A NOVEL APPROACH FOR IMPROVING TCP CONGESTION RESPONSE IN PACKET REORDERING SCENARIO USING DCR

Vivek Kumar Kushwaha[1], K Vinay Kumar[2]

[1]M.Tech Student, [2]Assoc. Professor, CSE, NITK Surathkal, (India)

ABSTRACT

Transmission control protocol (TCP) is widely used protocol for reliability and order of delivery of Data packets in the internet. Congestion control mechanism used by TCP makes this faster and smooth. Because of congested links and route changes in network, packet reordering generally occurs. Packet Reordering decreases the TCP performance by over-estimating packet loss and then unnecessarily follow congestion control mechanism for re-ordered packets. Due to packet reordering, it would be impossible for sender to find out that the packet has lost or re-ordered. So if there is a mechanism which unambiguously identifies that packet is re-ordered and not lost, then the throughput of TCP will be improved by eliminating unnecessarily performing congestion control mechanism for re-ordered packets. So in this paper, we are proposing a novel method in which we took lost packet information on router level and use this information for identifying which packet is lost and which packet is reordered. Based on this we perform congestion control only for lost packets not for reordered packets. Our experimental result shows significant improvement on this.

Keywords: Congestion Control, DCR, Packet Reordering, Smooth Congestion Response, TCP

I. INTRODUCTION

In internet TCP provide highly reliable transmission of data packet for most applications.TCPis popular for its congestion control mechanism but this mechanism becomes less effective when packet reordering happens in the network. During the data transmission packets are re-ordered mainly in a wireless network. This re-ordering happens due toseveral reasons: First a path offers fewer loads on link than another one, second route changes in networks, third due to side effect of the normal delivery process. When a packet is lost in the network TCP receiver will inform about this to sender either by sending three duplicate acknowledgements or sender will timeout. In response first sender will halve the congestion window and then, retransmit lost packet. Nowsender performs congestion avoidance phase and retransmit lost packets. This problem is challenging, because TCP reordering is common phenomenon now if packets are reordered in a network then they will be delayed to reach destination. Receiver sends duplicate ACKs for reordered packets and sender performs congestion control mechanism in response, which is needless because packets are delayed not lost. Due to this ambiguity performance of TCP becomes poor.TCP receiver has no mean to identify this ambiguity. This problem is important because several researchers have addressed this problem and gave various solution to achieve better TCPperformance [1], but most of them are reactive in nature means either they will delay acknowledgement (ACK) for some time or they will allow sender to enter in congestion avoidance phase to find out false

retransmission, so this prediction may be wrong for several cases, which degrades TCP throughput. So, if there is a mechanism which unambiguously identifies that packet is re-ordered and not lost, then the throughput of TCP is improved by eliminating unnecessary congestion control mechanism for re-ordered packets. Motivated by the fact, in proposed work, this ambiguity is resolved up to certain extent by keeping track of dropped packets at router. For each data flow this information will be stored in next incoming packet, when this packet will reach thereceiver, stored information will be checked against receiver's last received packet using our described algorithm to know whether the packet is lost or reordered. Input to our algorithm is lost packet information, current received packet sequence number and highest received packet sequence number at receiver side and output is next expecting packet at receiver either lost(by setting TCP optional bit 0) or reordered(setting bit1)at receiver. So receiver can use this information for sending congestion response to convey sender about congestion status. Ifa packet reorders in network then receiver can use delay congestion response (DCR) mechanism to convey sender about reordering event by sending 3+k duplicate ACKs(by setting reordering bit) and wait till the period before retransmissiontimeout occurs otherwise if a packet is lost in a network,then receiversends 3 duplicate acknowledgments and sender has to follow normal congestion control mechanism. Value of k evaluated through experiment. We have conducted experiment on network simulator 2 to validate our problem and solution.

