International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-3 Number-1 Issue-8 March-2013

A Novel Approach for Improving the Performance of TCP by TCP Reno and SACK Acknowledgement in high traffic density conditions with cognitive radios

Reena rai[1] and Maneesh Shreevastava[2]
Department of Information Technology LNCT Bhopal, (M.P.) India[1,2]

Abstract

The key factors that contribute to TCP's performance degradation as TCP losses, MAC link failure detection latency, Route computation time and, Link failure notification latency Transmission Control Protocol (TCP) is the dominating end-to-end transport layer protocol which provides secure and reliable data transfer with some other protocols. We identify the key factors that contribute to TCP's performance with evaluate the congestion control algorithms in Reno, Vegas and SACK TCP from different aspects. In this review paper, we contend that existing approaches to improve TCP performance over mobile ad-hoc networks have focused only on a subset of the factors affecting TCP performance by SACK and TCP. For Effective resource utilization, such retransmission rate, bandwidth utilization, and packet window size, is compared. Our objective to improve the performance of TCP Reno, TCP Vegas and TCP SACK from many aspects of the both TCP Vegas and TCP SACK make some performance improvements to TCP Reno. and this paper is also concern fair resource allocation from two main categories, one is competition between different TCP congestion control algorithms and the other is fairness between different delay links.

Keywords

TCP Reno, SACK and Vegas, TCP, MANETs, Wireless, routing protocol, data transmissions, destination, TCP performance, TCP's timers.

1. Introduction

Early TCP implementation uses go-back-n model with cumulative positive acknowledgement and requires a retransmit time-out to retransmit the lost packet. These TCP is used to minimize network congestion. The operation of TCP in wireless/mobile communications has been an important research issue in recent years, owing to the impressive growth experienced in that area of modern telecommunications during the past decade. In our Paper, we will evaluate the congestion control algorithms in Reno, Vegas and SACK TCP from different aspects. First, we will compare the performance of these algorithms: how much of the available network bandwidth does it utilize? How frequently does it retransmit packets? How does TCP help to modify window size on congestion? These comparisons are based on each version TCP running separately on a congested network. The second evaluation is the fairness of sharing the network. This comparison is taken in two categories of experiment. One is the fairness between different delay connections running the same version TCP. Significant contributions, such as the one presented in [1], indicate that the unmodified, standardized operation of TCP is not well aligned with the peculiarities of cellular environments. Terminal movement across cell boundaries, leading to handover, is misinterpreted by common TCP implementations as sign of congestion within the fixed network. Some time to handle such congestion, TCP slows down transmission by retransmissions and reducing window sizes, if any relevant need arises.

Some algorithms may bias against long delay connection, such as Reno TCP and SACK. The other experiment is carried out between different versions TCP when they compete each other on the same connection. TCP Vegas does not receive a fair share of bandwidth when competing with other TCP Reno or SACK connections. Since bias exists in both categories, how different queue algorithms may affect the fairness is also studied.

Our assumption that packet losses due to network loss are minimal and most of the packet losses are due to buffer overflows at the router. It becomes increasingly important for TCP to react to a packet loss, take action to reduce congestion. TCP ensures reliability by starting a timer whenever it sends a segment. If it does not receive an acknowledgement from the receiver within the 'time-out' interval then it retransmits the segment. In the end we shall do a head to head comparison to further bring into light the differences.

2. Related Work

Transmission Control Protocol (TCP) [9] is the dominating end-to-end transport layer protocol which provides secure and reliable data transfer together with some other protocols. In this paper, they contend that existing approaches to improve TCP performance over mobile ad-hoc networks, and it have focused only on a subset of the factors affecting TCP performance by TCP Reno, Vegas and SACK. Effective resource utilization, such as bandwidth utilization. for evaluate these TCP congestion control algorithms from many aspects are present and they also concern fair resource allocation from two main categories, 1 is fairness between different delay links, and the other is competition between different TCP congestion control algo.

In this Research Paper [2] they implemented Multipath routing algo for heterogeneous network. Multipath routing separates the traffic among different paths to minimize congestion in terms of multiple alternative paths through a network which can provide a variety of benefits such as minimize delay and congestion, improved security, or maximize bandwidth. they propose a newly improved QoS multipath routing algorithm for heterogeneous networks.

