International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Performance Analysis of TCP Variants under MANET Environment and using NS-2

Anwar Khan[1], Dharmendra Sharma[2]

[1]Pursuing M.E., S D Bansal, Indore, India
[2]Assistant Professor, Dept of C&S, S D Bansal, Indore, India

Abstract: ***TCP was designed for wired networks and the sender assumes that packet loss is an indicator of network congestion, but this assumption may not apply to Mobile Ad hoc Networks (MANETs). In Mobile Ad Hoc networks, performance of the standard TCP is significantly degraded due to characteristics of MANET such as route failures due to node mobility and link errors. In this paper, the authors investigate the performance of TCP variants such as New Reno, SACK, TCP Tahoe and Vegas. Simulation results from the implementation of different static scenarios have been obtained. Different routing protocols such as Ad hoc On-Demand Distance Vector (AODV) and Destination Sequence Distance Vector (DSDV) have been investigated to obtain the performance of TCP variants in this paper.***

Keywords: TCP, MANET, ADHOC, NS-2, Variants

1. Introduction

Mobile Ad hoc Networks (MANETs) are a collection of mobile nodes forming a dynamic autonomous network. Nodes communicate with each other without the intervention of centralized access points or base stations. In such a network, each node acts both as a router and as a host. A MANET has several advantages over traditional wireless networks, including ease of deployment, speed of deployment, and decreased dependence on a fixed infrastructure. Typical applications of MANETs include personal communication with laptops and PDAs, group communication at conferences and presentations, communication in military, between moving vehicles and in emergency situations. Though MANETs are becoming extremely popular with the advent of various types of mobile devices; rapidly changing connectivity, network partitions, higher error rates, security threats, frequent collision probability, bandwidth and power constraints together pose new problems in designing protocols. This paper is organized as follows. A brief introduction to TCP protocol operations is given in section II. In section III, an overview of routing protocols is presented. Section IV presents the simulation environment and topologies. Section V gives discussion and analysis. Finally, summary and conclusion of the paper are given in section VI.

2. TCP (Transmission Control Protocol)

Congestion Control Algorithm

TCP is known as a full duplex protocol meaning each TCP connection provides a pair of byte streams in both directions. TCP implements the congestion control mechanism with each of these byte streams so that the receiver can limit the sender from transmitting more data in the network [5]. This section discusses about four intertwined congestion control mechanisms: slow start, congestion avoidance, fast retransmit and fast recovery. A TCP must not be more aggressive in sending data than these four algorithms allow.

Slow Start and Congestion Avoidance

The TCP sender employs the slow start and congestion avoidance algorithms to avoid more data to be sent in the network than it is capable of. For implementing these algorithms, two flow control variables, namely, the congestion window and the advertised window are included in each TCP connection state. The TCP sender imposes the congestion window while the receiver imposes the advertised window. The minimum of the congestion window and the advertised window regulates the data transmission. Besides, The slow start threshold (ssthresh), known as a state variable, is used to decide which one is to be used among the slow start or congestion avoidance algorithms for controlling the data transmission. During the beginning of the transmission, there are many unfamiliar conditions present in the network; therefore TCP needs to gradually discover the network by assessing the bandwidth and determining the available capacity [6]. This will eventually prevent the network from being congested with large bursts of data.

Figure 1 shows the slow start and congestion avoidance mechanisms executed by the TCP. Upon establishing a new connection, TCP starts the slow start mechanisms and sets the congestion window size to one segment. The congestion window size is incremented by one for each ACK received by the TCP sender. Thus, 1 packet is sent in the first round trip time (RTT), 2 packets are for the second RTT, 4 packets are for the third RTT and continue incrementing exponentially. This is why slow start phase is also known as the exponential growth phase where slow start increases the window size by the number of segments acknowledged. This process will be continuing until either of the following situations occurs:

1. An acknowledgment is not received for some segments.
2. A predetermined slow start threshold value is reached.
3. The congestion window size becomes equal to the receiver's advertised window size.

