Proceedings of the 5th WSEAS Int. Conf. on APPLIED INFORMATICS and COMMUNICATIONS, Malta, September 15-17, 2005 (pp54-60)

Simulation based Evaluation of TCP in Wireless LAN Environment

MOUNIR FRIKHA, FAKHER KRICHEN
Informatics NTSID
ENIS
Soukra Road Km 4.5 SFAX
TUNISIA

Abstract: TCP Algorithms were developed to avoid congestion in wired networks and to increase the throughput. The efficiency of TCP in the wired networks has been demonstrated because it was designed for it. Because of its well-known performance, it was used for the wireless network, but these algorithms loose their performance when the Binary Error Rate (BER) increases, such as in wireless LANs.
In wireless networks, the implicit assumption which TCP makes that losses indicate network congestion is no longer valid. Losses in wireless networks can result from bit errors, fading and handoffs.
In this paper we will compare the result of using classic TCP algorithms in wireless, we will demonstrate the relationship between the throughput and the BER and we will improve the throughput offered by changing the length of the initial congestion window.

Key-Words: TCP, Congestion, Wireless, Window, BER, Reno, New Reno, Sack, Vegas

1 Introduction

The emergence of new services and features offered by hosts, supporting multimedia applications and requiring reliable data transfer, is constraining to use the maximum bandwidth of links offered and minimize losses in the network caused by congestion. TCP is a solution of these constraints, it is widely used today, it performance is approved in wired networks. Wired links have stable transmission characteristics and low BER (between 10^{-6} and 10^{-9}) and the lost of packet is mostly caused by buffer overflows, indicating the congestion of a node. TCP uses this information to adapt the throughput of the link by reducing the congestion window of the sender.
The Transmission Control Protocol (TCP) is intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks [11].
The Protocol TCP was developed to offer a secure service of high performance data transfer between two hosts connected. To be able to offer this service, over a protocol layer less efficient, the following functionalities would be necessary:

- Basic data transfer: TCP is able to transfer a continuous data flow between two hosts, by packaging this amount of data into packets or datagrams.
- Reliability: TCP must recover from data that is damaged, lost, duplicated or delivered out of order.
- Flow Control: TCP provides a means for the receiver to govern the amount of data sent by the sender. This is achieved by returning a "window" with every ACK indicating a range of acceptable sequence numbers beyond the last segment successfully received. The window indicates an allowed number of octets that the sender may transmit before receiving further permission.
- Multiplexing: the TCP provides a set of addresses or ports within each host to allow for many processes within a single Host to use TCP communication facilities simultaneously.
- Connections: TCP maintain certain status information for each data stream. The combination of this information, including sockets, sequence numbers, and window sizes, is called connection.
- Precedence and Security: The users of TCP may indicate the security and precedence of their communication. Provision is made for default values to be used when these features are not needed.

Wireless links, which are being used more frequently because of the growth of mobile hosts supporting multimedia applications, represent high BER, in the order of 10^{-3} and sometimes as high as 10^{-1}. The TCP considers each loss as a result of congestion, and reduces the congestion window of the sender causing the data transmission to slow down.
In this paper we will represent the relation between the increase of the BER in wireless networks and the throughput of the network, and we will propose an amelioration of the data sending rate by an appropriate choice of the length of the initial window.

2 Wireless Networks:

Wireless Networks can be classified into three:

- Cellular Networks: Networks in which a mobile host is connected to the fixed network with the help of the Base Station. This is the most common for the wireless networks. Most of the solutions proposed to TCP over wireless networks use this model.
- Ad-Hoc Networks: Networks formed by mobile hosts which are connected to each other within a radio distance. This kind of a model is not well deployed and very few solutions have been proposed to this model.
- Satellite Networks: Networks in which satellite link is in between the sender and the receiver. These have very high BERs and high latency because the Satellites are at a great distance from the surface.

3 Simulation Model:

A typical wireless data application includes data flows between a fixed host in the wired network and a mobile host via a base station, represented in the following figure.