Different types of adhoc routing protocols are discussed in this paper such as Ad-Hoc On Demand Multipath Distance Vector (AOMDV), QoS Ad-Hoc On Demand Multipath Distance Vector (QAOMDV), Ad-Hoc On-Demand Distance Vector (AODV). These routing protocol are used in wireless network which is designed to form multiple routes from source to the destination and also avoid the loop formation so that it reduces congestion in the channel. The performance of AODV ,AOMDV,QAOMDV protocols are compared and proved the new routing protocol is better than others. The QAOMDV works better than other protocols in terms of delay, load balance, bandwidth, outing overhead and packet delivery ratio have been considered by varying the traffic load in the network.

This paper analyzes the performance of different multi-path routing algorithms such as AOMDV, AODV and QAOMDV routing algorithms for wireless segment of heterogeneous network has been compared. The heterogeneous network is the combination of fixed and mobile network. Multipath routing protocols that computes multiple paths during route discovery avoids high overhead, latency and bandwidth. It is observed the performance of a QoS multipath routing protocol of AOMDV, QAOMDV, , is efficient than AODV, DSR, AOMDV and DSDV.

Their Simulation results shows that the performance of QAOMDV is better than other routing protocol in wireless side and hierarchical routing is used in wired network. they proved that Multipath routing algorithm provides low delay and high throughput, better bandwidth utilization and low packet loss during data transmission. Finally the Timing analysis gives the comparison between different traffic pattern and Different routing protocols are compared by Average End to End delay with pause time.

3. Proposed Technique

TCP congestion control lies in Additive Increase Multiplicative Decrease (AIMD), halving the congestion window for every window containing a packet loss, and increasing the congestion window by roughly one segment per RTT otherwise. and TCP congestion control is the Retransmit Timer, including the exponential bakeoffs of the retransmit timer when a retransmitted packet is itself dropped. The 3rd fundamental component is the show Slow-Start mechanism for the initial probing for available bandwidth. The 4th TCP congestion control mechanism is ACK-clocking, where the arrival of ACK at the sender is used to clock out the transmission of new data.

A. Communication Model

In the scenario used in this study, five mobile nodes communicate with one of two fixed nodes (hosts) located on the Internet through a gateway. As the goal of the simulations was to compare the different approaches for gateway discovery, the Traffic/CBR source was chosen to be a constant bit rate (CBR) source. Each source mobile node generates packets every 0.2 seconds in this study. In the other words, each source generates only 5 packets per second. Since each packet contain 256 bytes of data, the amount of generated data is 5*256*8 bit/s = 10.24 kbit/s, for each source. The main parameters in MTCbrSim.tcl are \connections" (number of sources) and \rate" (packet rate); see Table 1, and 2

Table 1: The Main Parameters of Reno

S No	Parameter	Value
1	Channel type	Channel/WirelessChannel
2	Radio-propagation model	Propagation/TwoRayGround
3	Antenna type	Antenna/OmniAntenna
4	Link layer type	LL
5	Interface queue type	Queue/DropTail/PriQueue
6	Max packet in ifq (interface priority queue)	50
7	Network interface type	Phy/WirelessPhy
8	MAC type	Mac/802_11
9	Ad-hoc routing protocol	Reno/ SACK
10	Number of mobile nodes	4
11	X dimension of the topography	300
12	Y dimension of the topography	250
13	Simulation time	100 ms

The TCP variants discussed in this paper, except TCP Vegas, all adhere to this underlying framework of Slow-Start, AIMD, Retransmit Timers, and ACK-clocking. None of these changes alter the fundamental underlying dynamics of TCP congestion control. Instead, these changes help to avoid unnecessary Retransmit Timeouts, correct unnecessary Fast Retransmits and Retransmit Timeouts resulting from disordered or delayed packets, and reduce unnecessary costs (in delay and unnecessary retransmits) associated with the mechanism of congestion notification.

(a)TCP congestion control:

- Main algorithms
- Slow start
- Congestion Avoidance
- Fast Retransmit
- Fast Recovery
- TCP SACK (Selective Acknowledgement)

(b)TCP Tahoe:

The Tahoe TCP implementation added a number of new algorithms and refinements to earlier TCP implementations. The new algorithms include Slow-Start, Congestion Avoidance, and Fast Retransmit [3]. 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 [4].