If either of these events takes place, TCP enters the congestion avoidance (linear growth) phase. Each time an ACK is received, congestion avoidance suggests that the congestion window size should be increased by (segment size*segment size)/congestion window [8]. Here, segment size and congestion window is maintained in bytes**.**

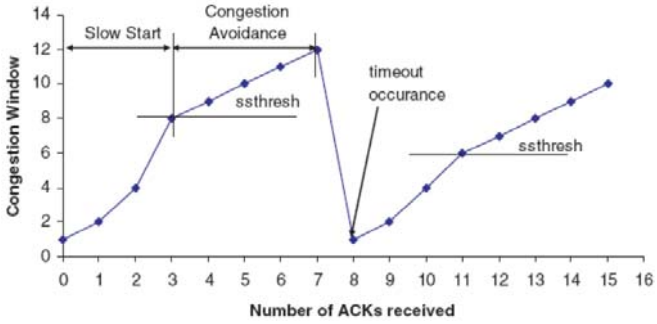

Figure 1 Slow start and congestion avoidance mechanism [7]

Fast Retransmission and Fast Recovery

Whenever a packet segment is transmitted, TCP sets a timer each time and thus ensures the reliability. TCP retransmits the packet, if it does not obtain any acknowledgement within the fixed time-out interval. The reason for not getting any acknowledgement within a specific duration is due to either the packet loss or the network congestion. Therefore the TCP sender implements the fast retransmit algorithm for identifying and repairing the loss. This fast retransmit phase is applied mainly based on the incoming duplicate ACKs. As TCP is not able to understand whether a packet loss or an out-of-order segment causes the generation of the duplicate ACK, it waits for more duplicate ACKs to be received [9]. Because in case of out-of order segment, one or two duplicate ACKs will be received before the reordered segment is processed. On the other hand, if there are at least three duplicate ACKs in a row, it can be assumed that a segment has been lost. In that case, the sender will retransmit the missing data packets without waiting for a retransmission timer to expire.

After the missing segment is retransmitted, the TCP will initiate the fast recovery mechanism until a non-duplicate ACK arrives. The fast recovery algorithm is an improvement of congestion control mechanism that ensures higher throughput even during moderate congestion [6]. The receiver yields the duplicate ACK only when another segment is reached to it; therefore this segment is kept in the receiver's buffer and does not consume any network resources. This means, data flow is still running in the network, and TCP is reluctant to reduce the flow immediately by moving into the slow start phase. Thus, in fast recovery algorithm, congestion avoidance phase is again invoked instead of slow start phase as soon as the fast retransmission mechanism is completed.

In our work we compare the four TCP variants which are as follows:

- TCP Tahoe
- New RENO
- SACK
- Vegas

TCP Tahoe

Modern TCP implementations contain a number of algorithms aimed at controlling network congestion while maintaining good user throughput. Early TCP implementations followed a go-back- model using cumulative positive acknowledgment and requiring a retransmit timer expiration to re-send data lost during transport. These TCPs did little to minimize network congestion.

The Tahoe TCP implementation added a number of new algorithms and refinements to earlier implementations. The new algorithms include Slow-Start, Congestion Avoidance, and Fast Retransmit. The refinements include a modification to the round-trip time estimator used to set retransmission timeout values. All modifications have been described elsewhere.

The Fast Retransmit algorithm is of special interest in this paper because it is modified in subsequent versions of TCP. With Fast Retransmit, after receiving a small number of duplicate acknowledgments for the same TCP segment (dup ACKs), the data sender infers that a packet has been lost and retransmits the packet without waiting for a retransmission timer to expire, leading to higher channel utilization and connection throughput.

