Acknowledgment Time Delay Approach to Optimize TCP Performance in Hybrid Networked Systems

K. Lakshmi Nadh
Research Scholar
JNT University Kakinada
India

Y.K. Sundara Krishna
Dept. of Computer Science
Krishna University
Machilipatnam, India

K.Nageswara Rao
Principal
PSCMRC of Engg & Tech
Vijayawada, India

ABSTRACT
Transmission Control Protocol (TCP) is connection oriented transport protocol used on IP in wireless medium and it insists lossless data transmission in proper order. When TCP is used as a transmission protocol where physical layer is wireless medium, results high packet reordering due to bursty traffic and drastic variation in quality of service with respect to time. By sharing the same path for data and acknowledgement increases the traffic and collision, resulting in reduced throughput. In order to improve QoS this paper proposes a solution "Improved Delay the Duplicate Acknowledgement" (IDDA). This reduces traffic and spurious retransmissions, thereby improving TCP performance.

Keywords
Transmission Control Protocol (TCP), Fast Retransmission, Wireless Networks, packet reordering, Random loss, congestion control.

1. INTRODUCTION
The Transmission Control Protocol (TCP) is one of the core protocols of the Internet Protocol Suite [1] which is used extensively by many of the Internet's most popular applications [2], including Web (HTTP), file transfer (FTP), and e-mail (SMTP), Secure-Shell, and some streaming-media applications. TCP provides reliable stream delivery service that guarantees delivery of a data stream sent from one host to another. The order of the bytes transmitted from each host is identified with a unique sequence number (*seqno*) [3]. So that the data can be transferred reliably and in order, despite of any fragmentation, disordering, or packet loss that occurs during transmission. When a destination receives a data packet, it acknowledges the receipt of the packet by sending back an acknowledgement packet with the next expected data *seqno*. The round trip time (RTT) [4] is sampled and averaged to calculate the retransmission timeout (RTO) used by TCP to achieve reliable delivery. RTT timing has traditionally taken the form of starting a timer before a sending a packet and the packet is transmitted. After receiving the proper acknowledgement packet the timer will stop.

TCP primarily uses cumulative acknowledgment scheme [5], where the receiver sends an acknowledgement packet signifying that the receiver has received all data preceding the acknowledged *seqno*. Essentially, the first data byte in a packet is assigned a *seqno*, which is inserted in the *seqno* field, and the receiver sends an acknowledgment specifying the *seqno* of the next packet is expect to receive. For example, if the sender sends 4 bytes with a *seqno* of 100, 101, 102, & 103 assigned, then the receiver would send back an ACK of 104 since that is the next byte it expects to receive in the next packet. By sending an ACK of 104, the receiver is signaling that it received bytes 100, 101, 102, & 103 correctly. If, by some chance, the last two bytes were corrupted then an ACK value of 102 would be sent since 100 & 101 were received successfully.

TCP performance over networks subjected to congestion losses, which can be improved by enhancing following congestion control algorithms [2] [6] [7] [8]: slow start, congestion avoidance, fast retransmit, and fast recovery [9].

Slow start: A slow start suggests that the sender set the congestion window (*cwnd)* to 1 and then for each ACK received it increase the *cwnd* by 1. For the first RTT receiver sends 1 packet, in the second receiver sends 2 and in the third receiver sends 4. Thus sender increases exponentially until a packet loss, and decreases our sending rate by reducing congestion window to one and start over again.

Congestion avoidance: For congestion avoidance Tahoe uses 'Additive Increase Multiplicative Decrease'. A packet loss is taken as a sign of congestion and Tahoe saves the half of the current window as a threshold value. It then set *cwnd* to one and starts slow start until it reaches the threshold value. After that it increments linearly until it encounters a packet loss.

Fast retransmit: So Reno [9] suggests an algorithm called 'Fast Re-Transmit'. Whenever receiver receives 3 duplicate ACK's that indicates the segment was lost, so sender re-transmits the segment without waiting for timeout.

Fast recovery: Like Reno [9], New-Reno [10] also enters into fast-retransmit when it receives multiple duplicate packets. In the fast recovery phase which allows for multiple re-transmissions in new-Reno and it notes the maximums segment which is outstanding. The fast-recovery phase proceeds as in Reno, when a fresh ACK is received.

