IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661, ISBN: 2278-8727Volume 8, Issue 2 (Nov. - Dec. 2012), PP 37-40
www.iosrjournals.org

A Review on Snoop with Rerouting in Wired cum Wireless Networks

Rupali Advirkar[1], Ameya Naik[2]
[1, 2] *(EXTC Department, K.J. Somaiya College of Engineering, University of Mumbai, India)*

Abstract: *TCP (Transmission Control Protocol) has been performing well over the traditional wired networks. However it cannot react efficiently in wireless network where packet losses occur mostly because of congestion and wired cum wireless networks where the significant packet losses are due to bit errors or handoffs. In order to evaluate the Quality of Service it is essential to measure the performance of network in terms of throughput, efficiency, delay, packet dropping probability etc. These parameters are usually influenced by variations in the network parameters such as packet size and packet interval. Consequently it could not differentiate whether the increased round trip time is due to route change or network congestion. As a solution to this problem a mechanism (Snoop protocol) is proposed for efficient rerouting. The rerouting of packets using Snoop helps avoid congestion and to improve the performance of TCPs with rerouting.*

Keywords: *rerouting, round trip time, Snoop, wired cum wireless networks.*

I. Introduction

Today's fast growth of Internet traffic, how to efficiently utilize network resources is essential to a successful congestion control. The network itself really only provides a reliable signaling and routing system for sending small packets of data from one host to another. Packets may be lost or reordered without warning. Hosts who wish to communicate across the network must work within these constraints. Communicating hosts must provide for themselves any greater level of reliability they need. The design is based on the "end-to-end" principal [1] which places as much of the communications protocol operations as possible at the end points. There are currently two very common "transport protocols" running on the end hosts. UDP simply sends packets and provides no extra reliability. If the user doesn't get a response to a request, they are usually expected to request again if they so choose. TCP by contrast ensures that arrival of all packets at their destination is confirmed, retransmitting any which are lost; ensures that any reordering of the packets is corrected, keeps different communication sessions from interfering and attempts to send at the highest rate it can without causing excessive packet loss due to congestion in the network. This last responsibility of TCP is called "congestion control" and was added in the late eighties in response to several instances of congestion collapse on the early internet.

Congestion decisions that adjust the transmission rate based on round trip time (RTT) variations, We used TCP Vegas [2] and Snoop [3] etc, However, rerouting may lead to the change of the end-to-end fixed delay, which is the sum of propagation delay and packet processing time, and therefore bring about inaccurate estimation of the minimum round trip time (RTT_{min}). This may affect the TCP's adjustment of the congestion window size (CWND) [4], [5]. Several works [5], [6] have been proposed to improve TCP performance with rerouting, but they have some limits and drawbacks.

Proposed mechanism is Snoop with RB to counteract the rerouting problem used to assume increase in RTT for rerouting. In order to know whether the end-to-end fixed delay is changed, Snoop [7] obtains the forward and backward queuing times by performing it's accumulate queuing time (AQT) scheme in routers along the round-trip time. However, this mechanism works well only when all bottleneck routers along the round-trip path are AQT-enabled. In this paper, we investigate how to improve the performance with rerouting and proposed mechanism for different TCPs like Snoop and Vegas. The proposed mechanism is to set reserved bit and re-measure the minimum RTT. This mechanism may improve the TCP performance with re-routing in wired cum wireless networks [8].

II. Related work

2.1 TCP Vegas

TCP Vegas was first introduced by Brakmo et al. in [2]. 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. It uses the difference between the estimated throughput and the measured throughput as a way of estimating the congestion state of the network. First, Vegas set minimum RTT to the smallest measured RTT, and the expected throughput is computed. For example, TCP Vegas estimates a proper amount of extra data (Δ) to be kept in the network pipe. Moreover, Δ is between two thresholds α and β.

$$\alpha \leq (\text{Expected} - \text{Actual}) \times \text{RTT}_{min} \leq \beta \quad (1)$$

$$\text{Expected} = \frac{\text{cwnd}}{\text{RTTmin}} \quad (2)$$

Where, cwnd is the current window size.