New-Reno TCP

We include New-Reno TCP in this paper to show how a simple change to TCP makes it possible to avoid some of the performance problems of Reno TCP without the addition of SACK. At the same time, we use New-Reno TCP to explore the fundamental limitations of TCP performance in the absence of SACK. The New-Reno TCP in this paper includes a small change to the Reno algorithm at the sender that eliminates Reno's wait for a retransmit timer when multiple packets are lost from a window. The change concerns the sender' s behavior during Fast Recovery when a *partial ACK* is received that acknowledges some but not all of the packets that were outstanding at the start of that Fast Recovery period. In Reno, partial ACKs take TCP out of Fast Recovery by “deflating” the usable window back to the size of the congestion window. In New-Reno, partial ACKs do not take TCP out of Fast Recovery. Instead, partial ACKs received during Fast Recovery are treated as an indication that the packet immediately following the acknowledged packet in the sequence space has been lost, and should be retransmitted. Thus, when multiple packets are lost from a single window of data, New-Reno can recover without a retransmission timeout, retransmitting one lost packet per round-trip time until all of the lost packets from that window have been retransmitted. New-Reno remains in Fast Recovery until all of the data outstanding when Fast Recovery was initiated has been acknowledged.

SACK

TCP with 'Selective Acknowledgments' is an extension of TCP RENO and it works around the problems face by TCP RENO and TCP New-Reno, namely detection of multiple lost packets, and re-transmission of more than one lost packet per RTT. SACK retains the slow-start and fast-retransmits parts of RENO. It also has the coarse grained timeout of Tahoe to fall back on; increase a packet loss is not detected by the modified algorithm. SACK TCP requires that segments not be acknowledged cumulatively but should be acknowledged selectively. Thus each ACK has a block which describes which segments are being acknowledged. Thus the sender has a picture of which segments have been acknowledged and which are still outstanding. Whenever the sender enters fast recovery, it initializes a variable pipe which is an estimate of how much data is outstanding in the network, and it also set CWND to half the current size. Every time it receives an ACK it reduces the pipe by 1 and every time it re-transmits a segment it increments it by 1. Whenever the pipe goes smaller than the CWD window it checks which segments are un-received and send them. If there are no such segments outstanding then it sends a new packet [11]. Thus more than one lost segment can be sent in one RTT.

VEGAS

Vegas is a TCP implementation which is a modification of Reno. It builds on the fact that proactive measures to encounter congestion are much more efficient than reactive ones. It tried to get around the problem of coarse grain timeouts by suggesting an algorithm which checks for timeouts at a very efficient schedule. Also it overcomes the problem of requiring enough duplicate acknowledgements to detect a packet loss, and it also suggests modified slow start algorithms which prevent it from congesting the network. It does not depend solely on packet loss as a sign of congestion. It detects congestion before the packet losses occur. However it still retains the other mechanism of Reno and Tahoe, and a packet loss can still be detected by the coarse grain timeout of the other mechanisms fail.

3. Routing in MANET

Routing protocols in MANET are categorized as: proactive and reactive routing protocols. In the following subsections a brief description for each of them is given.

Proactive (Table-Driven) Routing Protocols

In this category, routing protocols such as Destination Sequence Distance Vector (DSDV) [7] attempt to maintain consistent and up-to-date routing information from each node to every other node in the network. This kind of approach has the property of lower latency and higher overhead. In DSDV the routes to all destinations are readily available at every node at all times. Here messages are passed between nodes to maintain their routing table. The routing table of each node consists of the shortest paths to all destinations from it. Each routing table entry in DSDV is tagged with a sequence number which is provided by the destination node. This is used to avoid the count-to-infinity problem associated with distance-vector protocols. The messages used in DSDV to maintain routing table are:

- Periodic update messages
- Triggered update messages

The periodic update messages are those in which the whole routing table is transmitted to all neighbors of a node, at regular intervals of time. Triggered update messages are transmitted when there is any change in network topology. In triggered update message only significant changes in routing table are transmitted between the nodes.