The rest of the paper is organized as follows: The Section 2 summarize the causes of the packet reordering and its consequences on TCP. Section 3 provides TCP delayed ACK mechanism and a survey of existing solutions and drawbacks. Section 4 discusses proposed solution IDDA. A performance study of the surveyed algorithms is presented in Section 5. Finally, conclusion will be provided in Section 6.

2. CAUSES AND CONSEQUENCES OF PACKET REORDERING
All Recent studies show that packet reordering plays a vital role in packet transmission and it is not a rare event in wireless networks [11] [12]. Packet reordering (PR) [13] is a phenomenon in which packets with higher sequence numbers are received earlier than those with smaller sequence numbers. In Fig1, The TCP sender consecutively sends eight

packets: P1, P2, P3, P4, P5, P6, P7 and P8. The same sequence arrives at the TCP receiver in the following order: P1, P3, P4, P5, P6, P2, P7 and P8, where P2 is reordered with reordering block size = 1 and reordering delay time = 4 packets. Upon receiving the three dupacks of packet P2, the sender triggers fast retransmission and retransmits the packet by reducing the size of cwnd groundlessly.

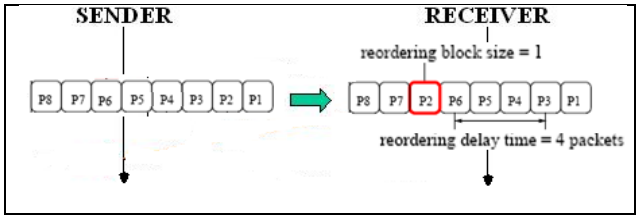

Fig1: Packet reordering

2.1 Causes of PR

In wireless networks, packet reordering is due to [13] [14] multipath routing, Parallelism in Packet router, Load splitting, route fluttering, link layer retransmissions, router forwarding lulls etc.

Multi-Path Routing

When packets from the same flow take two or more paths through a network, it becomes very likely that packets will arrive out of order. The TCP receiver will reorder the packets before presenting the data to the application. The TCP sender fast retransmit mechanism will often mistake packet reordering for packet loss and will continue to reduce its offered load to unacceptably low levels.

Parallelism in Packet router

This is a promising approach to build a high-speed and inexpensive packet router. Thus, packet reordering is more likely to occur in high-speed networks. Successive packets that arrive at a router, even on the same link, may be forwarded and/or switched simultaneously by independent hardware. This simple parallel approach ignores ordering between packets processed simultaneously, and introduces packet reordering.

Load splitting

Load splitting among multiple links. The analysis in a recent study [15] shows that multiple links with slightly different link delays may introduce significant packet reordering.

Route Fluttering

Routing fluttering is a network phenomenon in which the forwarding path to a certain destination oscillates among a set of available routes to that destination. This result from route instability due to shaky links, and heavy loads or bursty traffic can present topological changes in the wireless environment. Similar to packet-level multipath routing, route fluttering causes packets to be forwarded on different paths and arrive at a destination can introduce chaotic in packets.

Link-Layer Retransmissions

Link-layer retransmission mechanisms [16] have been proposed to efficiently recover transmission losses due to high channel error rates in wireless networks. Such retransmitted packets are sent only after the losses are detected. These packets may then be interspersed with other packets belonging to the same traffic flow.

Router Forwarding Lulls

Some routers can halt its forwarding activity for buffered packets when it processes a routing update. These buffered packets are interspersed with new arrivals, thus causing packet reordering [17].

2.2 Consequence of PR

The effects of packet reordering are determined by how the TCP sender responds to these various tasks [18] [19]: Spurious fast retransmit [6], Keeping Congestion Window Unnecessarily Small, Spurious timeout and ACK compression.

Spurious fast retransmit

Most TCP's implement an algorithm known as fast retransmit [6] used to recover quickly from occasional packet loss. The fast retransmit algorithm is triggered by a series of duplicate acks. If the TCP sender gets three duplicate acks, it assumes that the data immediately after the byte being acked has been lost, and retransmits that data. The idea is that the duplicate acks indicate a packet has been lost, and by quickly retransmitting the data. If the duplicate acks cause due to reordering, then the fast retransmission is unnecessary and wastes bandwidth.