II. LITERATURE SURVEY

Several researchers have paid attention to achieve robustness of TCP during packet reorderingPaxan [2] has conducted several experiments for proving packet reordering in internet. He observed that in absence of congestion event packet reordering can cause more than one duplicate ACK which seems as congestion event. To handle this, he proposed time delayed fast recovery TD-FR. If receiver does not immediately respond about out of order packets, sender never knows about re-ordering event for(10ms), but it has disadvantage if actual packet is lost then unnecessary receiver will wait for it which decrease efficiency of TCP.In Eifel algorithm [3] tries to detect loss event and reordering event separately. If sender receives 3 duplicate ACK, then it enters in fast recovery phase same as TCP Reno and when sender receives non duplicate ACK based on its contents, it either restores the previous window or continues fast recovery. Main contribution of Eifel algorithm is avoiding retransmission ambiguity. It maintains a time stamp t1 for first retransmitted packet and checks it with time stamp of received ACK (t2), if t2 < (t1) confirms that retransmission is suspicious thus, it became RFC standard.Zhang and Wang [4] shown that MANETS during route changes that are in triangles many packet may be lost. So, previously proposed congestion control algorithms interpret wrong and reduce sending rate. Thus, they proposed, if we identify the time when route change in network we can disable congestion control for this period, because packet reordering happens with this approach they achieved 50% throughout improvement over SACK version.TCP persistent reorderingBohacek [5] addressedthat duplicate packet can't be a loss event, so he turns to retransmission timeout as a true congestion indicator for different network. TCP-PR uses a timestamp t for every sent packets and it comprises with an estimated time stamp T and detect loss when t<T. This algorithm is more helpful in networks like mobile ad-hoc network, where ACKs are considered as suspicious. Selective ACK and Duplicate SACK algorithms used to reduce unnecessary retransmission of packets. In SACK(selective ACK)[6] when sender sends some data to receiver convey two information in ACK 1.Last ordered received packet 2.Number of unordered received packets figure clearly shows the difference DSACK is extension of

SACK which convey range of duplicate packets of it received. But in both algorithms, authors do not clearly show how to identify loss event and reordering event**.** In SACK we can identify a reordering event sender receives a SACK Followed by cumulative ACK provided no packets are retransmitted but if transmission happens then there is no mean to detect re-order event so RR-TCP [7] resolved it. This issue by using DSACK algorithm .If sender receives an ACK followed by DSACK then both transmission and retransmission is successful RR-TCP Provides a way to calculate reordering length. By this use can predict delay of packets thus set duplicate ACK thresholdaccordingly. But it fails to achieve benefits of robust loss detection when duplicate threshold is too high.

III. THE PROBLEM

3.1 Problem Justification

Packet Reordering decreases the TCP performance by over-estimating that packet has lost and then unnecessarily follow congestion control mechanism for re-ordered packets. TCP receiver identifies congestion in a network by packet drop and informs the sender about congestion by sending three duplicate acknowledgments for next expected packet in order or by timeout occurs. Due to packet reordering it would be impossible to find out that packet has lost or re-ordered [8]. So if there is a mechanism which unambiguous identification of whether the packet is re-ordered or lost, then the throughput of TCP is improved by eliminating unnecessarily performing congestion control mechanism for re-ordered packets.Smooth congestion response(SCR) is a mechanism to tell that how effectively receiver convey congestion event to sender when it occurs [9]. It mainlyhas three parameters to judge that: first no false Retransmission, second all bandwidth available on link should be utilized and third no unnecessary delay for sending response received packets.Packet reordering is common phenomenon in wireless networks. It has several negative impacts on TCP performance [10]. Thus due to following reasons it fails to achieve SCR characteristics:

3.1.1 False Retransmission

When a data packet is re-ordered in the network, then receiver continuously sends duplicate ACKs to sender so in response sender will retransmit that data segment that is re-ordered not lost. Thus unnecessarily consume network bandwidth and increase congestion.

3.1.2 Poor Transmission Rate

TCP assumes congestion if packets are dropped in routers. When sender receives 3 duplicate ACKs it will perform congestion avoidance mechanism to reduce the current window to half of its size. If multiple retransmission occurs in a similar window, then every time, its window size decreases by half this will result poor transmission rate. In next section we are going to describe different proposals to mitigating these problems.