Reactive (On-Demand) Routing Protocols

In this category, routing protocols such as Ad hoc On-Demand Distance Vector (AODV) [8] source-initiated on-demand routing which creates routes only when desired by the source node. When a node requires a route to destination, it initiates a route discovery process within the network. In general, on-demand routing protocols are characterized by higher latency and lower overhead. AODV includes loop freedom and that link breakages cause immediate notifications to be sent to the affected set of nodes. Additionally, AODV has support for multicast routing and avoids the Bellman Ford "counting to infinity" problem. The use of destination sequence numbers guarantees that a route is fresh.

AODV uses different messages to discover and maintain links. Whenever a node wants to try and find a route to another node, it broadcasts a Route Request (RREQ) to all its neighbors. The RREQ propagates through the network until it reaches the destination or a node with a fresh enough route to the destination, then the route is made available by unicasting a RREP back to the source. The algorithm uses hello messages (a special RREP) that are broadcasted periodically to the immediate neighbors. These hello messages are local advertisements for the continued presence of the node and neighbors using routes through the broadcasting node will continue to mark the routes as valid. If hello messages stop coming from a particular node, the neighbor can assume that the node has moved away and mark that link to the node as broken and notify the affected set of nodes by sending a link failure notification (a special RREP) to that set of nodes.

4. Simulation Environment

We have implemented our work i.e. Creation of MANET Scenario for NS-2 and then to analyze Different routing protocols with the use of Various performance matrices Like Packet Delivery Ratio, End to End delay, Residual Energy and Overall Throughput. In our case firstly we have created scenario file for IEEE 802.11 standard which has to be used along with our TCL Script than we have created a TCL script consist of various routing protocols in our case these are AODV and DSDV than a particular MANET scenario or topology in our case it consist of 50, 75 and 100 static nodes with various TCP variants which

are NEW RENO, SACK, TCP TAHOE and VEGAS for Two Ray Ground model.

Our implementation consists of typical installation process of ns-2 complexity of topography creation and a detailed understanding of AWK scripts. In our case the total environment size taken is of 2 KM.

End to End Delay

The end-to-end delay is the time needed to traverse from the source node to the destination node in a network. The end-to-end delay is measured in ms, The delay assesses the ability of the routing protocols in terms of use- efficiency of the network resources.

End to End Delay for 50 nodes: Figure shows the End to End Delay under various protocols i.e. AODV and DSDV for the 50 nodes.

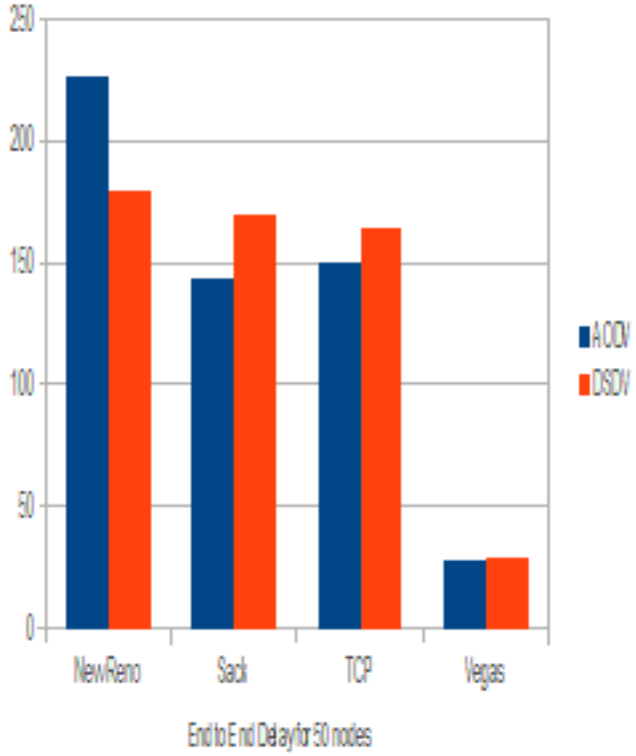

Figure 2 End to End Delay for 50 nodes

End to End Delay for 75 nodes:- Figure shows the End to End Delay under various protocols i.e. AODV and DSDV for the 75 nodes.