Keeping Congestion Window Unnecessarily Small

Fast recovery is generated with fast retransmit. A spurious fast retransmission would cause extra workload to the network and make congestion window half. Therefore, the congestion window becomes small relative to the available bandwidth of its transmission path with packet reordering.

Spurious timeout

Reordering can falsely inflate the RTT estimate when no unnecessary retransmissions are sent, which can potentially hurt performance in that TCP would have to wait longer before sending a legitimate retransmission. In the case when a segment is retransmitted needlessly because of reordering, the corresponding RTT sample [20] must be marked as invalid.

ACK compression and traffic

ACK compression is another issue that possibly arises with wireless link delays. Packet reordering causes not only data segments, but also ACKs to arrive at a destination out of order. The former phenomenon is called forward-path reordering, while the second is known as reverse-path reordering [21]. ACK-clocking or self-clocking refers to the property that the receiver can generate ACKs no faster than data segments can get through the network [22]. For forward-path reordering, an ACK for several new segments, which follows a number of duplicate ACKs, can in turn allow a source to inject several pending segments into the networks. Even when there is no data segment being reordered, disordered ACKs lead to a source transmitting several segments together rather than one or two segments per ACK. This causes loss of its ACK clocking and bursty traffic, which may lead to transient network congestion and congestion collapse from undelivered packets [23].

3. DELAYED DUPLICATE ACKNOWLEDGEMENT SOLUTIONS FOR PACKET REORDERING

3.1 The TCP delayed ACK mechanism

With every segment that TCP sends, the receiver issues an ACK to acknowledge the receipt of the data. Instead of sending an ACK for every segment, the delayed ACK [24] [25] [40] can be enabled, which gives the TCP receiver have to wait some time before sending an acknowledgement. In the implementation of TCP protocol, delayed ACK is used to improve the TCP transfer performance [26]. A host that is receiving a flow of TCP data packets can increase efficiency by sending few than one ACK packet per data packet received. Because a packet loss event will cause the change of the behavior pattern of the end points, investigation of the ration between the bidirectional packet numbers give clues of

network packet loss. The receiver contains a timer bound variable interval that gives the number of seconds to wait for the second in-order packet. The time-out of this timer causes immediate ACK generation of the first packet. In addition, out-of-order packets cause immediate ACK generation.

3.2 Existing Solutions

A number of TCP solutions have been proposed to solve the problem of packet reordering related acknowledgment delay are: DelAck TCP-ADA and TCP-ADW.

3.2.1 DelAck

Jimenez and Altman have promoted the use of delayed ACK techniques (DelAck) [27] [30], to improve the performance of TCP in multihop wireless ad-hoc networks. DelAck is a receiver-side solution, which reduce channel competitions among data segments and ACKs and also reduce performance diminution due to packet reordering. In transmission path, the forward and reverse traffic between adjacent hosts can share and compete for same channel. In order to reduce channel competitions, the DelAck is delay the acknowledgement for the coming data packets. In this approach a receiver generate an ACK for every d data segments. An ACK is also generated whenever the first unacknowledged data segment has been received for a certain time period, say 0.1 s. The value of d can be configured so that d increases with the segment seqno.
Drawbacks:

- The value of d is orthogonal to the packet seqno and which depends on the size of the congestion window and available bandwidth.
- The bursts of TCP segments may be put into the network every time a delayed ACK is received by a source. This can start to transient network congestion and congestion collapse due to undelivered packets.

3.2.2. TCP-ADA

Kankipati and Singh developed TCP with "adaptive delayed acknowledgment" (TCP-ADA) [28] [30]. It is a receiver-side solution to reduce intra-flow channel contention in mobile ad hoc networks. The approach of TCP-ADA is similar to that of DelAck [27, as long as TCP-ADA postpones acknowledgment for a time period. After receiving the data segment, TCP-ADA updates Δ, an exponentially weighted moving average of the inter arrival time between two consecutive packets. A destination will defer sending an ACK of the segment for a time period of $\beta\Delta$. The deferment period is restarted every time a data segment arrives before the postponement timer expires. An ACK is sent to a source if the total deferment period reaches a certain threshold.
Drawbacks:

- The source has to be idle for about one RTT to receive an ACK before it can send new segments to the destination then the loss of ACK-clocking to the network.
- If one ACK for a full congestion window of data. The loss of that ACK leads to expiration of the retransmission timer and the initiation of the slow start phase to restarts cwnd.

