Analyzing Throughput of TCP in IEEE 802.11 Based Ad-hoc Networks

KOMAL TAHILIANI
SR.LECTURER(COMPUTER SCIENCE)
SHREE INSTITUTE OF SCIENCE AND TECHNOLOGY
BHOPAL(M.P)

PROF.ANJU SINGH
ASST.PROF(INFORMATION TECHNOLOGY)
BARKATULLA UNIVERSITY
BHOPAL(M.P)

DR.R.C.JAIN
DIRECTOR
SATI
VIDISHA(M.P)

Abstract

In order to allow programmers to follow conventional techniques while creating applications that uses an internet, software in the internet must provide the same semantics as a conventional computer system, it means it must guarantee reliable communication. Transport protocol provides reliability, which is fundamental for all the applications. The transmission Control protocol (TCP) is the transport level protocol that provides a completely reliable connection –oriented, full duplex stream transport service that allows two application programs to form a connection, send data in either direction, and then terminate the connection. Wireless multihop ad-hoc networks are network systems that can be deployed without relying on any infrastructure such as base stations hence are mobile networks such networks causes packet losses due to congestion and transmission errors . here a packet level model is proposed to investigate the impact of channel errors on the TCP performance over IEEE- 802.11 based multihop wireless networks A markov renewal approach is used to analyze the behaviour of TCP – Vegas . This paper motivates and describes the three key techniques employed by Vegas, and presents the results of a comprehensive experimental performance study—using both simulations and measurements the model takes into account the different proportions between the interference range and transmission range and adopting more accurate and realistic analysis to the fast recovery process. The results show that the impact of the channel error is reduced significantly due to the packet retransmissions on a per-hop basis and a small bandwidth delay product of ad hoc networks

Keywords: congestion; Interference Range.

Introduction

It is noted that TCP uses packet loss as an indicator of congestion. In wired networks, this works well because the transmission channel is so reliable and the topology is stable that the only statistically relevant cause of loss is congestion. However, in mobile networks packets can be lost by a number of different causes, such as transmission errors, link failures, and topology changes, for which TCP's response of reducing its transmission rate is inappropriate. The result is less than ideal performance Fundamental Principles of TCP the Transmission Control Protocol (TCP) is a widely used transport protocol in wired and wireless communications, layered on top of IP networks to provide reliable end-to-end congestion control. Apart from establishing, maintaining and dissolving connections between communicating pairs, a TCP agent is responsible for behaving fairly towards other network flows including other TCP agents whilst not exceeding network capacity. The way this fairness and sensible resource usage is achieved though is not explicitly specified; as such there are different TCP variants, each of which nevertheless obeys basic behavioral rules. TCP sends data in segments which do not exceed a maximum segment size as negotiated via a three-way handshake between the communicating agents during an initial connection establishment phase. Each byte (octet) of data has a sequence number assigned to it. When the receiver receives a segment, it notes the bytes of data (or sequence number range) of the segment and responds by sending back a cumulative acknowledgement (ACK) which confirms that all bytes up to the given sequence number have successfully arrived. The TCP sender also maintains a retransmission timeout (RTO) timer, which on expiration indicates that a segment has been lost and is to be retransmitted. The functionality offered by cumulative ACKs, the RTO timer as well as a checksum on the segment header and data ensures reliability on top of IP. Another important functionality of TCP is flow and congestion control through the use of Recent traffic monitoring over the Internet has confirmed the popularity of the Reno and New Reno TCP variants as well as the increasing adoption of the TCP selective acknowledgements (SACK) modification. A promising reactive solution to the problem of congestion control has further been presented in with the introduction of TCP Vegas which has received much attention in the literature.

The main result reported in this paper is that Vegas is able to achieve between 37 and 71% better throughput than Reno. Moreover, this improvement in throughput is not achieved by an aggressive retransmission strategy that effectively steals bandwidth away from TCP connections that use the current algorithms. Rather, it is achieved by a more efficient use of the available bandwidth. Our experiments show that Vegas retransmit between one fifth and one-half as much data as does Reno.