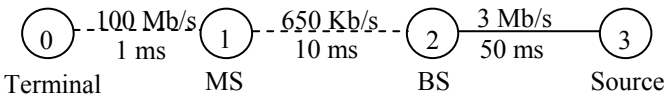

Fig.1 : Topology of the simulation model

The path from the fixed host (node n°3) to the base station (node n°2) is modelled as a link with a bandwidth of 3 Mb/s and propagation delay of 50 ms. We assume the bottleneck of the TCP lies on the wireless link between the BS and the mobile station (node n°1), modelled with a bandwidth of 650 Kb/s and a propagation delay of 10 ms. The wireless link between MS and the terminal (node n°0) is assumed with no error rate, height bandwidth (100Mb/s) and low propagation delay (1ms).

4 Comparison of TCP algorithms in wireless topologies:

2.1 TCP Reno:

TCP Reno uses slow start, congestion avoidance, fast retransmit and fast recovery, phases.
In Reno, the congestion window takes the value of min (awin, cwnd+ndup).
awin: the announced congestion window of the receiver.
ndup: is equal to 0, until the number of duplicated acknowledgement (dupACK) reach the threshold.
At the fast recovery phase the sender transmits only one packet, waits for the reception of a half window size dupACKs and then it sends a new packet for each additional dupACK received. In the case when it receives a new acknowledgement it goes out of the fast recovery phase and reinitializes ndup to 0.

2.2 TCP Sack:

The weakness of TCP Reno is important when more than one packet of the same window is lost because of the limited information that offer the cumulated ACK, the sender can only be informed about the lost of one and only one packet by RTT (Round Trip Time).
The Selective ACK mechanism (SACK) surmount these limits by using the option field in the TCP datagram, Sack algorithm let the receiver informing the sender about the reception of non contiguous data blocs. The receiver waits for the reception of data to full the blanks in the received bloc's sequence. When the missing segments are received, the receiver acknowledges all data by a standard ACK.
TCP Sack algorithm is an extension of TCP Reno; it uses the same procedures for the increase and the decrease of the congestion window. As in Reno, TCP Sack enter in the fast recovery phase after reception of an appropriate number of duplicated ACK, the sender then retransmit the

packet and reduce the congestion window size to its half.

2.3 TCP Vegas:

TCP Vegas algorithm changes the process by which the window size varies. Vegas uses three techniques to increase the flow and to reduce the losses:

- It bases the decision of retransmission of the lost packages by the time passed between the emission of a segment and the reception of it acknowledgement.
- It anticipates the congestion and adjusts the rate of transmission.
- It modifies the mechanism of slow start to prevent the loss of the packages during the research of the value of the bandwidth.

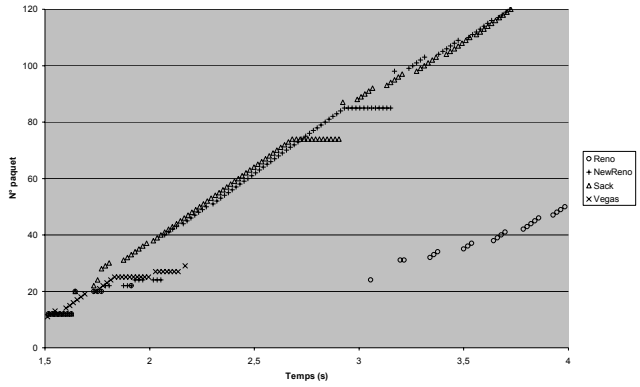

Fig.2 : Congestion Algorithms Comparison (Buffer=3 packets, TEB=10^{-3})

Fig.2 shows a comparison of TCP algorithms (Reno, New Reno, Sack and Vegas) in wireless topology with a Buffer of 3 packets and a BER = 10^{-3}.By using TCP Sack the number of acknowledge packets, is greater than in Reno, New Reno and Vegas.