(c)TCP Reno:

The Reno TCP implementation retained the enhancements incorporated into Tahoe TCP but modified the Fast Retransmit operation to include Fast Recovery [5]. Fast Recovery operates by assuming each dup ACK received represents a single packet having left the pipe. Thus, during Fast Recovery the TCP sender is able to make intelligent estimates of the amount of outstanding data. Reno significantly improves upon the behavior of Tahoe TCP when a single packet is dropped from a window of data, but can suffer from performance problems when multiple packets are dropped from a window of data.

Table 2: The Main Parameters of TCP- Ad-hoc

S No	Parameter	Value
1	Transmission rate	10.24 Kb/s
2	Simulation time	100 s
3	Topology size	600m x 500m
4	Number of nodes	04
5	number of sources	4
6	Traffic type	TCP/Vegas
7	Packet rate	10 packets/s
8	Packet size	1000 bytes
9	Maximum speed	20 m/s
10	Queue Size	10 ackets/s

(d).TCP SACK:

The congestion control algorithms implemented in SACK TCP are a conservative extension of Reno's congestion control, in that they use the same algorithms for increasing and decreasing the congestion window, and make minimal changes to the other congestion control algorithms. Adding SACK (Selective Acknowledgement) to TCP does not change the basic underlying congestion control algorithms. The main difference between the SACK TCP implementation and the Reno TCP implementation is in the behavior when multiple packets are dropped from one window of data.

During Fast Recovery, SACK maintains a variable called pipe that represents the estimated number of packets outstanding in the path. The sender decrements pipe by two rather than one for partial ACKs, the SACK sender never recovers more slowly than a Slow-Start. Detailed description of SACK TCP can be found in [6].

(e)TCP Vegas:
The idea is that when the network is not congested, the actual flow rate will be close to the expected flow rate. Otherwise, the actual flow rate will be smaller than the expected flow rate. TCP Vegas adopts a more sophisticated bandwidth estimation scheme. It uses the difference between expected and actual flow rates to estimate the available bandwidth in the network. TCP Vegas, using this difference in flow rates, estimates the congestion level in the network and updates the window size accordingly. This difference in the flow rates can be easily translated into the difference between the window size and the number of acknowledged packets during the round trip time, using the equation TCP Vegas tries to keep at least α packets but no more than β packets in the queues.

4. Experiment and Result Analysis

To justify the observation in [7] that TCP Reno is biased against the connections with longer delays. The reason for this behavior is as follows. While a source does not detect any congestion, it continues to increase its window size by one during one round trip time (RTT). Obviously, connections with a shorter delay can update their window sizes faster than those with longer delays, and thus capture higher bandwidths. To our understanding, TCP SACK does not change this window increasing mechanism, so we expect the same unfair behavior with TCP SACK. We try to designing the simulation scenarios as follows.

The network topology is shown in Topology fig 1. S1 and S2 will be set to be the same TCP agents, such as two Reno, two Vegas or two SACK TCP agents, respectively.

Results of X=1ms (the same propagation delay as comparison baseline) and X=23ms (the RTT of longer delay connection is 8 times of the shorter one) will be collected to show the fairness between different delay connections.

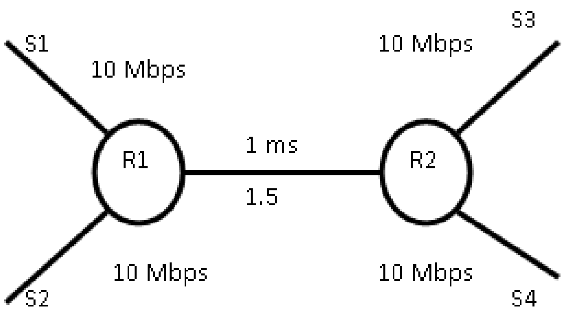

Fig 1: Topology Network

B. Simulation study

In order to verify our analytical model, the utilization of the Reno is obtained by computer simulation. A scenario is simulated using ns2 with a network of two subnets (receiver and sender) that communicate through a server with a base station. Fig 2 shows both simulated and calculated utilizations.