In view of the difference between traditional networks and ad-hoc networks we aim to develop a mathematical model to investigate TCP performance in ad-hoc networks , here we propose an analytical model for a single persistent TCP flow in the presence of packet losses due to channel error over an IEEE-802.11-based static multihop linear chain . We do not consider the impact of mobility, more complex topology, and interactions between multiple TCP flows ,as we would like to understand how the set of protocol stacks including TCP, internet Protocol (IP), and MAC behave in baseline scenario before exploring the impact of additional variables. In the model a markov reneval approach is used to analyse the TCP behavior. The analytical results are validated against simulation results by using NS2.

Existing System

We consider TCP in terms of rounds where a round starts when the sender begins the transmission of a window of packets and ends when the sender receives an ACK for one or more of these packets. the several rounds of fast recovery process for Reno and New Reno are best explained via the examples shown in the Fig Suppose that Packets 1, 23 and 24 are lost When the window reaches w = 24 and that packets 2 through 22 are successful . this window is called the loss window . In this case, Reno and NewReno have the same behaviour in the second round. The source first receives 21 duplicate ACK's , each with a sequence number requesting Packet 1. The first three ACK's trigger the fast retransmit of Packet 1 and cause the window to drop to 12 . Then , this window is temporarily inflated by the number of duplicate ACK's . Once the duplicate ACK triggered by Packet 13 is received, the window is inflated to 12 + 12 = 24. The number of Packets in the pipe known by the source is still 24, since each ACK carries the sequence number requesting for packet 1 . During this period , transmission of new packets is not permitted packets beyond the allowed window . for every subsequent duplicate ACK, TCP continues to inflate its window and transmits one new packet , up transmitting nine new packets , with the largest sequence number being 24+9 =33.

Reno

In reno as shown in the figure (a), This inflation is removed, and the window is cut back to 12 when the sequence number carried by ACK advances, that is , when the ACK for the retransmission comes back with a sequence number requesting Packet 23 . At that point only one new packet 34, since the outstanding packet is 33 – 22 = 11. In the third round, the lost packet 23 is retransmitted with the arrival of four duplicate ACK's requesting packets 23 . The window is further decreased in half and followed by four new packet transmissions due to the window inflation. Following this new packet can be sent. The fifth round starts with the exit of the first recovery process when packet 24 is successfully retransmitted.

NewReno

In NewReno, as shown in Fig. 1b, with the successful retransmission of packet 1, one partial ACK requesting packet 23 arrives at the TCP source. Packet 23 is immediately retransmitted without waiting for enough duplicate ACKs. The congestion window is deflated by the amount of new data acknowledged minus one segment and is 33 _ 22 þ 1 ¼ 12. One more packet 34 is allowed to be transmitted, since the outstanding packet is 11. For each additional duplicate ACK received, the congestion window is incremented, which allows more new packets to be sent, as shown in the figure. Following this logic, the congestion window is artificially inflated to 12 þ 9 ¼ 21when an ACK requesting packet 24 arrives. It is not deflated, since only one new packet is acknowledged. With the successful retransmission of packet 24, the window is cut back to 12 again.