3.2 Problem Validation

Here we compare various algorithms based on fact that how effectively we achieve smooth congestion response and which helps in improving TCP performance for packet reordering scenario. In literature there has not been much satisfactory work done to achieve parameters of smooth congestion response.Most of researchers have used DCR[11] for identifying reordering scenarios. When experiment is conducted to validate truthfulness of DCR, it has given average performance as compared with other versions of TCP. So our aim is to increase the robustness of DCR in case of packet reordering using proposed algorithm.To validate problem statement

experiment has been done on NS2. Here DCR is implemented via RED(Random Early Detection) and compare DCR+RED performance with other popular TCP variant with various conditions like at different FTP flows and at different bandwidth to verify parameter like packet drop rate, mean queue length and link utilization. Results are shown in graphs in next page. As the result DCR without reordering gave average performance.

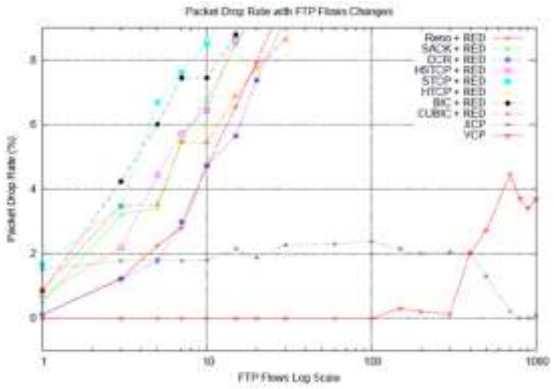

Figure 1.

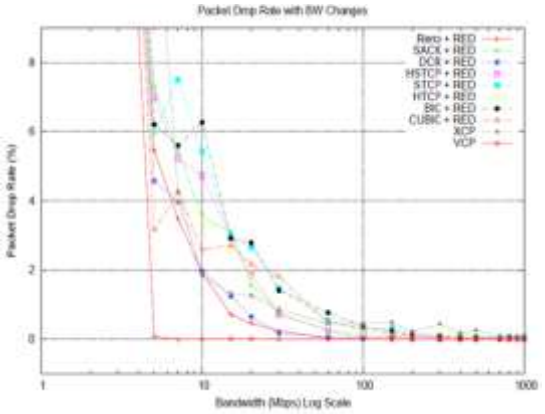

Figure 2.