3.2.3. TCP-ADW

Ammar and Othman devised the adaptive delayed ACK algorithm TCPADW [29] [30], is a receiver-side solution which addresses ACK delay window. The main goal was to have a better throughput by lowering the number of ACK to the minimal number, which guarantees the TCP reliability and reduce channel competition. In this scheme, unless the sender's retransmission timer expires, the receiver always increases the delay window based on the increase in the transmission rate (cwnd size), except at session startup. During the startup, the receiver sets the delay window to 1 and increases it based on the transmission rate. When the packet loss occurs, the receiver decreases its delay window to a certain value based on the hop count. In case of short path, the receiver decreases the delay window to half. For the long path, the receiver will decrease its delay window to the value two. Out-of-order packets cause ACK generation immediately in order to inform the sender of the packet loss in a timely manner.
Drawbacks:

- In this scheme when packet reordering occurs, it does not avoid spurious retransmissions.
- It does not solve packet reordering by delaying the ACK, it is for avoiding the competition and collision.

4. PROPOSED SOLUTION

TCP-IDDA [38] is a new approach, which is capable of decreasing the number of ACK's to improve TCP performance and avoid unnecessary retransmissions.

4.1 TCP-IDDA

Generating acknowledgement for each data packet also reduces TCP throughput over wireless networks than in wired networks. In wireless networks, acknowledgement packets share the path with data packets. This creates the competition and collision between ACK and DATA packets, resulting in reduced TCP throughput. Decreasing the number of ACKs enhances the performance of TCP as it reduces the competition and collision with DATA packets. This paper proposes a receiver-side solution IDDA is extension of Delayed Duplicate Acknowledgements [31] [32], which Delay the duplicate acknowledgement for a certain time period. The proposed solution is to decrease the number of acks to improve TCP performance. TCP throughput is improved when one ack acknowledges the full congestion window or out-of-order packets.
In the network environment of TCP must estimate the following settings and with mathematical calculation. The information has retrieved with PING for google.co.in with address (173.194.117.111) 56(84) bytes of data. The data is displayed in the following table1.

Table1: data with seqnumber and time

Seq	Time (ms)	Seq	Time (ms)	Seq	Time (ms)
1	316	11	285	21	285
2	316	12	300	22	302
3	303	13	297	23	269
4	309	14	314	24	260
5	322	15	319	25	241
6	316	16	311	26	256
7	313	17	277	27	251
8	294	18	281	28	270
9	306	19	265		
10	296	20	Reorder		

The RTT [4] was originally estimated in TCP by:

$RTT_{est} = (\alpha * RTT_{Old}) + ((1 - \alpha) * RTT_{New})$ (1)

Where α is constant weighting factor ($0 \leq \alpha < 1$). Choosing a value α close to 1 makes the weighted average immune to changes that last a short time (e.g., a single segment that encounters long delay). Choosing a value for α close to 0 makes the weighted average respond to changes in delay very quickly. For simulation maximum RTT sample is taking into consideration.

For our calculation α=0.5.

$RTT_{est} = (0.5 * 316) + (0.5 * 303)$

=309.5ms

Compute Retransmission Time Out for the network environment

$RTO = \beta * RTT_{est}$ where $\beta > 1$ (2)

A low value of ' β ' will ensure quick detection of a packet loss. Any small delay will however cause unnecessary retransmission. A typical value of ' β ' is kept at 2.

For our calculation β =1.5.

RTO =1.5*309.5

=464.25ms

In order to observe congestion, the traditionally TCP implementations have increased cwnd by precisely SMSS bytes upon receipt of an ACK covering new data. Thus TCP implementations increase cwnd,

cwnd += min (N, SMSS) (3)

Where SMSS is (Sender Maximum Segment Size), which is the size of the largest segment that the sender can transmit. Another common formula that a TCP MAY use to update cwnd during congestion avoidance is given in equation (4):

cwnd += SMSS*SMSS/cwnd (4)

TCP sender sequentially sends packets into the network by representing its receiver address.

$$\lim_{i \to w} p(i) \quad (5)$$

Where $p(i)$ is number of packets with window size w ($1 \leq i \leq w$).