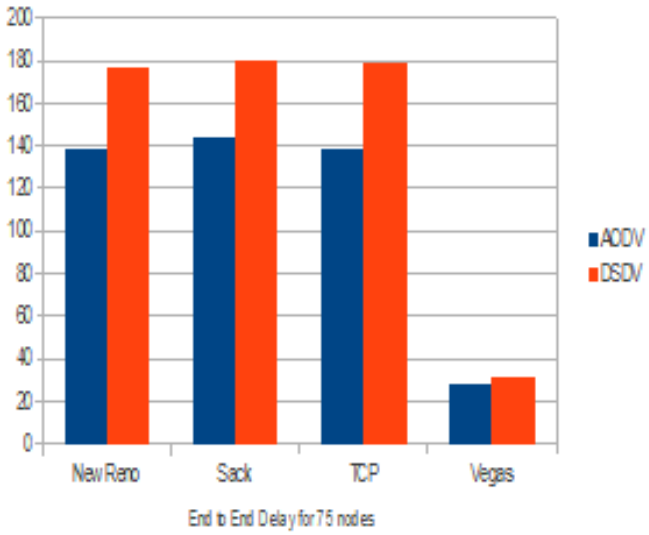

Figure 3 End to End Delay for 75 nodes

End to End Delay for 100 nodes:- Figure shows the End to End Delay under various protocols i.e. AODV and DSDV for the 100 nodes.

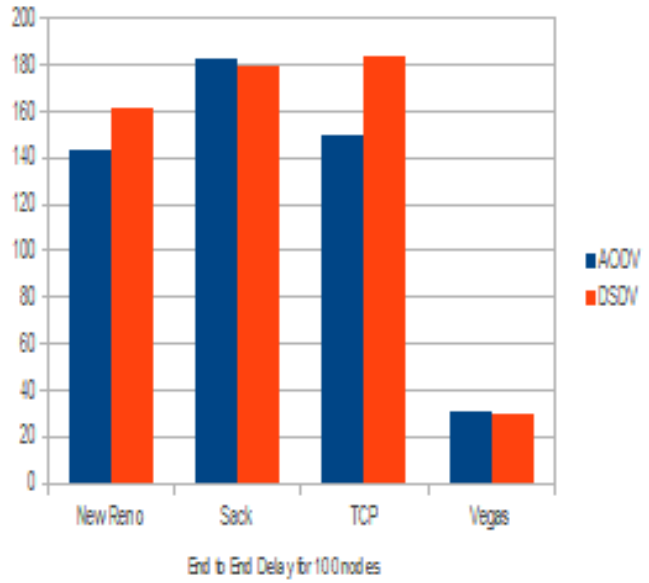

Figure 4 End to End Delay for 100 nodes

5. Analysis of End to End Delay

From above figures which are 4.4, 4.5 and 4.6 shows the End to End Delay in ms for AODV and DSDV protocols with various node densities which are 50, 75 and 100 nodes. From there it is clear that for NEWRENO End to End Delay of AODV very large as compare to DSDV for 50 nodes but when we analyze for 75 nodes and 100 nodes AODV much better than DSDV, when we analyze for 50 nodes and 75 nodes AODV have small End to End Delay as compare to DSDV for SACK, TCP TAHOE and VEGAS. But when the node density increased End to End Delay of ADOV with SACK and VEGAS increased as compare to DSDV and for TCP TAHOE DSDV remain better as compare to AODV.

Throughput

The average rate at which the data packet is delivered successfully from one node to another over a communication network is known as throughput. The throughput is usually measured in bits per second (bits/sec). A throughput with a higher value is more often an absolute choice in every network.

Throughput for 50 nodes: Figure shows the Throughput under various protocols i.e. AODV and DSDV for the 50 nodes.