Vegas calculate the current Actual throughput, with each packet being sent. It 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 according to,

$$\text{Actual} = \frac{\text{cwnd}}{\text{RTT}} \quad (3)$$

Then, Vegas compares Actual to Expected and computes the difference

$$\text{Diff} = \text{Expected} - \text{Actual} \quad (4)$$

which is used to adjust the window size. Note that Diff is non-negative by definition.

Define two threshold values, $0 \leq \alpha < \beta$.

1. If Diff < α -Vegas increases the window size linearly during the next RTT.
2. If Diff > β -Vegas decreases the window size linearly during the next RTT.
3. Otherwise, it leaves the window size unchanged.

Figure 1 [11] shows the expected rate is a theoretical rate of a TCP flow in a congestion-free network state. This rate can occur if all transmitted data packets are successfully acknowledged within the minimum RTT (i.e., no loss, no congestion). Assuming that RTTmin is constant, the expected rate is directly proportional to the size of the congestion window with a proportionality coefficient of 1/RTTmin. The actual rate (bold solid line in Figure.1 [11]) can be expressed as the ratio between the current congestion window and the current RTT value.

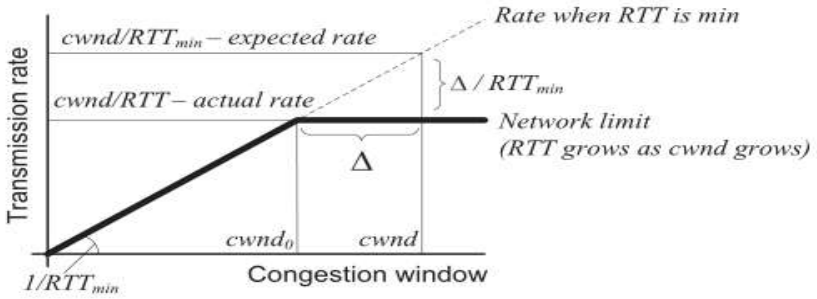

Fig. 1. TCP Vegas-Δ function of expected and actual rate

However, due to the finite capacity of the path, we can always find a point cwnd0 on the graph when the actual rate is numerically equal to the expected rate, and all attempts to send at a faster rates (i.e.,> cwnd0/RTTmin) will fail. Clearly, the number of packets enqueued during the last RTT is the difference Δ between the current congestion window and the inflection point in graph; thus we have Δ = cwnd − cwnd0. According to our assumptions, this excess of data packets is the only cause of a corresponding RTT increase. Thus, Δ can be expressed as a function of the congestion window size, RTT and RTTmin:

$$\Delta = \text{cwnd} \times \frac{\text{RTT} - \text{RTTmin}}{\text{RTT}} \quad (5)$$

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 flow 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 flow rate, and hence should increase the flow rate. Therefore, the goal of TCP Vegas is to keep a certain number of packets or bytes in the queues of the network [2]. The threshold values, α and β, can be specified in terms of number of packets rather than flow rate.

• When it receives a duplicate ACK, Vegas checks to see if the RTT is greater than timeout. If it is, then without waiting for the third duplicate ACK, it immediately retransmits the packet.

• When a non-duplicate ACK is received, if it is the first or second ACK after a retransmission, Vegas again checks to see if the RTT is greater than timeout. If it is, then Vegas retransmit the packet.

2.2 Snoop protocol

The snoop protocol developed by Berkeley is an example of a TCP-Aware link-layer protocol [10].It is a hybrid solution which uses concepts from the link-layer schemes and the split connection approaches. Unlike the other solutions discussed, the basic snoop protocol is specifically designed for data transfers from the fixed

host to the mobile host. To improve the performance of data transfers in the other direction a NACK (negative acknowledgment) may be used between the mobile host and the base station. The basic snoop protocol is also unique from the others because it only requires modifications at the base station. A snoop base station caches each packet received on the wired link and then forwards it to the mobile host. The snoop agent then watches the ACKs being returned from the mobile host to the sender. The agent maintains round-trip-time estimates for the wireless link and performs local retransmission if an ACK is not received before the timer expires. A much finer timer is used at the base station than at the TCP sender so that multiple local retransmissions are possible before the sender times out. Also, a single duplicate ACK for a packet cached at the base station results in a local retransmission. The base station is able to interpret a single duplicate ACK (rather than three) as a packet loss because the wireless link delivers packets in order. The key to snoop is that unlike other link-layer protocols, it maintains state which it is able to use to determine whether a packet was lost due to congestion (before reaching the base station), or received at the base station and subsequently lost due to errors on the wireless link. This information is used to shield the sender from duplicate ACKs due to wireless losses, and to pass through duplicate ACKs caused by congestive losses. The state maintained by snoop also allows it to recognize TCP retransmissions so that it will not compete with them.