The packets are entered into the network in the order of p(i),p(i+1),p(i+2)…….p(i+w)

When the receiver gets the unsequenced packet from sender, it immediately sends negative ACK of previous sequence packet. That is, the receiver gets the packet order in the form of p(i),p(i+1),p(i+2),p(i+3),p(i+4),p(i+5)….p(i+27).

There the packet p (20) get reordered, it immediately responds to the first two consecutive out-of-order packets by sending negative dupack (ACK19) [33] immediately and calculates delay value. However, dupacks for further consecutive out-of-order packets are delayed for certain time d. If the next in-sequence packet is received within the interval, then the delayed dupacks are not sent. Otherwise, after the time d interval, the receiver sends ACK for every d.

In order to calculate delay value, When the packets get reordered, at the receiver end, it measures the gap [34] [39] between two adjacent data packets. At this time, we supposed

$\Delta\tau^{rp,pp}$ Stood for Gap value, which is difference between the arrival time of recently received packet and previous packet.

$$\Delta\tau^{rp,pp} = \tau^{rp} - \tau^{pp} \quad (6)$$

Where τ^{rp} , is recent packet time τ^{pp} is previous packet time at receiver side.

$\Delta\tau^{rp,pp}$=285-265

=20ms

Calculate one way round trip time from eq(1) RTT=309.5ms

$t_o = RTT/2$ (7)

=154.75ms

Then add the gap value of eq (6) to t_o

$T = t_o + \Delta\tau^{rp,pp}$ (8)

=174.75ms

Then measure T^1 by adding some time variance with eq (8)

$T^1 = T + \varepsilon T$ (9)

Where ε=0.25

T^1 =218.43ms

Send previously sent acknowledgement for every delay (d) value is obtained with eq (9).

$d = \gamma T^1$ (10)

Where γ=0.5

d=109.21ms

If the reordered packet p (20) is arrives with in delay interval, the receiver should not send the third dupack. Otherwise it will send the third dupack.

At the sender side it maintains acknowledgement threshold limit value.

$$\lim_{j \to 3} ACK(j) \quad (11)$$

When ACK value reaches to threshold limit, immediately sender concludes the packet is loss or dropout but not reordered. Then sender retransmits the lost packet.

Table2 shows the comparison of different mechanisms in terms of fast retransmission, with our proposed mechanism, the reordered packet is arrived at the receiver side before sending the third dupack.

Table2: Different mechanisms in terms of fast retransmission

Mechanism	Fast Retransmission (ms)
RTO	464.25
Duplicate ACK's (Traditional)	314
TCP-IDDA	400.25

5. PERFORMANCE EVALUATION

Our approach concentrates mainly on keeping the TCP throughput by delay the ACK to cwnd size. Simulation experiments carried out to investigate and evaluate our algorithm on hybrid networks. The main goal is to achieve maximal TCP throughput and avoiding the unnecessary fast retransmissions and reduce the needless reduction of congestion window size by delaying ACKs. The performance of IDDA is compared in terms of throughput, ACK's received and unnecessary retransmissions, using simulations conducted in ns-2 simulator [35] (version 2.35), to the following other algorithms: DelAck, TCP-ADA and TCP-ADW with metrics Throughput[36], Unnecessary retransmission rate[37] and Rate reorder/ack[26].