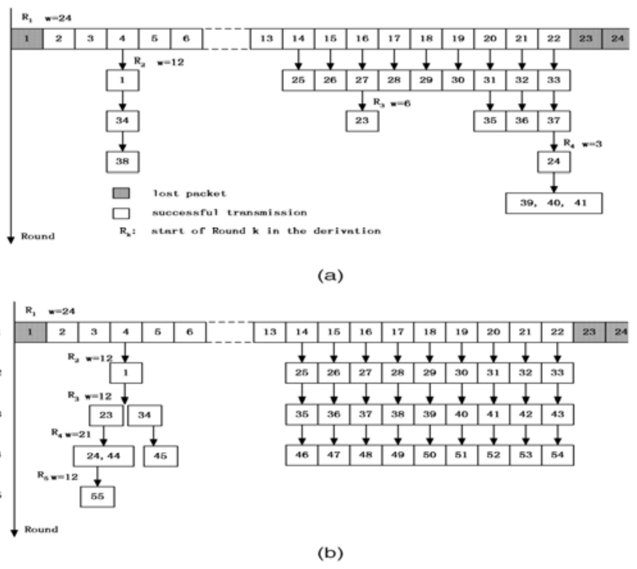


Figure . Fast Recovery Process

Proposed system

TCP Vegas

Jacobson and Karels developed a congestion control mechanism, which was later named TCP Tahoe [5]. Since then, many modification have been made to TCP, and different versions have been implemented such as TCP Tahoe and Reno .TCP Vegas was first introduced by Brakmo et al. There are several changes made in TCP Vegas. First, the congestion avoidance mechanism that TCP Vegas uses is quite different from that of TCP Tahoe or Reno. TCP Reno uses the loss of packets as a signal that there is a congestion in the network and has no way of detecting any incipient congestion before packet losses occur. Thus, TCP Reno reacts to congestion rather than attempts to prevent the congestion TCP Vegas, on the other hand, uses the difference between the estimated throughput and the measured throughput as a way of estimating the congestion state of the network. We describe the algorithm briefly here. For more details on TCP Vegas, refer to [2]. First, Vegas sets Base RTT to the smallest measured round trip time, and the expected throughput is computed according to

$$\text{Expected} = \frac{\text{WindowSize}}{\text{Base RTT}} \qquad (1)$$

Where WindowSize is the current window size. Second, Vegas calculates the current Actual throughput as follows. With each packet being sent, Vegas records the sending time of the packet by checking the system clock and computes the round trip time (RTT) by computing the elapsed time before the ACK comes back. It then computes Actual throughput using this estimated RTT

$$\text{Actual} = \frac{\text{WindowSize}}{\text{Base RTT}} \qquad (2)$$

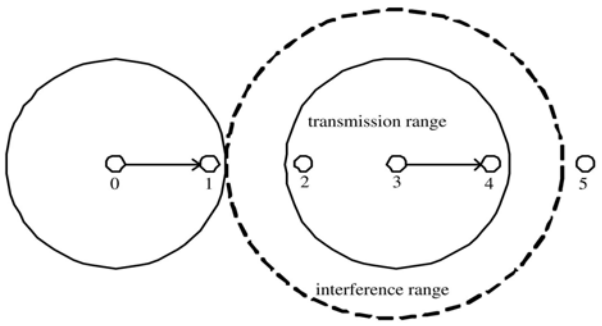
Then, Vegas compares Actual to Expected and computes the difference

$$\text{Diff} = \text{Expected} - \text{Actual}; \qquad (3)$$

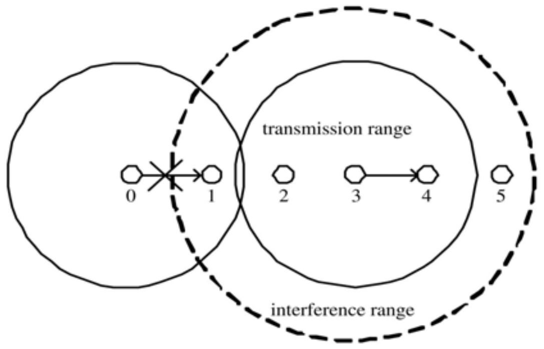
Which is used to adjust the window size? Note that Diff is non-negative by definition. Define two threshold values, $0 < a < b$. If Diff < a Vegas increases the window size linearly during the next RTT. If Diff > a then Vegas decreases the window size linearly during the next RTT. Otherwise, it leaves the window size unchanged. What TCP Vegas attempts to do is as follows. If the actual throughput is much smaller than the expected throughput, then it suggests that it is likely that the network is congested. Thus, the source should reduce the low rate. On the other hand, if the actual throughput is too close to the expected throughput, then the connection may not be utilizing the available low rate, and hence should increase the low rate. Therefore, the goal of TCP Vegas is to keep a certain number of packets or bytes in the queues of the network [2, 9]. The threshold values, a and b, can be specified in terms of number of packets rather than low rate. Now note that this mechanism used in Vegas to estimate the available bandwidth is fundamentally different from that of Reno, and does not purposely cause any packet loss. Consequently this mechanism removes the oscillatory behavior from Vegas, and Vegas achieves higher average throughput and efficiency. Moreover, since each connection keeps only a few packets in the switch buffers, the average delay and jitter tend to be much smaller. Though it has been observed that TCP Vegas when competing with other TCP Reno connections, does not receive a fair share of bandwidth, i.e., TCP Reno connections receive about 50 percent higher bandwidth. It is seen that due to the aggressive nature of TCP Reno, when the buffer sizes are large, TCP Vegas loses to TCP Reno that fulls up the available buffer space, forcing TCP Vegas to back off. TCP Vegas has problems that have not been discussed before, which could have a serious impact on the performance. It is the issue of rerouting. Since TCP Vegas uses an estimate of the propagation delay, baseRTT, to adjust its window size, it is very important for a TCP Vegas connection to be able to have an accurate estimation. Rerouting a path may change the propagation delay of the connection, and this could result in a substantial decrease in throughput.