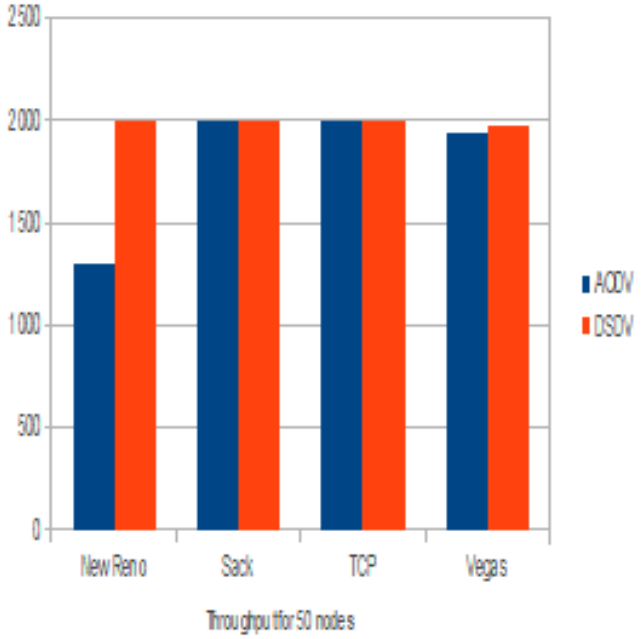

Figure 5 Throughput for 50 nodes

Throughput for 75 nodes: Figure shows the Throughput under various protocols i.e. AODV and DSDV for the 75 nodes.

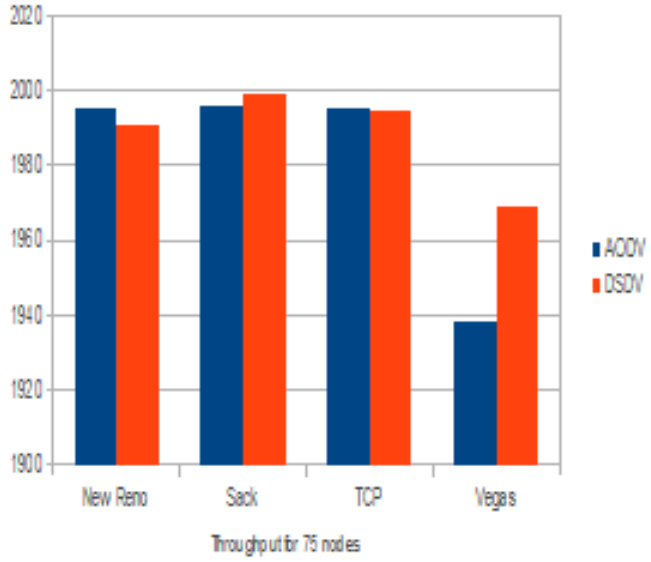

Figure 6 Throughput for 75 nodes

Throughput for 100 nodes: Figure shows the Throughput under various protocols i.e. AODV and DSDV for the 100 nodes.

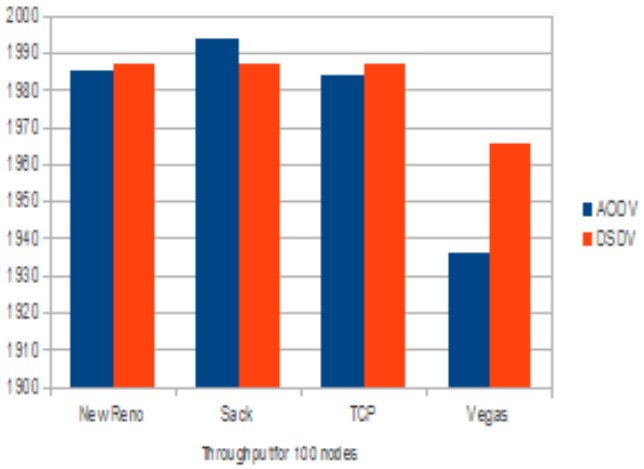

Figure 7 Throughput for 100 nodes

6. Analysis of Throughput

From above figures which are 4.7, 4.8 and 4.9 shows the Throughput for AODV and DSDV protocols with various node densities which are 50, 75 and 100 nodes. From there it is clear that the overall Throughput of DSDV protocol with Vegas is good as compare to the other variants with AODV and DSDV. So we analyze that the DSDV is better as compare to AODV protocol in terms of Throughput.