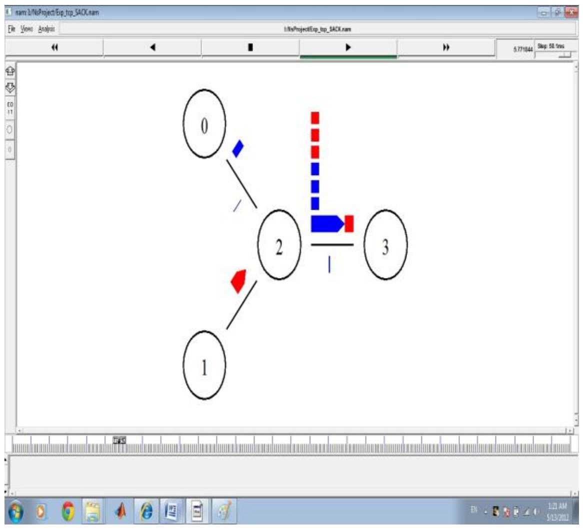

Fig 2: shows both simulated and calculated utilizations

(a). Fast retransmit algorithm:
Old TCPs would recognize the lost packets and the network congestion by a timeout mechanism. After sending a packet, the receiver waits for a period of time (RTO).
The performance of the routing protocols in terms of throughput is examined with respect to mobility of the nodes.

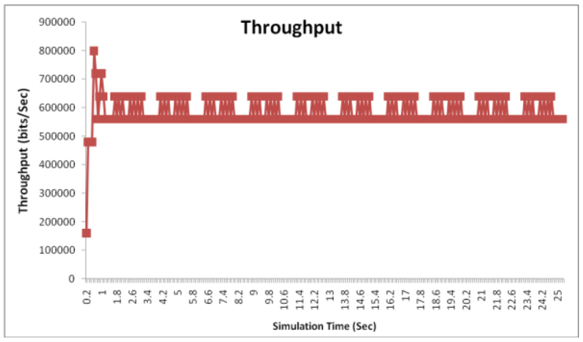

Fig 3: Average throughput for different node speeds before the bytes values on the traffic sinks

Fig 3 displays a graphical representation of analysis on the throughputs derived from various mobility scenarios before the bytes values on the traffic sinks. The X axis shows the simulation time in seconds while the Y axis shows the throughput in bits/sec. the throughput rises gradually and starts surpassing 1,50,000 bit/sec at some later stage.

The average throughput of our algorithm received in such a network is about 8,10,001 bit/sec. the medium mobility network, the throughput in a high mobility network keeps on rising gradually, however, with a lower rate than that of the medium rate network.

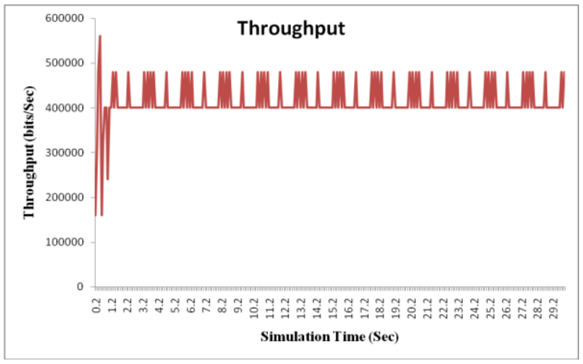

Fig 4: Average throughput for different node speeds after the bytes values on the traffic sinks

A fig 4 displays a graphical representation of analysis on the throughputs derived from various mobility scenarios after the bytes values on the traffic sinks. The X axis shows the simulation time in seconds while the Y axis shows the throughput in bits/sec. the throughput rises gradually and starts surpassing 1,50,000 bit/sec at some later stage. The average throughput of our algorithm received in such a network is about 550,001 bit/sec. the medium mobility network, the throughput in a high mobility network keeps on rising gradually, however, with a lower rate than that of the medium rate network.
Meanwhile, in the case of our algorithm, the decrease of the throughput is somewhat noticeable but not dramatic in high mobility scenarios in after the bytes values on the traffic sinks. Among the two scenarios, it appears that the low mobility results in the highest average throughput of 4, 00,000 bit/sec, which is good result as much as that of a medium and a high mobility rate .