5 Evaluation of the initial window length variation in Wireless Networks:

In this paragraph we will simulate TCP Sack with a BER equal to 10^{-3} and buffer of 3 packets, and we will vary the length of the initial window, which is set to 1 in the TCP Sack's original version. We obtain the following results:

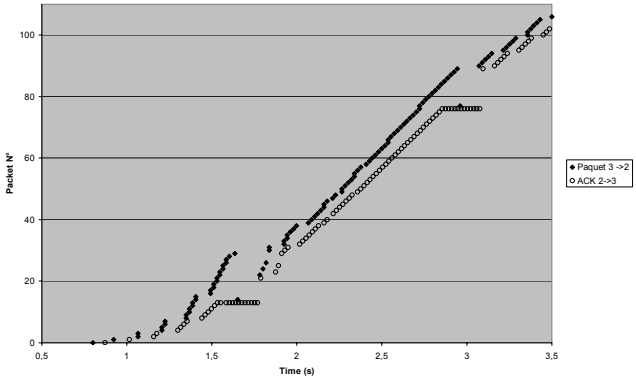

Fig.3 : TCP Sack Simulation Results (Buffer=3 packets, TEB=10^{-3}, wininit =1)

Fig.3 shows TCP Sack in wireless topology with a buffer of 3 packets, a BER of 10^{-3} and a windowinit of 1 packet.

In Fig.3, the Base Station starts sending messages by transmitting the packet number 0 and waits for it acknowledgment, after what it sends the packet number 1 because of the length of the initial congestion window set to one. The packets labelled 0 to 13 are sent without error, as sending TCP's congestion window increases exponentially according to the Slow-Start algorithm. The packet numbered 14 is lost in the bottleneck path. The sender (node 3) sends the next group of 16 packets until the labelled 29. Because of the packet lost, the node 0 sends an acknowledgement of packet 13, for each packet received from 15 to 29 showing their reception with no error. When the node 3 receives duplicated ACK of this packet containing the information of the lost of the packet 14, it resends it and continues transmission as defined in Fast Recovery algorithm and Congestion Avoidance when reaching the Slow-Start threshold.

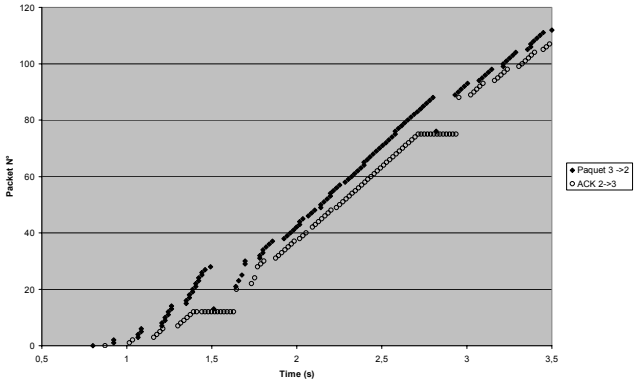

Fig.4 : TCP Sack Simulation Results (Buffer=3 packets, TEB=10^{-3}, wininit =2)

Fig.4 shows TCP Sack in wireless topology with a buffer of 3 packets, a BER of 10^{-3} and a windowinit of 2 packets.
In this situation, the node 3 sends, after the acknowledgement of packet 0, two packets numbered 1 and 2 because of the initial congestion window set to 2. As in preceding simulation the sender increases the congestion window exponentially regularly that the sent packets are acknowledged. In this situation the packet number 13 is lost (we reach the congestion before than in the preceding simulation). As in Fig.3 the lost packet belongs to the fourth congestion window. After the reception of duplicated ACK of the packet 12, the receiver sends acknowledgements of the packets 14 to 20 which have been received correctly. The sender concludes that the packet 13 is not received to destination and then retransmits it. After the reception of the acknowledgement of the packet 20, the sender continues sending the lost packets and continues the transmission as defined in the TCP Sack algorithm.

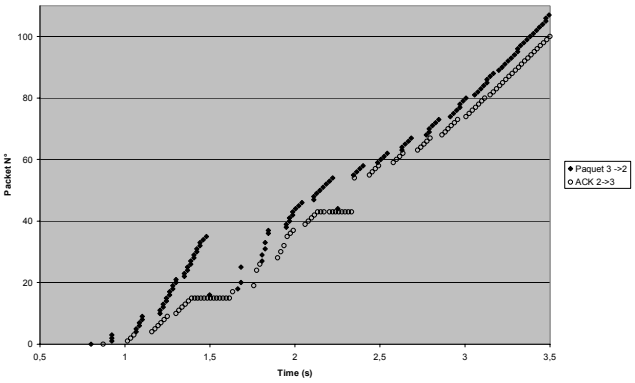

Fig.5 : TCP Sack Simulation Results
(Buffer=3 packets, TEB=10^{-3}, wininit =3)