System Model

We consider a static multihop string topology with n hops (node 0 through node n). A single persistent TCP connection is set up between the source node 0 and the sink node n. Node 0 is an infinite data source that always has packets to send. At each node, the first-in, first-out (FIFO) buffer size is infinite. The distance between any two adjacent nodes is identical. For transmission ranges, the distance satisfies the condition where transmissions from one node can only reach its one-hop neighbors. Fig. 2 depicts two different scenarios for an 802.11-based linear chain network with six nodes, where Rtx < Rin < 2Rtx. In Fig. 2a, node 1 is outside the interference range of node 3; hence, the transmission between node 0 and node 1 is successful. On the contrary, in Fig. 2b, node 0 fails to transmit to node 1. Node 1 cannot send Clear-To-Send (CTS) to node 0, since it is within the interference range of node 3, and it can hear the on going transmission of node 3. The second scenario, as depicted in Fig. 2b, corresponds to the commonly used network model, where 2Rtx _ Rin and 2Rtx < Rin. Since our objective is to investigate the impact of channel error on the TCP performance, we ignore the packet losses caused by the MAC layer contention by setting the Request-To-Send (RTS) short retry limit to infinity, as opposed to the default value of 7 in IEEE 802.11. Hence, our model does not experience any packet losses invoked by buffer overflow, mobility, and MAC layer contention. This model allows the isolation of packet losses due to different causes and aids the investigation of the impact of different loss parameters on the TCP throughput performance. Meanwhile, we assume that the TCP data packets may be lost, but the TCP ACKs are never lost due to their small sizes. However, in our model, which uses IEEE 802.11, it is not realistic to model a bursty packet error because a TCP packet transmission can occur only when the MAC layer RTS-CTS frame exchange is successful. The bad channel condition can only corrupt several consecutive transmissions of an RTS packet. In addition, a corrupted packet may not always be discarded. A TCP data packet is discarded only when its number of retransmission attempts reaches the long retry limit on one link.



(a)



(b)

Conclusion:

Reliable data delivery is the most important aspect of any network , wired networks uses TCP protocol for the purpose , but when we talk about wireless network the performance of TCP degrades due to the unstable and mobile nature of the network . Hence to improve the performance of TCP in IEEE- 802.11 based ad – hoc network with multiple wireless lossy links we propose a packet level for TCP vegas , that captures the details of the fast recovery process in the event of packet loss induced by the channel error. Considering the spatial reuse property of the wireless channel, the model takes into account the different propotions between the interference range and the transmission range. The proposed model is based on the semi – Markov renewal reward process. We find that the TCP performance for different path length varies with different values of the long retry limit, it is also shown that Vegas out performs Reno and new reno with heavy packet loss scenarios in terms of the throughput and timeout probability