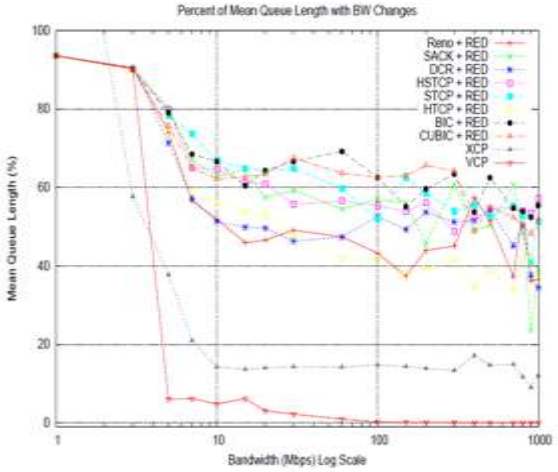

Figure 3

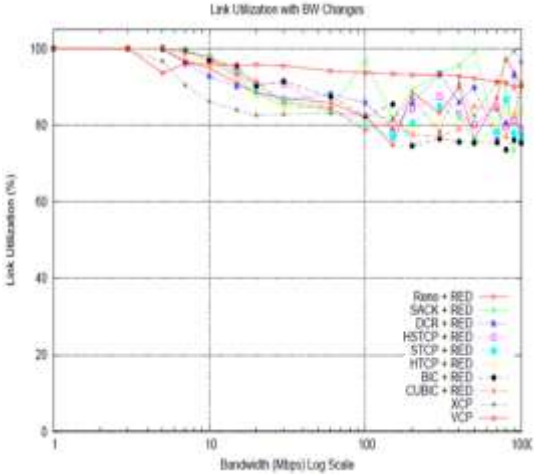

Figure 4

In figure 1 we measure between packet drop rate and different FTP flow changes for various TCP versions like TCP Reno,SACK,HSTCP,HTCP,BIC,CUBIC,XCP and VCP with TCP DCR. In figure it is clearly shown that for small numberof flows (i.e. 5) DCR has lesser packet drop rate than all otherversions except XCP and VCP.But for large number of flow (about 15) its packet drop rate increases exponentially and it is higher than Reno and CUBIC also. In second figure comparison is done between bandwidth and packet drop rate. We have seen packet drop rate of TCP Reno and VCP is lesser than DCR for increasing bandwidth. Figure 3 shows how mean queue length vary when bandwidth changes. For DCR mean queue length decreases slowly than XCP, VCP.Reno as bandwidth increases. Figure 4 shows link utilization against bandwidth changes. For DCR it also remains same as other variants.

IV. PROPOSED SOLUTION

If we clearly identify the pattern of reordering of packets or lost packets, then we can reduce problems which are the bottleneck for smooth congestion response. So here in the proposed mechanism by using a proactive approach we can resolve the ambiguity at the receiver end so receivers can correctly respond upon reordering or

packet lost.Here we are using packet lost information at router level to resolve ambiguity. We store least and maximum sequence number of lost packets at router in next incoming packet and send to receiver. When receiver receives this packet (current packet)then checks least sequence number stored in current incoming packet with last in order received packet sequence no. If there is gap between those entries it means packet is probably re-ordered otherwise packet is lost. So receiver can use this information for sending congestion response to convey sender about congestion status. If packet reorders in network then receiver can use delay congestion response mechanism to convey sender about reordering patterns by sending 3+k duplicate ACKs(by setting reordering bit) and wait him till this period before retransmission otherwise if a packet lost in a network then send 3 duplicate acknowledgments and sender has to follow normal congestion control mechanism. Here reordering is done by delaying the packets on certain intervals.

V. EXPERIMENTAL SETUP AND RESULTS

Experiments are performed on network simulator 2.First topology of network is created which is dumb bell topology in this topology a router is connected through various sender hosts called source router and sink router is connected through various sink hosts. Between source and sink router there is a lot of number of router.List of parameters using for implementation is mention below:

Network simulator Ns-2.35

Dumbbell topology(with AQM)

Link bandwidth =10 mbps

RTT duration = 80 ms

Value of k (duplicate threshold 3+k) = 0,1,2,3

Number of forward FTP flows = 5

Streaming packet size= 840 bytes

Streaming packet rate=640 kb/s

Queue size= 100 packets

5.1. Effect of Reordering

Reordering impact is critical on TCP performance [12]. Here we compare throughput achieved by our algorithm in reordering scenario with TCP SACK algorithm. Here reordering is performed bydelaying the packets on certain intervals.It is observed that for a TCP connection: When no reordering of packets is done, then average throughput was 805420 B/s and when reordering of packets is done then average throughput was 712000 B/s. Thus due to reordering throughput is decreased by 11.22%. If we increase delay,throughput is gradually decreased.