Fig.5 shows TCP Sack simulation results obtained in case of an initial window set to 3. As in previous simulations, the sender (node 3) starts sending a number of packets equal to the size of the congestion window, and when receiving their ACKs, it starts sending the next congestion window packets by increasing exponentially its size. In this simulation the first packet lost is number 16. The receiver sends acknowledgements of the packet 15, for each received packet which belongs to the current congestion window until the packet 27. The packet lost is retransmitted when receiving the dupACKs, and then continues with transmitting the next messages.

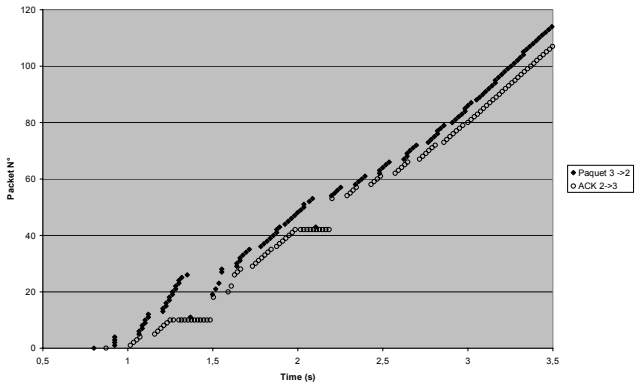

Fig.6 : TCP Sack Simulation Results
(Buffer=3 packets, TEB=10^{-3}, wininit =4)

In the simulation shown in Fig.6, the size of the initial congestion window is set to 4. In this case the first lost packet is numbered 11. This packet belongs to the third congestion window contrary to the preceding simulations.

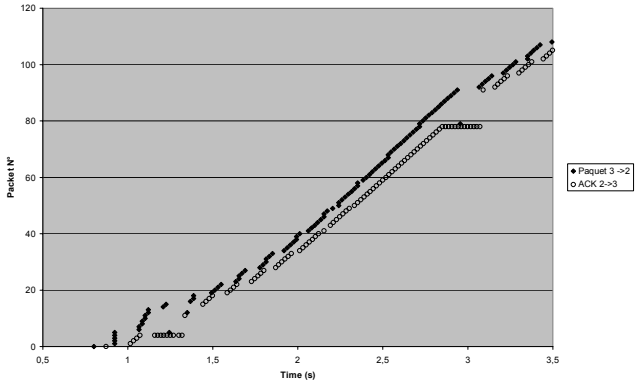

Fig.7 : TCP Sack Simulation Results
(Buffer=3 packets, TEB=10^{-3}, wininit =5)

The Fig.7 shows the packets sent by the node 3 to the node 2 and the acknowledgments received, when changing the size of the initial window and fixing it at 5. In this case the first lost packet is numbered 6. This is caused by the massif transmission despite the size of the buffer used, of 3 packets. We can conclude that the configuration used is mal adjusted of the wireless configuration already defined.

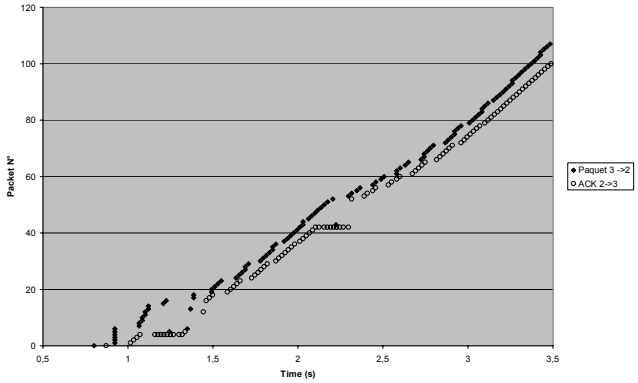

Fig.8 : TCP Sack Simulation Results (Buffer=3 packets, TEB=10^{-3}, wininit =6)

By changing the size of the initial window of the congestion algorithm, we obtain the result shown by Fig.8. In this case the first lost packet, as in the previous configuration, is labelled 6.