III. The Proposed System

When a link in a network fails or heavy congestion in the network, traffic that was using the failed link must change its path in order to reach its destination. It is rerouted from a primary path to a backup path. The primary and the backup path can be totally disjoint or partially merged in case of wired network, Fig. 2 shows wired cum wireless networks. Wireless routers are free to move randomly and organize their path to reach destination.

When the route of a connection is changed, if the new route has a shorter fixed delay, it will not cause any serious problem because most likely some packets will experience shorter RTT, and minimum RTT will be updated eventually. On the other hand, if there is a longer fixed delay in the new route, it would be unable to tell whether the increased RTT is due to network congestion or route change. The source host may misinterpret the increased RTT as a signal of congestion in the network and decrease its window size. This is just the opposite of what the source should do. The Time to Live (TTL) is a timer field (8 bits) in the IP header which indicates how long a packet should be allowed to survive before it is discarded and TTL essentially determines the maximum number of hops permitted. In other words, when the TTL field is decremented down to zero, the packet is discarded. Moreover, if the routing path is not changed, the TTL value of each packet will be same. Therefore, we modify the end hosts including both the sender and receiver to detect the change of TTL value in our mechanism. In the sender side, while the sender knows that the TTL value of an acknowledgement (ACK) is changed, it will re-measure the minRTT. In the receiver side, a receiver will tell the sender to re-measure minRTT when the TTL value of a data packet is varied. Thus, we use a reserved bit in TCP header, called RB bit, to represent the change of the routing path. The sender records the value of 'RB' and the receiver will keep the 'RB' bit value of each ACK unchanged until the TTL value of a data packet is changed again. When the value of 'RB' bit changes from 0 to 1 (or from 1 to 0), it means that the forward routing path has been changed. In the beginning, the value of 'RB' bit is set to 0. Once a receiver detects the change of the forward routing path, it will alter the 'RB' bit value to inform the source to re-measure the minRTT. Whenever a sender receives an ACK, it compares the value of the 'RB' bit with its current value. If they are different, the sender will re-measure the minRTT. The proposed mechanism is called Snoop-RB.