7. Conclusion

From our Result it is clear that the TCP variant SACK is best between these four variants along with VEGAS is burst variant in terms of END to END Delay and Throughput. When we analyze Protocols we cannot analyze clearly that which one is better because with different scenario both protocols gives better performance, but when we analyze for END to END Delay and Throughput DSDV better for high node density and AODV gives better results for low and moderate node density.

8. Future Scope

We have done the work of TCP variants comparison for MANET scenario the same can be done on WSN scenario as well as on some other protocols also.

References

[1] S. Senouci, and G. Pujolle, “Energy efficient consumption in wireless ad hoc networks” IEEE ICC2004 (International conference on communications), Paris JUNE 2004.
[2] M. Zorzi and R. Rao, “Energy Efficiency of TCP in a local wireless environment “mobile networks and applications,, vol. 6, no. 3, July 2001.
[3] S. Agrawal and S. Singh, “An Experimental Study of TCP’s Energy Consumption over a Wireless Link” 4th European personal Mobile Communications Conference, Feb 20-22, 2001, Vienna Austia.
[4] H. Singh and S. Singh, “Energy consumption of TCP Reno, New Reno, and SACK in multi-hop wireless

networks", in ACM SIGMETRICS 2002, June 15-19 2002.
[5] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit and fast recovery algorithms," *RFC 2001*, IETF, January 1997.
[6] V. Bhanumathi and R. Dhanasekaran, "TCP variants - A comparative analysis for high bandwidth – delay product in mobile adhoc network," in *2nd International Conference on Computer and Automation Engineering (ICCAE), 2010*, Singapore, 2010, pp. 600-604.
[7] D. Kliazovich and F. Granelli, "Cross-layer congestion control in ad hoc wireless networks," *Ad Hoc Networks*, vol. 4, no. 6, pp. 687-708, November 2006.
[8] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," *RFC 2581(Proposed Standard), Obsoleted by RFC 5681*, IETF, September 2009.
[9] Y. G. Doudane, S. M. Senouci, and A. S. Ghaleb, "A performance study of TCP variants in terms of energy consumption and average goodput within a static ad hoc environment," in *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, New York, NY, USA, pp. 503-508, 2006.
[10] V. Jacobson., "Congestion avoidance and control", SIGCOMM symposium on communications architectures and protocols, pages 314-329, 1988. An updated version is available via ftp://ftp.ee.lbl.gov/papers/congavoid.ps.z.
[11] K. Fall and S. Floyd., "Simulation-based comparison of Tahoe, Reno, and sack TCP", in ACM computer communications review, july 1996.
[12] Thomas Clausen, "Comparative Study of Routing Protocols for Mobile Ad-Hoc NETworks", INRIA, March 2004.
[13] L. Brakmo and L. Peterson, "TCP Vegas: end-to-end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communication*, vol. 13, pp. 1465-1480, Oct. 1995.
[14] A. Huhtonen, "Comparing AODV and OLSR Routing Protocols", session on Internetworking, April 2004.
[15] D. Triantafyllidou and K. Al Agha, "Evaluation of TCP performance in MANETs using an optimized scalable simulation model," in *15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007, MASCOTS '07*, pp. 31-37, November 2008.
[16] K.Kathiravan, S. Thamarai Selvi, and A.Selvam, "TCP performance analysis for mobile adhoc network using on-demand routing protocols," *Ubiquitous Computing and Communication Journal*, pp. 370-376, April 2007.
[17] A. Al Hanbali, E. Altman and P. Nain, "A survey of TCP over mobile ad hoc networks," *Research Report no. 5182, INRIA Sophia Antipolis research unit*, May 2004