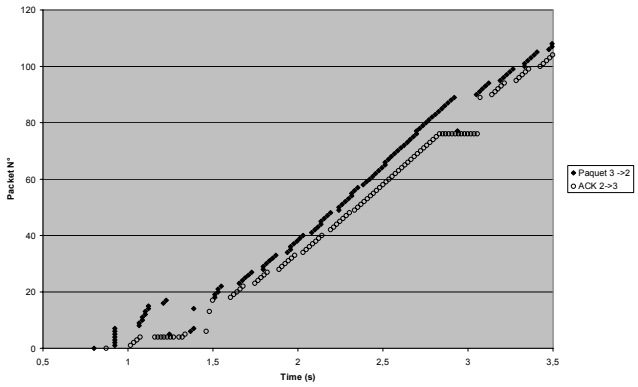

Fig.9 : TCP Sack Simulation Results (Buffer=3 packets, TEB=10^{-3}, wininit =7)

In Fig.9 we represent the packets sent by the node 3 and the acknowledgement received, when using a wireless LAN topology with a buffer of three packets, a BER of 10^{-3}, in the wireless link, and fixing wininit to 7.

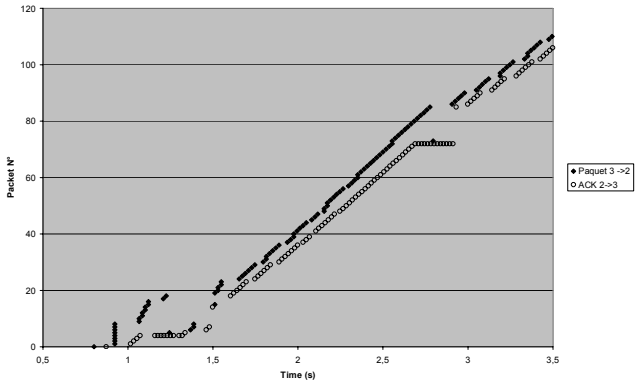

Fig.10 : TCP Sack Simulation Results (Buffer=3 packets, TEB=10^{-3}, wininit =8)

Fig.10 shows the TCP Sack simulation results obtained when fixing the size of the initial window's congestion algorithm to 8.

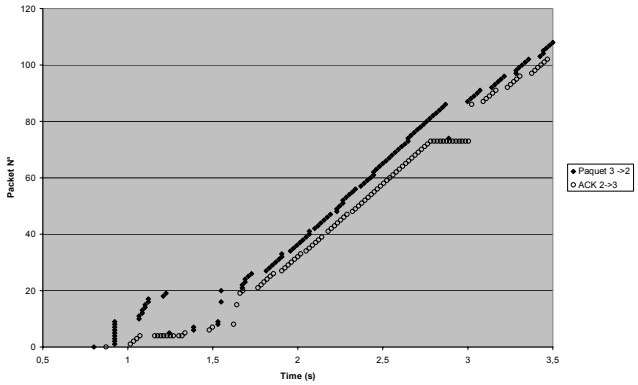

Fig.11 : TCP Sack Simulation Results (Buffer=3 packets, TEB=10^{-3}, wininit =9)

Previous figures labelled Figure 7 to Figure 11 represent the simulation of TCP Sack algorithm in wireless links with an initial congestion window set respectively to 5, 6, 7, 8 and 9. The first packets lost in theses simulations are all numbered 5. This packet belongs to the first congestion window. We can guess from this result that the efficient throughput is less than their obtained in the first half simulations.

By a graphical comparison we conclude that the best performances are obtained for a window init equal to 3 or 4 packets, for high values the performance is reduced because in this case the first lost packet, in the topology and the configuration used in these simulations, belongs to the first transmission window, so the algorithm pass quickly in congestion avoidance phase and this reduce the TCP performance.

Initial Congestion Window Size	Efficient Bandwidth (Kb/s) BER = 10^{-3}
1	499522,796
2	506383,035
3	523916,677
4	516087,86
5	505335,698
6	505666,587
7	499840,782
8	511328,302
9	504941,833

Table 1 : Network Throughput Comparison

6 Conclusion:

In this paper we studied the results of simulations of the TCP Sack congestion algorithm in WLAN connections, obtained while varying the size of the congestion initial window from 1 to 9.