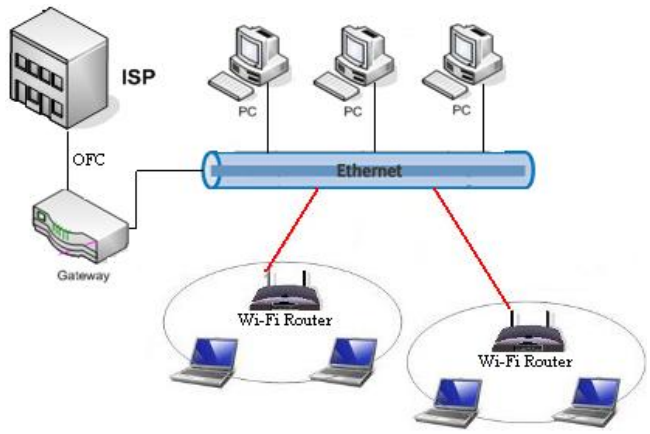

Fig2: Hybrid topology

5.1 Simulation Environment and Results

In this section, we describe our simulation environment and parameter settings used for evaluating the performance of IDDA over hybrid topology which uses Internet Service Provider (ISP), Gateway, Ethernet LAN and Wi-Fi as shown in Fig2. The simulated network is having 30 wireless nodes which have valid transmission range of 200m apart. In the Linux to know the link utility by using Traceroute command it can identify total number of hops from source to destination with time. In our simulation case it is having maximum of 30 hops that was identified with Traceroute i.e. from INDIA to US (Ex. amazon.com). During simulations, the data packets are continuously transmitted up to the end of simulation and the source of all TCP flows originated at the first node. The data transmission rate is 2Mbps, otherwise stated. Each data point represents an averaged result of 5 simulation runs with different random seeds. FTP is the traffic source we used all of our simulations. The packet size is 1000 bytes and the initial window limit is set to 8 packets. The size of an acknowledgment packet is same as the size of data packet.

Packet reordering is simulated by modifying the error model object of ns-2 such that randomly selected packets can be delayed for a random amount of time. This allows us the flexibility to choose the percentage of the packets to be delayed, the distribution for choosing the packets randomly as well as the distribution for the delays. The TCP-IDDA agent is implemented by modifying the tcp-sack1 implementation of TCP-SACK agent in ns-2.

The TCP Sink/Sack1 agent is used for the receivers. FTP sources start sending data at time 0 and are staggered to avoid synchronization. All simulations [37] are run for 150 seconds. The receiver advertises a large window such that the sending rate is not limited by the receiver dynamics.

Throughput Evaluation

Fig3 presents the result of typical variation of TCP throughput under varying bandwidths ranges from 10 to 30 Mbps. The loss rate and reorder rate set to 5 and 3%, respectively. From the graph, we observe that when bandwidth increases, the throughput also increases, except IDDA the throughput of all other TCP's fluctuates lightly. Compared to other delay mechanisms, IDDA can utilize the bandwidth efficiently. When bandwidth greater than 15 Mbps, IDDA begins to increase the throughput.

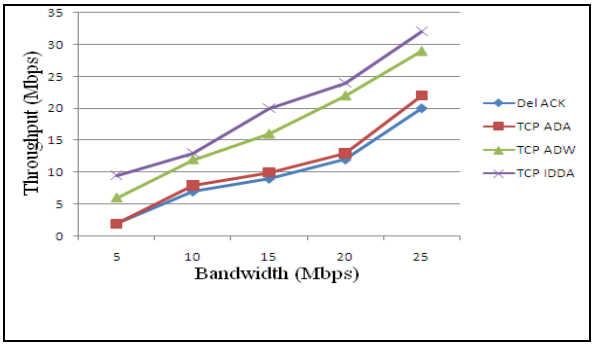

Fig3: TCP throughput according to various bandwidths

Simulation results in Fig4 shows the throughput gain of IDDA in presence varying reorder rate ranges from 1 to 5%. When reorder rate increases, the performance of IDDA becomes better than DelAck, TCP-ADA and TCP-ADW. As a result, DelAck, TCP-ADA and TCP-ADW frequently reduce the size of cwnd and send spurious retransmissions and thereby decrease the throughput performance. When the rate of reorder increases, the spurious retransmissions of all TCP's increases. The delay mechanisms such as DelAck, TCP-ADA and TCP-ADW cannot perform well according to the different reorder rates because these solutions cannot properly detect the reorder packets and results in the increases of spurious retransmissions. On the other hand, IDDA can detect reorder packets compared to other TCP mechanisms.

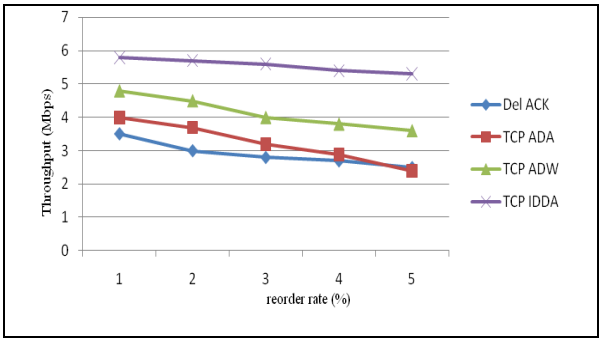

Fig4: TCP throughput according to various reorder rates.