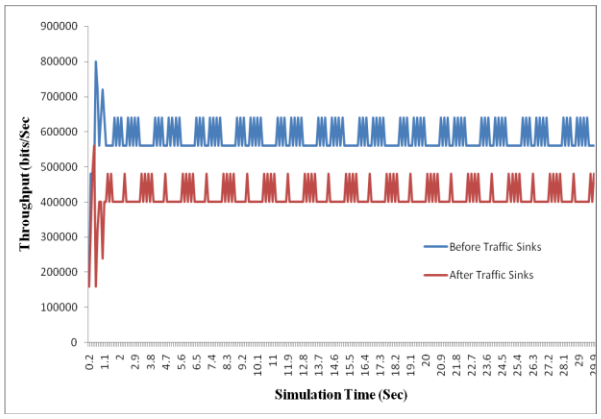

Fig 5: a comparative analysis throughput for different node speeds before and after the bytes values on the traffic sinks

C. Performance Metrics

A comprehensive list of the metrics for TCP performance evaluation is described in the TMRG RFC \Metrics for the Evaluation of Congestion Control Mechanisms" by S. Floyd. In the first step, this tool tries to implement some commonly used metrics described there. Here we follow the RFC and classify the metrics into network metrics and application metrics. They are listed as follows.

(a). Throughput
(b). Delay
(c). Jitter
(d). Loss Rate

(a). Throughput

For network metrics, we collect bottleneck link utilization as the aggregate link throughput. Throughput is sometimes different from good put, because good put consists solely of useful transmitted traffic, where throughput may also include retransmitted traffic. But users care more about the useful bits the network can provide. So the tool collects application level

End-to-end good put no matter what the transport protocol is employed. For long-lived FTP traffic, it measures the transmitted traffic during some intervals in bits per second. For short-lived web traffic, the Pack Mime HTTP model collects request/response good put and response time to measure web traffic performance.
Voice and video traffic are different from above. Their performance is affected by packet delay, delay jitter and packet loss rate as well as good put. So their good put is measured in transmitted packet rate excluding lost packets and delayed packets in excess of a predefined delay threshold.

(b). Delay
We use bottleneck queue size as an indication of queuing delay in bottlenecks. Besides mean and max/min queue size statistics, we also use percentile queue size to indicate the queue length during most of the time.
FTP traffic is not affected much by packet transmission delay. For web traffic, we report on the response time, defined as the duration between the client's sending out requests and receiving the response from the server. For streaming and interactive traffic, packet delay is a one-way measurement, as defined by the duration between sending and receiving at the end nodes.

(c). Jitter
Delay jitter is quite important for delay sensitive traffic, such as voice and video. Large jitter requires much more buffer size at the receiver side and may cause high loss rates in strict delay requirements. We employ standard packet delay deviation to show jitter for interactive and streaming traffic.

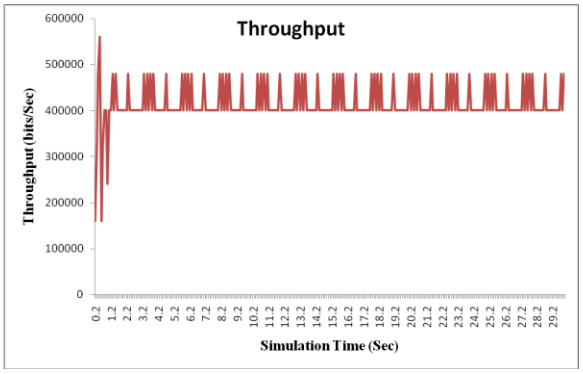

Fig 6:Throughput of NS2 Simulation on performance of TCP on mobile nodes

(d).Loss Rate
To obtain network statistics, we measure the bottleneck queue loss rate. We do not collect loss rates for FTP and web traffic because they are less affected by this metric. For interactive and streaming traffic, high packet loss rates result in the failure of the receiver to decode the packet. In this tool, they are measured during specified intervals. The received packet is considered lost if its delay is beyond a predefined threshold [8].

