications like-mail, web browsing and mobile computing over wireless channels is gain in importance. Transport Control Protocol (TCP) is a reliable end-to-end, transport protocol that is widely used to support applications like telnet, ftp, and http.

TCP was designed primarily for wire line networks where the channel error rates are very low and congestion is the primary cause of packet loss. Since its original deployment, several modifications to TCP, including Reno, NewReno, Fack, and Vegas, have been proposed and their performance analyzed in wire-line networks.

Fall and Floyd evaluated the performance of TCP Reno, New Reno and SACK by simulations that focused on comparing the differences of packet loss recovery mechanisms (Fall and Floyd, 1996). In addition, in their simulation, packet loss occurrence was designed specifically to highlight the improved performance of SACK. Lakshman and Madhow, investigated the performance of TCP Tahoe and Reno in the presence of large bandwidth-delay product and random losses (Lakshman and Madhow, 1997). They assumed finite buffer size. Nayma and Sumyea analysis of TCP Tahoe, Reno and New Reno and Sack over cellular mobile system but they do not consider Vegas and Fack, they consider two parameters for performance analysis and do not vary TCP receivers (Nayama and Sumyea, 2005).

[1]Lecturer, Dept. of Computer Engineering, [2]Lecturer, Dept. of Electrical and Electronic Engineering, [3]Lecturer, Dept. of Mathematics and Physics and [4]Lecturer, Dept. of Computer Science and Information Technology, HSTU, Dinajpur.

VERSIONS OF TCP AND FORWARD ERROR CORRECTION

This TCP implementation has become known as TCP Tahoe. TCP Reno adds one new mechanism called Fast Recovery to TCP Tahoe. TCP NewReno is the newest standardized TCP variant. It contains changes to both the congestion control and the retransmission mechanism of TCP Reno. The use of Sacks permits the receiver to specify several additional data packets that have been received out-of-order within one dupack, instead of only the last in order packet received. TCP Vegas proposes its own unique retransmission and congestion control strategies.

TCP Fack is Reno TCP with forward acknowledgment (Lang, 2002).

For forward error correction redundant information is proactively sent to the receiver in order to allow it to correct errors. The redundant information is generated by means of channel coding. Well-known examples of channel coding are Hamming codes, Reed-Solomon codes and Bose – Chaudhuri – Hochquenghem (BCH) codes.

Hamming code provides a practical solution. The Hamming code can be applied to data units of any length and uses the relationship between data and redundancy bits that can be added to the end of the data unit or interspersed with the original data bits.
For the no of redundant bits $2^r >= m+r+1$ inequality must be satisfied. Here m is the no of data bit and r is the no of redundant bit. If m=7 to satisfy the inequality r=4.

SIMULATION MODEL

The simulation model (network topology) used in this simulation is shown in figure 1. As shown in the figure there are total five nodes. One sender, one base station and three receivers are used. Here the link between the base station and sender is wired and the link between base station and receiver is wireless. So for one sender six versions of TCP sender (supported by NS2) is used and for three receiver four versions of TCP receiver (supported by NS2) is used. Table 1 show the simulation parameter used in simulation.