References

[1] A.A. Abouzeid, S. Roy, and M. Azizoglu,(2000) “Stochastic Modeling of TCP over Lossy Links,” Proc. IEEE INFOCOM, pp. 1724-1733.
[2] M. Allman, V. Paxson, and W. Stevens(1999) “TCP Congestion Control”, IETF RFC 2581..
[3] K. Chen, Y. Xue, S.H. Shah, and K. Nahrstedt (2004) “Understand Bandwidth-Delay Product in Mobile Ad Hoc Networks,” Elsevier Computer Comm. J., special issue on protocol engineering for wired and wireless networks, vol. 27, no. 10, pp. 923-934.
[4] K. Chen, Y. Xue, and K. Nahrstedt(2002) “On Setting TCP’s Congestion Window Limit in Mobile Ad Hoc Networks,” J. Wireless Comm. and Mobile Computing, vol. 2, no. 1, pp. 85-100.
[5] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla(2003) “The Impact of Multihop Wireless Channel on TCP Throughput and Loss,” Proc. IEEE INFOCOM, vol. 3, pp. 1744-1753.
[6] S. Floyd, T. Henderson, and A. Gurtov(2004), “The NewReno Modification to TCP’s Fast Recovery Algorithm”, IETF RFC 3782.
[7] S. Floyd and T. Henderson(1999), The NewReno Modification to TCP’s Fast Recovery Algorithm, IETF RFC 2582.
[8] V. Jacobson, Modified TCP Congestion Avoidance Algorithm(1990), message to end2end-interest mailing list, ftp://ftp.ee.lbl.gov/email/vanj. 90apr30.txt.
[9] A. Kumar, “Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link,”(1998) IEEE/ACM Trans. Networking, vol. 6, pp. 485-498.

[10] A. Kumar and J. Holtzman, "Comparative Performance Analysis of Versions of TCP in a Local Network with a Mobile Radio Link,"(1998) Sadhana: Indian Academy of Sciences Proc. in Eng. Sciences, vol. 23, pp. 113-129.

[11] B. Kim and J. Lee, "A Simple Model for TCP Loss Recovery Performance over Wireless Networks," (2004)J. Comm. and Networks, vol. 6, no. 3, pp. 235-244.

[12] X. Li, P.Y. Kong, and K.C. Chua, "Analysis of TCP Throughput in IEEE-802.11-Based Multi-Hop Ad Hoc Networks,"(2005) Proc. 14th IEEE Int'l Conf. Computer Comm. and Networks (ICCCN '05), pp. 297-302.

[13] J. Li, C. Blake, D.S.J. Couto, H.I. Lee, and R. Morris, "Capacity of Ad Hoc Wireless Networks,"(2001) Proc. ACM MobiCom, pp. 61-69.

[14] T. Lakshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss,"(1997) IEEE/ACM Trans. Networking, vol. 5, no. 3, pp. 336-350.

[15] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm,"(1997), ACMComputer Comm. Rev., vol. 27, no. 3, pp. 67-82.

[16] V. Misra, W. Gong, and D. Towsley, "Stochastic Differential Equation Modeling and Analysis of TCP-Windowsize Behavior,"(1999) Proc. Int'l Federation for Information Processing WG 7.3 Conf. (Performance '99).

[17] A. Medina, M. Allman, and S. Floyd, "Measuring the Evolution of Transport Protocols in the Internet,"(2005) Computer Comm. Rev., vol. 35, no. 2.

[18] J. Padhye, V. Firoiu, D.F. Towsley, and J.F. Kurose, "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation,"(1998) ACM Computer Comm. Rev., vol. 28, pp. 303-314.

[19] Scalable Mobile Network Simulator, http://pcl.cs.ucla.edu/projects/ glomosim/, Nov. 2006.

[20] W. Stevens, TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, IETF RFC 2001, 1997.

[21] S. Xu and T. Saadawi, "Revealing the Problems with 802.11 Medium Access Control Protocol in Multi-Hop Wireless Ad Hoc Networks,"(2002), Computer Networks, vol. 38, no. 4, pp. 531-548.

[22] S. Xu and T. Saadwi, "Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks?"(2001), IEEE Comm. Magazine, vol. 39, no. 6, pp. 130-137..

[23] M. Zorzi and R. Rao, "Effect of Correlated Errors on TCP," Proc. Conf. Information Sciences and Systems (CISS '97), pp. 666-671, 1997.