D. Performance Evaluation
We have changed the number of mobile nodes and measured the performance in terms of the number of total packets sent. If the numbers of mobile nodes are limited, the intermediate transmission path of the wireless link becomes unreliable and hence there is huge probability of packet losses and timeouts. As a result, the total number of packets sent is low.
Another reason for small number of packet transmission is limited antenna coverage. Increased number of mobile nodes as it uses modified quick start procedure under this packet loss and timeout conditions. But if the number of mobile nodes increases more, the optimum value of congestion and nodal delay also increases and hence the total number of packets sent falls again.
We have changed the speed of the mobile nodes and measured the total number of packet drops and percentage of packet drops. With the increase of this speed, probability of timeout increases as it performs handoff and wrong estimation of Round Trip Time (RTT). that the number of packet drops increases although the performance is not uniform. But the average performance is better than other existing approaches because of its improved functional criteria in case of timeout. The reason behind this behavior depends on the mobile ad-hoc network's topology pattern in simulation like nodes initial and final positions and their antenna parameters.

E. Loss Rate
To obtain network statistics, we measure the bottleneck queue loss rate. We do not collect loss rates for FTP and web traffic because they are less affected by this metric. For interactive and streaming traffic, high packet loss rates result in the failure of the receiver to decode the packet. In this tool, they are measured during specified intervals. The received packet is considered lost if its delay is beyond a predefined threshold.

5. Conclusion

In this research paper, we Propose to improve the performance of TCP Reno, TCP Vegas and TCP SACK from many aspects. of the both TCP Vegas and TCP SACK make some performance improvements to TCP Reno. TCP Vegas achieves higher throughput than Reno and SACK for large loss rate. TCP SACK is better when more than one packets are dropped in one window. TCP Vegas causes much fewer packets retransmissions than TCP Reno and SACK.

In conclusion, the mechanisms in the Atra contribute to Reducing the number of predicting route failures, route failures before they occur and Minimizing the latency for route error information delivery to sources, and thus, in the process, significantly improves throughput performance both when compared to the default protocol stack and an ELFN enabled protocol stack. We also suggest a change in Vegas algorithm to make Vegas more aggressive in the competition. This may be worthy of further investigation. the efforts in analysis of queuing algorithms effects lie in the gateway side of the network. There are many suggestions of modification that lie on the host side to improve the fairness.

References

[1] R. Caceres, and Iftode. "Improving the performace of reliable transport protocals in mobile computing Environments", IEEE JSAC, Vol 13, No 5, June 1995.

[2] S.Santhi, G.Sudha Sadasivam, "Performance Evaluation of Different Routing Protocols to Minimize Congestion in Heterogeneous Network", IEEE-International Conference on Recent Trends in Information Technology, PP 336-341, 2011.

[3] V. Jacobson, Congestion avoidance and control, ACM SIGCOMM Computer Communication Review, v.18 n.4, p.314-329, August 1988.

[4] Kevin Fall, Sally Floyd, Simulation-based comparisons of Tahoe, Reno and SACK TCP, ACM SIGCOMM Computer Communication Review, v.26 n.3, p.5-21, July 1996.

[5] V. Jacobson. "Modified TCP Congestion Avoidance Algorithm", Technical report, 30 Apr. 1990.

[6] Kevin Fall, Sally Floyd, Simulation-based comparisons of Tahoe, Reno and SACK TCP, ACM SIGCOMM Computer Communication Review, v.26 n.3, p.5-21.

[7] Jeonghoon Mo, Richard J. La, Venkat Anantharam, and Jean Walrand, Analysis and Comparison of TCP Reno and Vegas.

[8] G. Aggelou and R. Tafazolli, "On the Relaying Capacity of Next-Generation GSM Cellular Networks" , *IEEE Personal Communications Magazine*, pp.40–47,Feb. 2001.

[9] Mrs. Reena rai and Dr. Maneesh Shreevastava, "Performance Improvement of TCP by TCP Reno and SACK Acknowledgement", International Journal of Advanced Computer Research". ISSN (print): 2249-7277 ISSN (online): 2277-7970) Volume 2 Number 1 March 2012.

Reena rai was born in korba, dist. Korba,(Chhattisgarh) India on 26th September 1984. She received his Bachelor of Engineering Degree in Information Technology with first division and M.Tech in Information Technology. Her research interests include Computing Techniques, Security System, Robotics and Environmental with knowledge and skills for growth and development of Nation.