Unnecessary retransmission Evaluation

The Fig5 analyzes the percentage of spurious retransmissions of various algorithms under varying reorder rate ranges from 1 to 5%. The spurious retransmissions rate, which is defined as the ratio of spurious retransmissions to the total number of packets transmitted.

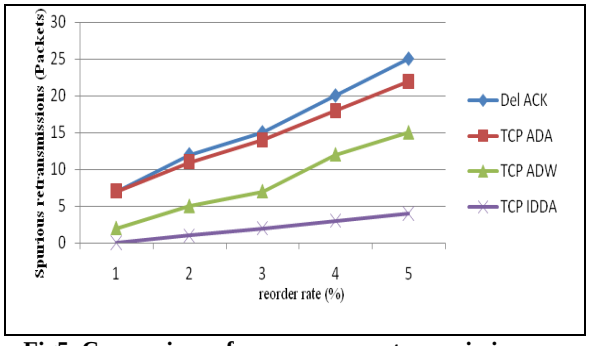

Fig5: Comparison of unnecessary retransmissions vs. reorder rate.

Acknowledgments received Evaluation

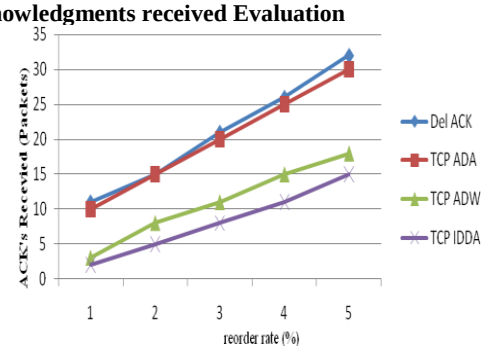

Fig6: ACK's received vs. reorder rate.

In the Fig6, it analyzes the percentage of acknowledgements received of various algorithms under varying reorder rate ranges from 1 to 5%. The acknowledgements received, which is defined as the ratio of acknowledgements received to the total number of packets reordered. When the rate of reorder increases, the acknowledgements received of all TCP's increases. However, IDDA has less number of ACK's received compared to other TCP mechanisms due to the ability of detecting the reorder packets. Compare to DelAck, TCP-ADA and TCP-ADW the IDDA has less number of acknowledgements received because of their capability to find the reorder packets.

Table3 a summary of the TCP enhancements listed in this paper and their functionality is presented. The comparison is based on features: the solution deals with packet reordering, reducing ACK overhead, spurious retransmission and usage of bandwidth.

Table3: Comparison of various TCP packet reordering Receiver side solutions

Strategy	Dealing with reordering	Reducing ACK overhead	Dealing with spurious retransmission	Usage of bandwidth
DelAck	No	Low	No	Low
ADA	No	Low	No	Low
ADW	Yes	Medium	No	Medium
IDDA	Yes	High	Yes	High

6. CONCLUSION

In this paper, we have presented the effect of number of acknowledgements generated on TCP throughput in wireless networks. The TCP-IDDA is a receiver side solution, which tries to minimize the competitions between data and ACK packets by reducing the number of ACK packets. Further, unnecessary retransmissions and needless reduction of congestion window size are significantly reduced by delaying the duplicate acknowledgement for a time when packet reordering occurs and it also improves bandwidth availability.

We also proposed some future research direction, including the need of a mechanism to resolve the other non-congestion losses including random loss with different topologies.

7. REFERENCES

[1] J. Postel, "Transmission Control Protocol," Request for Comments, RFC 793, Protocol Specification, DARPA Internet Program, Sept. 1981.

[2] Joerg Widmer, Robert Denda, and Martin Mauve, A Survey on TCP-Friendly Congestion Control IEEE Network • May/June 2001.

[3] A. McKenzie A Problem with the TCP Big Window Option, Network Working Group RFC 1110, August 1989.

[4] Improving Round-Trip Time Estimates in Reliable Transport Protocols by PHIL KARN and CRAIG PARTRIDGE in ACM Transactions on Computer Systems, vol 9, No4, and November1991.