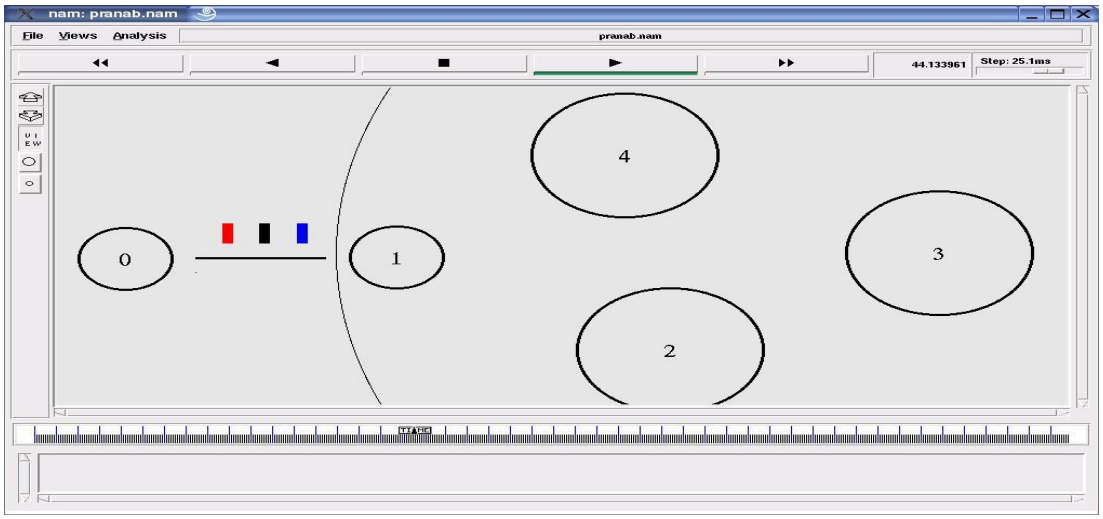


Fig. 1: Simulation Topology

Table 1: Simulation parameter

Between Base Station & TCP Receiver-	Between Base Station & TCP Sender-
Media: Wireless	Media: Wired
Standard: Orinoco 802.11b 11Mbps PC card	Standard: Ethernet 802.3
Frequency: 2.472 GHz	Delay: 1ms to 100ms

RESULTS AND DISCUSSION

For comparison between various TCP versions total 120 second simulation is performed. At first the throughput for various sender and receiver is represented in graph then the average throughput is also calculated. And found that Agent/TCP/Fack give the highest throughput. Now the congestion window size is compared and the average window size is also summarized in table. And found that Agent/TCP/Fack give the highest window size. And to compare between various versions throughput vs. delay is also considered.

Then to improve the TCP performance forward error correction is also used for that hamming code algorithm is used for loss probability of 10% the improvement of TCP is also shown in figure.

TCP Throughput vs. Time:

TCP Throughput vs. Time for Sender Agent/TCP & Agent/TCP/Reno:

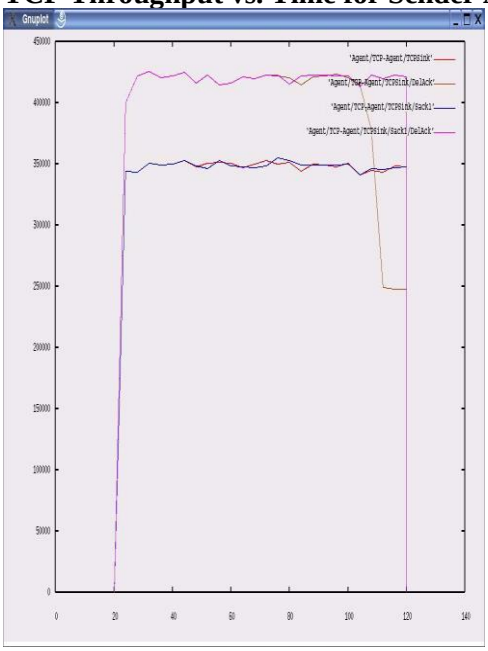


Fig 2: For Sender Agent/TCP

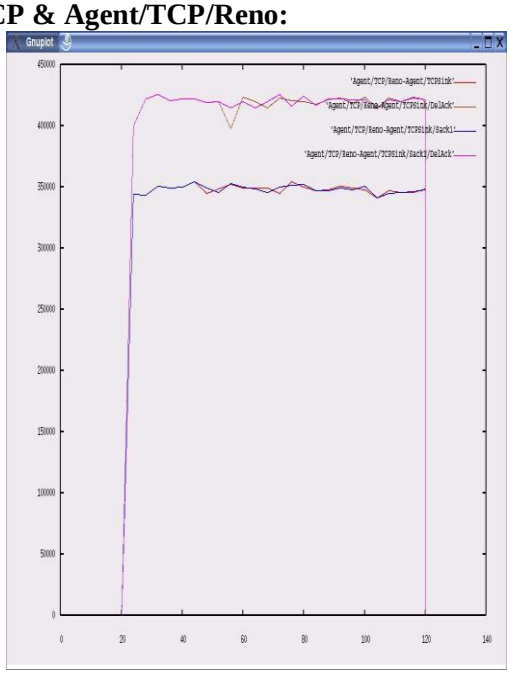


Fig 3: For Sender Agent/TCP/Reno

TCP throughput vs. Time for Sender Agent/TCP/Newreno & Agent/TCP/Sack1:

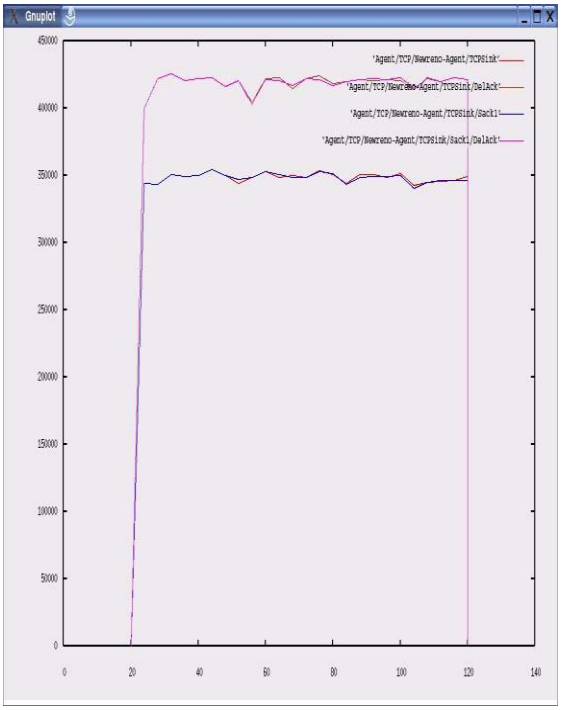


Fig 4: For Sender Agent/TCP/Newreno

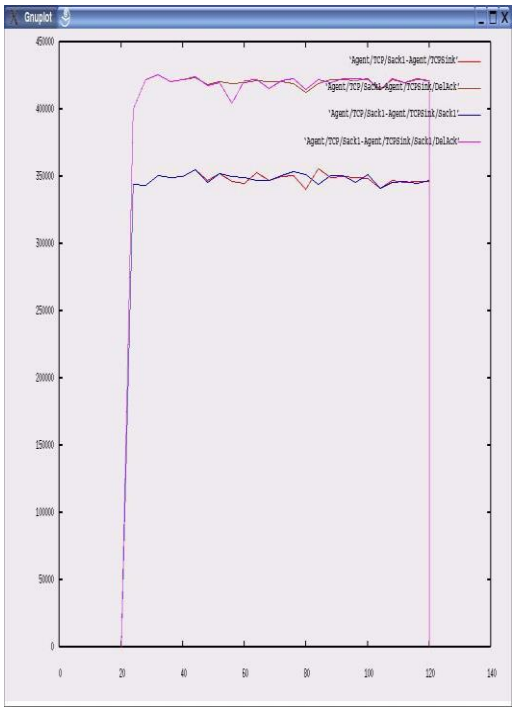


Fig 5: For Sender Agent/TCP/Sack1

TCP throughput vs. Time for Sender Agent/TCP/Vegas & Agent/TCP/Fack:

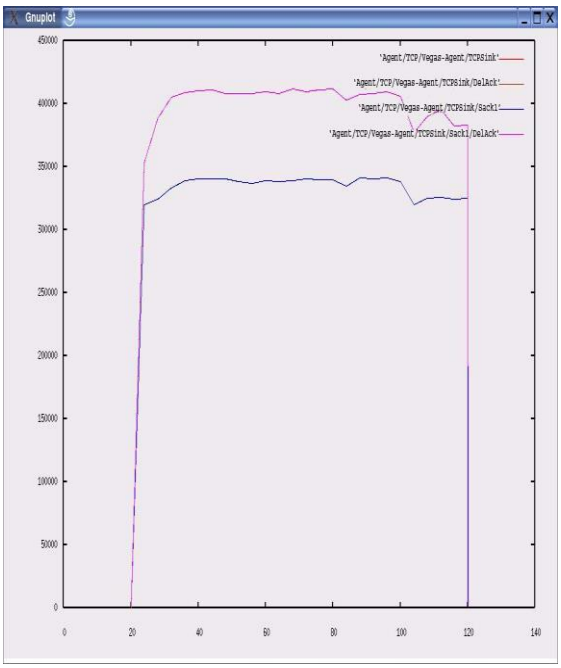


Fig 6: For Sender Agent/TCP/Vegas

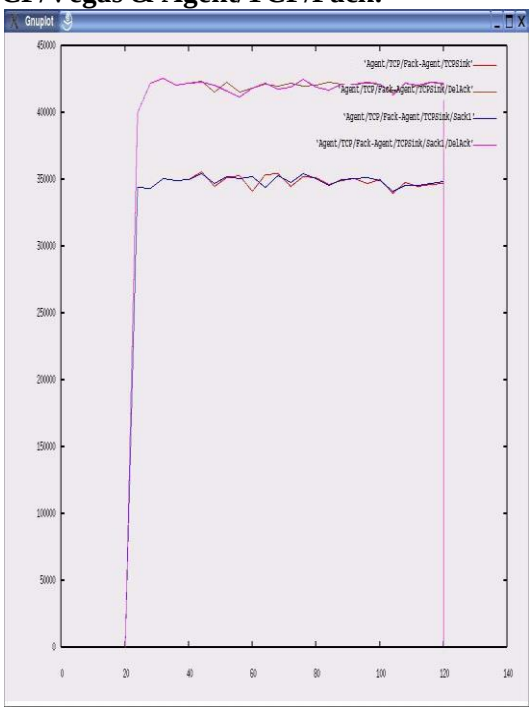


Fig 7: For Sender Agent/TCP/Fack

Congestation Window vs. time:
Congestation Window vs. time for Sender Agent/TCP & Agent/TCP/Reno:

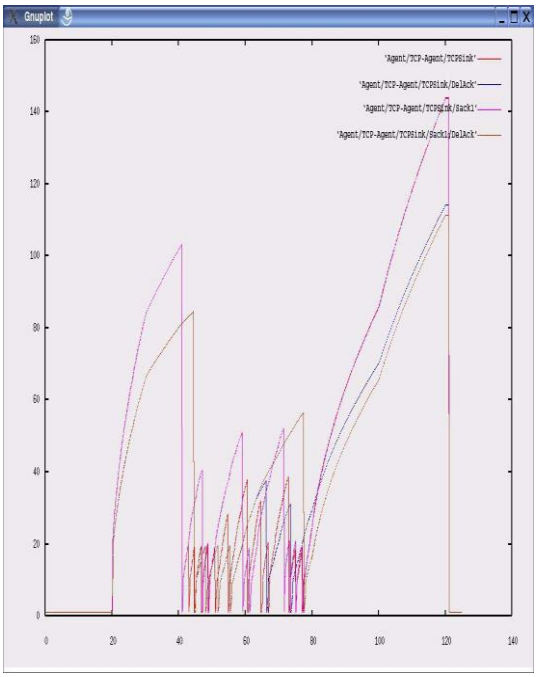


Fig 8: For Sender Agent/TCP

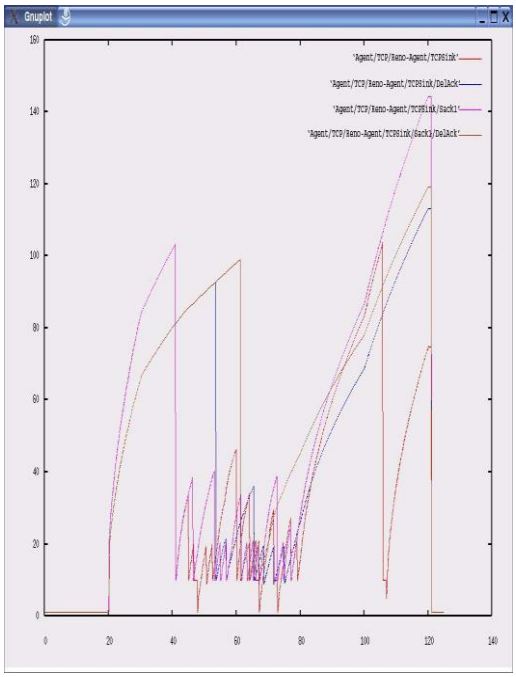
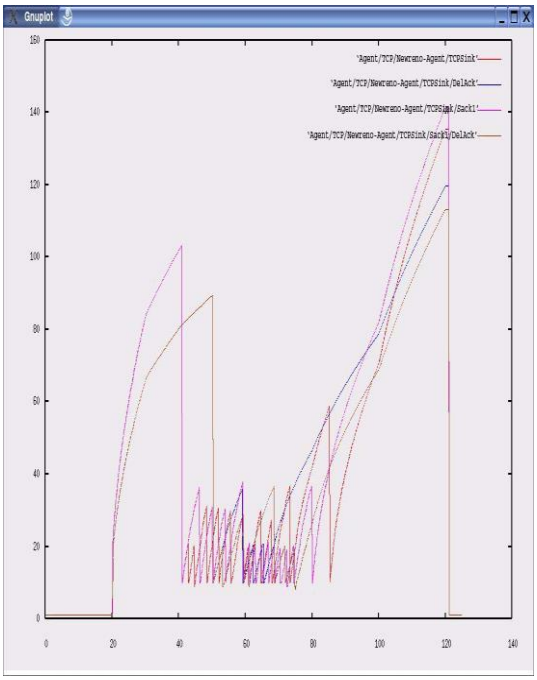