After the study of these results we can deduce that a good choice of the size of the initial window (wininit) improves TCP protocol, so it is necessary to have a rather clear idea on the state of the connection, on its available bandwidth and its error rate. Generally in the majority cases, the probability that the first lost package belongs to the first congestion window is almost null. Thus it is judicious to choose the size of the initial congestion window of a value strictly higher than one.

References

[1] JACOBSON Van, Congestion Avoidance and Control, *Lawrence Berkeley Laboratory,* Novembre 1988.

[2] JOEL CANNAU, Evaluation de performances d'une amélioration de TCP Westwood, *Université Libre de Bruxelles,* Année Académique 2002/2003.

[3] Brain Levy, Le point de vue d'un opérateur sur l'évolution des télécommunications, *Revue des Télécommunications d'ALCATEL,* 2002.

[4] NASRI Salem, *Cours Analyse des Performances : Réseaux et QoS,* ENIS 2002/2003

[5] ADJIH Cédreic, Multimédia et accés à l'Internet haut débit : l'analyse de la filière du câble. *Projet Hipercom, INRIA Roconcourt.*

[6] C. Pham, *Quality of Service & Scheduling,* Univ. Lyon1.

[7] FALL Kevin and FLOYD Sally, *Simulation based Comparison of Tahoe, Reno and SACK TCP,* Lawrence Berkeley National Laboratory, Mars 1996.

[8] M. Mathis, J. Mahdavi, S. Floyd, A.Romanow, *RFC 2018 TCP Selective Acknowledgment Options*, October 1996

[9] S. Dawkins, G. Montenegro, M. Kojo, V. Magret, *RFC 3155 Implication des liens défectueux sur les performances de bout en bout*, Network Working Group, Août 2001

[10] Lawrence S. Brakmo, Sean W. O'Malley, Larry L. Peterson, *TCP Vegas : New Techniques for Congestion Detection and Avoidance,* Department of Computer science University of Arizona.

[11] Marina Del Rey, *RFC 793 Transmission Control Protocol,* Protocol Specification, Information Sciences Institute: University of Southern California, September 1981.

[12] Jean Pierre Nziga, TCP Performance of Mobile IP in Wireless Networks.

[13] Andrei Gutrov, *Modelling Wireless Link for Transport Protocols,* University of Helsinki, Sally Floyd – ICIR.

[14] Nihal K. G. Samara Weera and Godred Fairhurst, *Reinforcement of TCP Error Recovery for Wireless Communication*, Elxctronics Research Group, Departement of Engineering, University of Aberdeen, UK.

[15] Song Cen, Pamela C. Cosman and Geoffrey M. Voelker, *End-to-end differentiation of congestion and wireless losses*.

[16] Sarra Ben Oubira, *Réservation des ressources pour les systèmes mobiles*, Rapport de projet de fin d'études, Ecole Supérieure des Communications de Tunis, Juillet 2004.

[17] Hamdi GHARBI, *Implémentation et Simulation d'un Protocole de Routage avec Qualité de Service dans les Réseaux Ad Hoc,* Rapport de projet de fin d'études, Ecole Supérieure des Communications de Tunis, Juillet 2004.

[18] Inés Ben Brahim, *Le routage et la gestion de la qualité de service dans les réseaux Ad Hoc,* Mémoire, Ecole Nationale d'Ingénieur de Sfax, Juillet 2004.

[19] George Xylomenos, *An approach of Enhancing Internet Performance over Wireless Links,* University of California, San Diego.

[20] Andrei Gutrov, *TCP Performance in the Presence of Congestion and Corruption losses,* University of Helsinki, Dept. of Computer Science, December 2000.

[21] Maulin Patel, Nisarg Tanna, Patrik Patel, Raja Banerjee, *TCP over Wireless Networks: Issues, Challenges and Survey of Solutions,* Computer Science Departement, University of Texas at Dallas, November 2001.

[22] Sonia Fahmy, Venkatesh Prabhakar, Srinivas R. Avasarala, Ossama M. Younis, *TCP Over Wireless Links: Mechanisms and Implications,* Department of Computer Sciences, Purdue University.