IV. Rerouting Technique

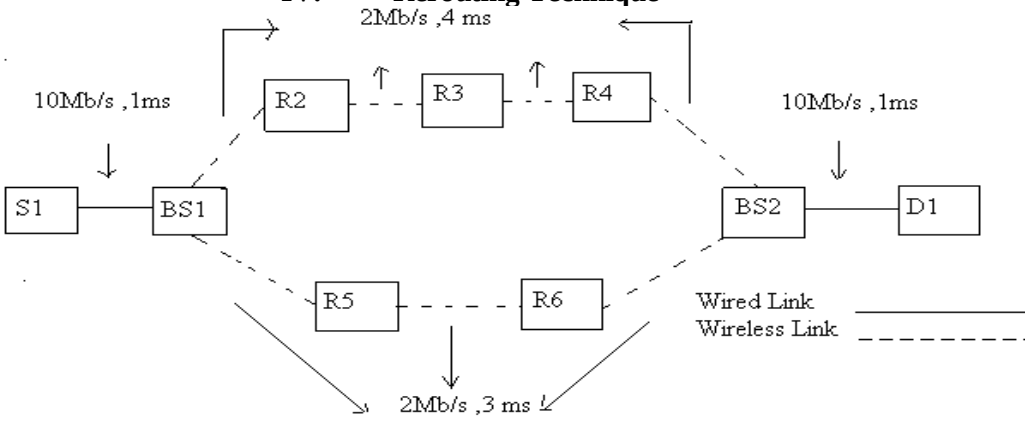

Fig. 2. A single network topology for the rerouting simulation

Figure.2 shows the first network topology. A source S1of Snoop, Snoop-RB, Vegas, Vegas-RB [12] sends data packets to its destination D1. The bandwidth and propagation delay are10 Mb/s and 1 ms for each full-duplex access link, 2Mb/s and 3 ms for the full-duplex trunk link from BS1 to R5, from R5 to R6 and from R6 to BS2, and 2Mb/s and 4 ms for the full-duplex trunk link from BS1 to R2, from R2 to R3, from R3 to R4 and from R4 to BS2. In the beginning, the packets are routed through S1, BS1, R5, R6, BS2, and D1 in order. At 20^{th} second, the connection link from BS1 to R5 is broken and then recovered at 40^{th} second. Therefore, the packets pass through the other path from 20^{th} till 40^{th} second. As shown in Fig.2, when the packets are routed through the path with shorter RTT. However, the performance of Snoop and Vegas degrades dramatically as the packets are rerouted through the other path. Snoop-RB and Vegas-RB is one of the alternatives to maintain a steady throughput regardless of the route change.

V. Conclusion

A Review on Snoop with Rerouting in Wired cum Wireless Networks shows that packet loss can be minimized when network is congested using rerouting techniques in Snoop with RB and Vegas with RB. By using the NS2 simulation, we can obtain the graphical representation to measure the network performance parameters.

Snoop and Vegas can be used in between base station and wireless routers, whereas Vegas can be used in between both wireless routers for rerouting purpose with reserved bit set.

References

[1] J.H.Saltzer ,D.P.Reed, and D.D.Clark, "End-to-end arguments in system design," ACM Trans. Comp. Sys., Vol. 2, no. 4, Nov. 1984, pp. 277-88. An earlier version appeared in 2nd int'l. Conf. Dist. Comp. Sys., Apr., 1981, pp. 509-12.

[2] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet," IEEE J. Sel. Areas Communication, vol. 13, pp. 1465-1480, 1995.

[3] "TCP Performance Enhancement using ECN & Snoop Protocol for Wi-Fi Network", Manish D Chawhan and Dr. Avichal Kapur.The 2nd IEEE International Conference on Computer and Network Technology (ICCNT'10), 2010, Bangkok, Thailand.

[4] R. J. La, J. Walrand, and V. Anantharam, "Issues in TCP Vegas,"UCB/ERL memorandum, no. M99/3, Electronics Research Laboratory, University of California, Berkeley, Jan. 1999.

[5] K. N. Srijith, L. Jacob, and A. L. Ananda, "TCP Vegas-A: improving the performance of TCP Vegas," Computer Commun., vol. 28, no. 4,pp. 429-440, Mar. 2005.

[6] Y.-C. Chan, C.-T. Chan, and Y.-C. Chen, "RoVegas: a router-based congestion avoidance mechanism for TCP Vegas," Computer Commun., vol. 27, no. 16, pp. 1624-1636, Oct. 2004.

[7] Srikanth Tiyyagura , Rajesh Nutangi , and Dr. P. Chenna Reddy ,"An Improved Snoop For TCP Reno And TCP Sack In Wired-Cum-Wireless Networks", Computer Commun.,IJCSE.,Vol. 2 No. 3 Jun-Jul 2011.

[8] C. Perkins, "IP mobility support for IPv4," IETF RFC 3344, Aug. 2002.

[9] Jeonghoon Mo, Richard J. Lo., Venkat Anantharam and Jean Walrand"Analysis And Comparison of TCP Reno and Vegas", Computer commun., University of California, Berkeley, July 13, 1998.

[10] Sarma Vangala and Miguel A. Labradorb ."Performance of TCP over Wireless Networks with the Snoop Protocol." Department of Computer Science and Engineering University of South Florida, Tampa, FL 33620.

[11] Alexander Afanasyev, Neil Tilley, Peter Reiher, and Leonard Kleinrock," Host-to-Host Congestion Control for TCP", IEEE Communications Surveys & Tutorials, VOL. 12, No. 3, Third Quarter 2010.

[12] Cheng-Yuan Ho, Yaw-Chung Chen and Cheng-Yun Ho "Improving Performance of Delay-Based TCPs with Rerouting" , IEEE Communications, VOL. 11, NO. 1, Jan 2007.