Fig 9: For Sender Agent/TCP/Reno

Congestation Window vs. time for Sender Agent/TCP/Newreno & Agent/TCP/Sack1:



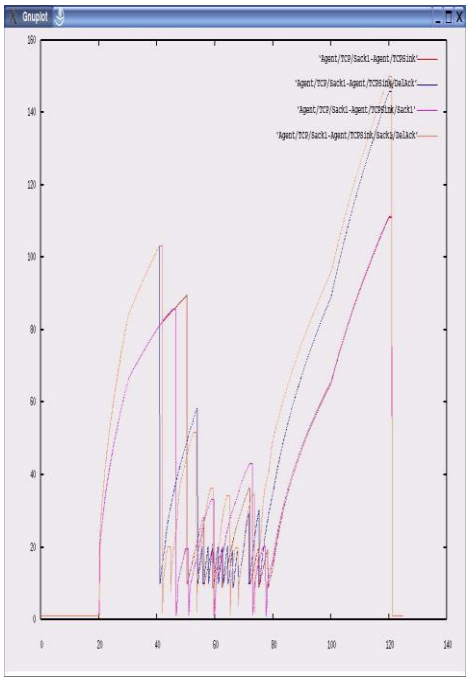


Fig 10: For Sender Agent/TCP/Newreno Fig 11: For Sender Agent/TCP/Sack1

Congestation Window vs. time for Sender Agent /TCP/Vegas & Agent/TCP/Fack:

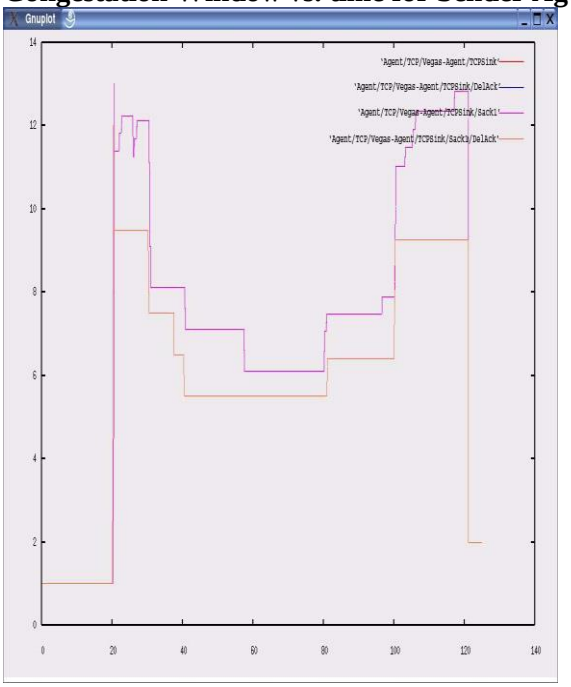


Fig 12: For Sender Agent/TCP/Vegas

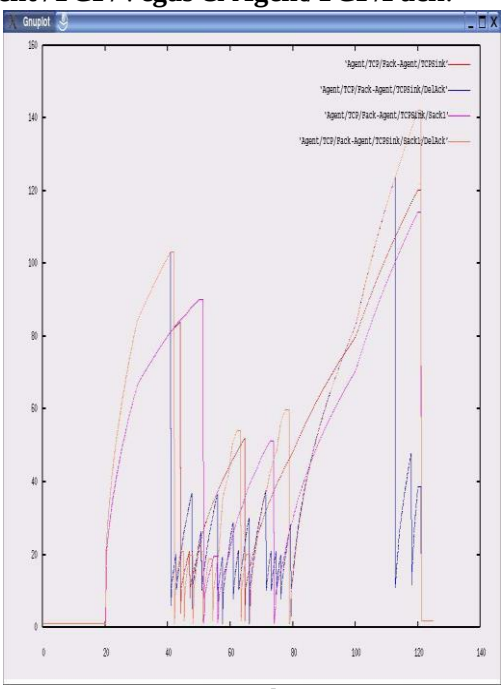


Fig 13: For Sender Agent/TCP/Fack

TCP Throughput vs. Delay:

TCP throughput vs. Delay for Sender Agent/TCP & Agent/TCP/Reno:

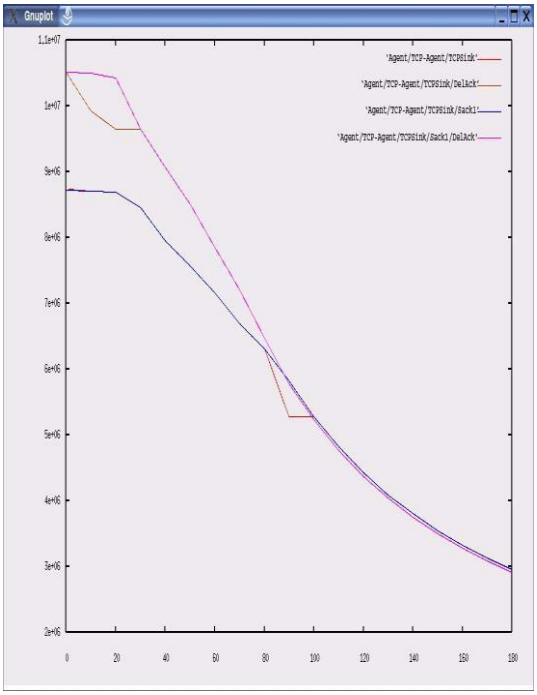


Fig 14: For Sender Agent/TCP

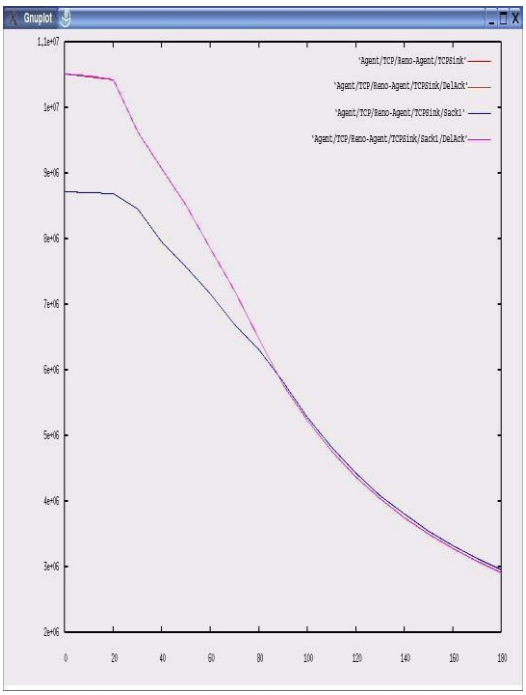


Fig 15: For Sender Agent/TCP/Reno

Agent/TCP/Reno

TCP throughput vs. Delay for Sender Agent/TCP/Newreno & Agent/TCP/Sack1:

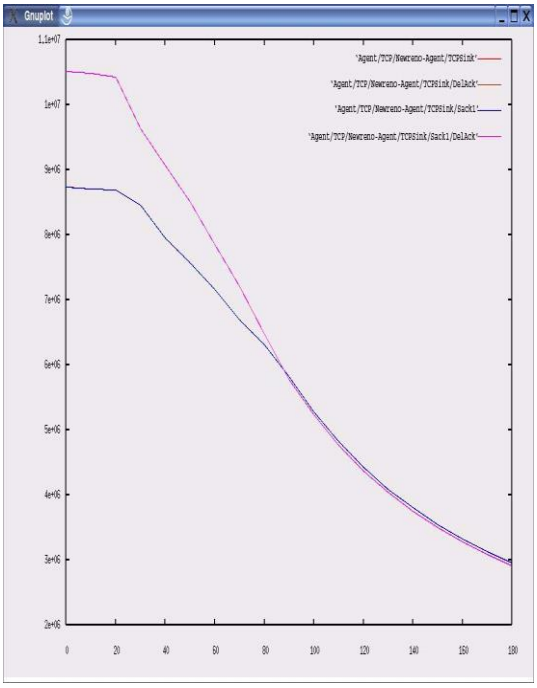


Fig 16: For Sender Agent/TCP/Newreno

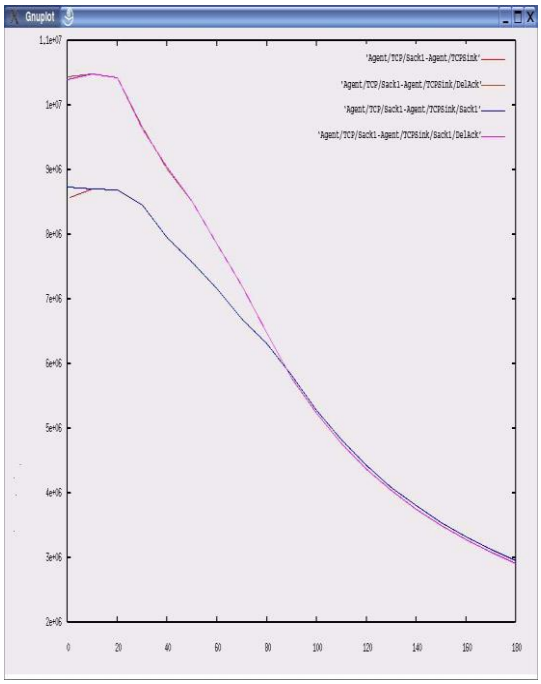


Fig 17: For Sender Agent/TCP/Sack1

TCP throughput vs. Delay for Sender Agent/TCP/Vegas & Agent/TCP/Fack:

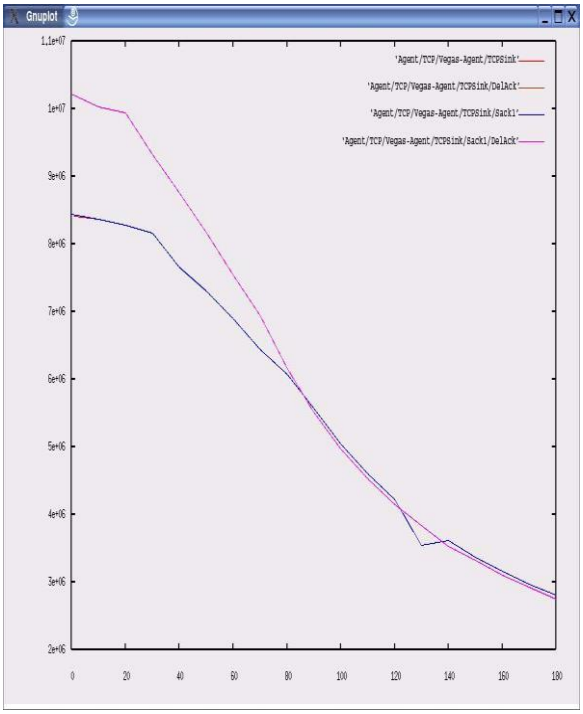


Fig 18: For Sender Agent/TCP/Vegas

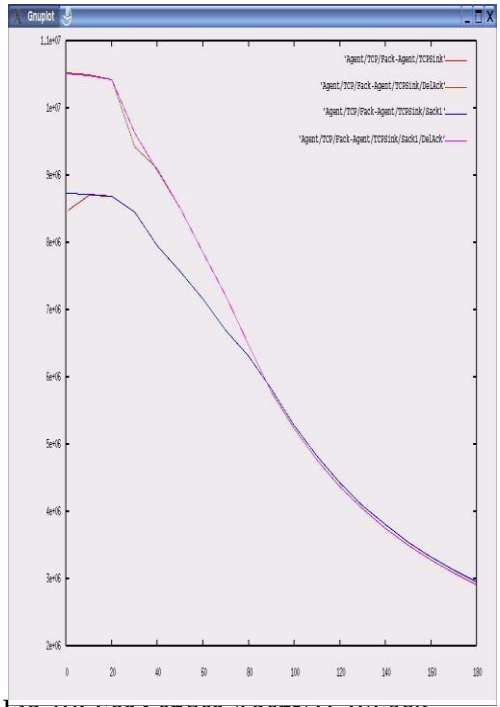


Fig 19: For Sender Agent/TCP/Fack

Comparison of original TCP and TCP with FEC:

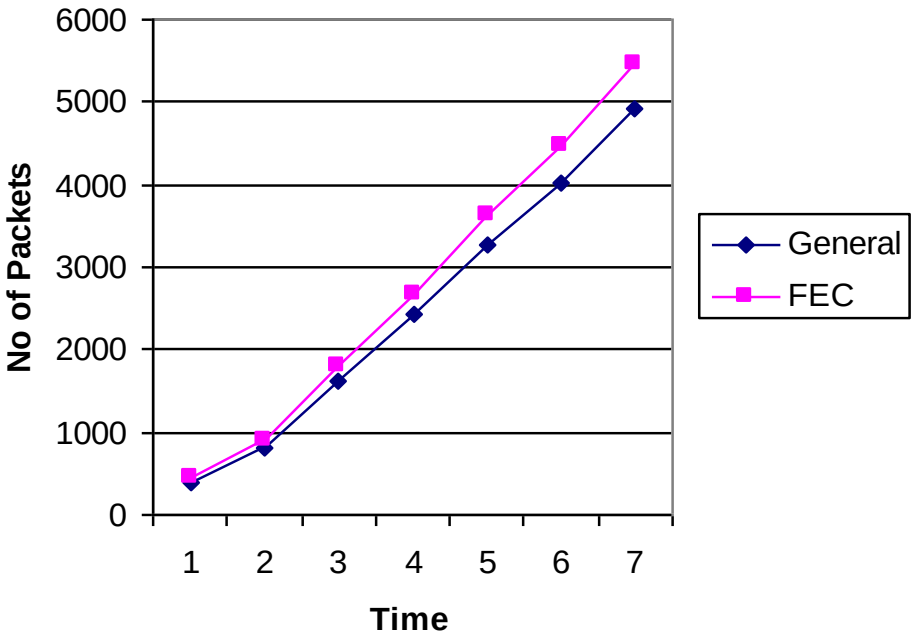


Fig 20: No of packet vs. time for original TCP and TCP with FEC

From the above figures for all the six senders and four receivers are considered for performance analysis. For implementation using NS2 one pair of TCP sender and TCP receiver should considered. For those figures it is found that sender Agent/TCP/Fack produce highest window size for receiver Agent/TCPSink. The others pairs can also found from those tables. For average throughput sender Agent/TCP and receiver Agent/TCPSink/Sack1/DelAck produce the highest throughput than the others. The pair between sender Agent/TCP/Fack and receiver Agent/TCPSink/DelAck produces the second highest. So form the all the figures receiver Agent/TCPSink/DelAck produce the maximum throughput.

For forward error correction here hamming code is implemented. Hamming code can only correct single error so for a high bit error rate the coding is not good enough. As for example this research deals with 10% error probability. i.e from hundred packets ten packets will be received with errors. But from figure it is clear that TCP with FEC produce better results than fresh TCP.

CONCLUSION

In this paper we investigate TCP senders and TCP receiver and forward error correction with TCP is also investigated. From our analysis we find that TCP Fack give better throughput than other over cellular mobile system. And we also show that FEC with TCP give good throughput. Further investigation can be carried out by implementing FEC with TCP using NS2. And the

performance analysis can also be carried out by including other workloads. For example Hyper Text Transfer Protocol (HTTP) transaction. And performance analysis can also be carried out by considering other errors.

REFERENCES

Fall K and Floyd S. 1996. Simulation based comparison of Tahoe, Reno, and Sack TCP. Computer Communication Review. 26(3): 5-21.

Lakshman V and Madhow U. 1997. The performance of TCP\IP for Networks with high Bandwidth Delay Products and Random loss. IEEE/ACM Transaction On Networking. 5(3).

Islam N and Helal S. 2005. Performance Analysis of TCP Tahoe, Reno, New Reno and SACK over Cellular Mobile System. 8th ICCIT, Department of Information and Communication Technology, University of Rajshahi. pp. 786-790.

Network Simulator (NS), The network simulator-ns-2.27. NS Simulator for beginners. December 4, 2003. Lecture notes, 2003-2004, Univ. de Los Andes, Merida, Venezuela and ESSI, Sophia-Antipolis, France.

The network simulator (ns) Manual. December 13, 2003. The VINT Project, A Collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC.

Lang T. 2002. Evaluation of different TCP versions in non-wireline environments. The University of South Australia, Institute for Telecommunications Research.