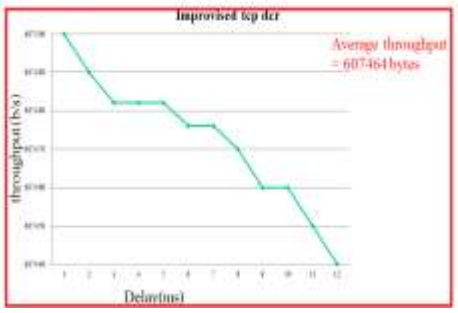

Figure 5

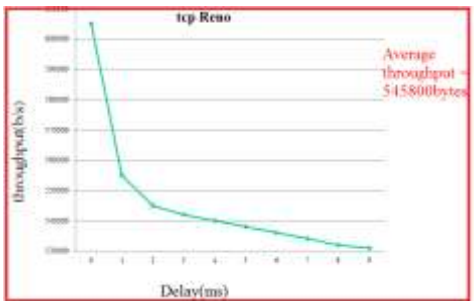

Figure 6

To inspect this experiments is performed for different values of k (for duplicate threshold) we have seen that for k=0 network behaves like ordinary network in case of reordering for k=1 network performance same with previous case.For k >= 2 network performance is much better than previous two case because false retransmission rate is decreased rapidly so throughput increases.Reordering is performed by 10 ms.In figure 5 it is shown that On transmitted data of 607500 bytes TCP DCR average throughput was 607464 bytes per second but for TCP Reno it was 545800 bytes per second (figure 6).Thus throughput decreasing rate is reduced in case of improved version of TCP DCR.

VI. CONCLUSION

Packet re-ordering is common behavior in wireless networks. But it has a negative impact on TCP robustness. To alleviate this effect, we have to carefully resolve ambiguity between the lost packet and reordered packet.So in this paper, we present a solution thatunambiguously informs sender whether the packet is lost or reorderedand it not only stops suspicious retransmission of packets, but also allows the sender to utilize full available bandwidth if packets are not lost.Information of lost packets on the router is used by the receiver to identify packet is reordered or not.This information is sent to sender in duplicateacknowledgement.Thus the sender need not to perform congestion control for reordered packets and he can wait for 3+k (here k=3) duplicate acknowledgements. Here we have used the mechanism of DCR only for reordering event. Experiment results show that our proposed algorithm with DCR is performed better than Reno. In future work we are trying to find reordering length of packets.Then we will compare reordering event with other TCP versions .

REFERENCES

[1] P. Reiher Afanasyev,N. Tilley and L.Kleinrock, Host to host congestion control for TCP,IEEE Communication Surveys and tutorials ,12(3),2010.

[2] V. Paxon, End to end internet packet dynamics, SIGCOMM,Computer Communication Review,vol. 27(4), 1997, 139-152.

[3] R. Ludwig and R. H. Katz, The Eifel algorithm: making TCP robust against spurious retransmissions, SIGCOMM Computer Communication Review, vol. 30(1), 2000,30–36.

[4] F.Wang,Y Zhang, Improving TCP performance over MOBILE ad hoc networks with out of delivery detection andresponse, Inpreceding of third ACM international symposium on mobile ad hoc networking and computing ,pages New York, NY,2002,217-225.

[5] S. Bohacek, J. Hespanha, J. Lee, C. Lim, and K. Obraczka, TCPPR: TCP for Persistent Packet Reordering,in Proc. International Conference on Distributed Computing Systems, 23, 2003 222–233.

[6] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, RFC2883-an Extension to the Selective Acknowledgement (SACK), RFC, 2000.

[7] M. Zhang, B. Karp, S. Floyd, and L. Peterson, RR-TCP: a reordering-robust TCP with DSACK,11th IEEE International Conference on Network Protocols, 2003, 95-106.

[8] Won suk kim Sang Hwa-Chung, An enhanced TCP scheme for distinguishing non-congestion losses from packet reordering over wireless mesh networks,13th International Conference on High Performance Computing and Communications,2011,440-447.

[9] M.Hassan and R.Jain,High performance TCP/IP networking:concepts,issues and solutions(Prentice Hall, 2003).

[10] Angela Doufexi, Amani Alheid,.Dritan Kaleshi,An analysis of impact of out of order recovery algorithms on MPTCP throughput,IEEE 28th international conference on advance information networking and applications, University of Bristol,UK,2014,156-163.

[11] Sumitha bhandarkar,a thesis on delayed congestion response protocols, Texas A.M. University, 2001.

[12] C. M. Arthur, A. Lehane, and D. Harle, Keeping order: Determining the effect of TCP packet reordering, in Proceeding of Third International Conference on Networking and Services (ICNS), June 2007.