[5] M. Mathis,J. Mahdavi,S. Floyd,A. Romanow , TCP Selective Acknowledgement Options, Standards Track RFC 2018,October 1996.

[6] W. Stevens, TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms RFC: 2001 NOAO,January 1997.

[7] K. Fall, and S. Floyd, "Simulation–Based Comparison of Tahoe, Reno and SACK TCP", Computer Communications Review ACM-SIGCOMM, Vol. 26, No. 3, July 1996.

[8] Bogdan Moraru, Flavius Copaciu, Gabriel Lazar and Virgil Dobrota, Practical Analysis of TCP Implementations: Tahoe, Reno, NewReno, RATES/ERASMUS 2001-2002.

[9] M. Allman, V. Paxson, and W. Stevens, TCP Congestion Control, IETF RFC 2581, Apr. 1999.

[10] S.Floyd and T.Henderson, "The NewReno modification to TCP's fast recovery algorithm," IETF RFC 2582, April 1999.

[11] Chunlei Liu, Fangyang Shen and Min-Te Sun, "A Unified TCP Enhancement for Wireless Mesh networks", Parellel Processing Workshops, 2007. ICPPW 2007. International Conference on vol,no..pp.71,10-14 sep 2007.

[12] Xiaoyuan Guo, Jiangchuan Liu "Path diversfied retransmission for TCP over wireless mesh networks" Quality of service (IWQoS),2010 18th International workshop on vol,no..pp.1-9,16-18 june 2010.

[13] KC Leung, V Li, D Yang, An overview of packet reordering in transmission control protocol (TCP): problems, solutions, and challenges. Parallel Distrib Syst IEEE Trans. 18(4), 522–535 (2007).

[14]Jie Feng, Zhipeng Ouyang, Lisong Xu *, Byrav Ramamurthy, Packet reordering in high-speed networks and its impact on high-speed TCP variants,Computer Communications ,2008.

[15] L. Zhang, Z. Zhao, Y. Shu, L. Wang, and O. W. Yang, "Load balancing of multipath source routing in ad hoc networks,", Communications, 2002.

[16] F. Hu and N.K. Sharma, "Enhancing Wireless Internet Performance," IEEE Comm. Surveys and Tutorials, vol. 4, no. 1, pp. 2-15, Dec. 2002.

[17] V. Paxson, "End-to-End Internet Packet Dynamics," IEEE/ACM Trans. Networking, vol. 7, no. 3, pp. 277-292, June 1999.

[18]R. Ludwig and R. H. Katz, "The Eifel algorithm: making TCP robust against spurious retransmission," ACM Computer Communications Review, vol. 30, no. 1, pp. 30–36, January 2000.

[19] Jon C. R. Bennett, Craig Partridge, and Nicholas Shectman, Packet Reordering is Not Pathological Network Behavior IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 7, NO. 6, DECEMBER 1999.

[20] V. Jacobson, R. T. Braden, and D. Borman, TCP Extensions for High Performance, rfc-1323, May 1992.

[21] J. Bennett, C. Partridge, and N. Shectman, "Packet Reordering is Not Pathological Network

Behavior," IEEE/ACM Trans. Networking, vol. 7, no. 6, pp. 789-798, Dec. 1999.

[22] V. Jacobson, "Congestion Avoidance and Control," ACM SIGCOMM Computer Comm. Rev., vol. 18, no. 4, pp. 314-329, Aug. 1988.

[23] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," IEEE/ACM Trans. Networking, vol. 7, no. 4, pp. 458-472, Aug. 1999.

[24] TCP PACKET CONTROL FOR WIRELESS NETWORKS,Wan Gang Zeng,THESIS SUBMITTED IN SIMON FRASER UNIVERSITY,August 24, 2006.

[25] R Braden, Requirements for internet hosts–communication layers, RFC 1122, IETF Network Working Group, October 1989.

[26] Hua Wu and Jian Gong, Packet Loss Estimation of TCP Flows Based on the Delayed ACK Mechanism, C.S.Hong et al.(Eds):APNOMS 2009,LNCS 5787,PP.540-543,2009.

[27] E. Altman and T. Jimenez, "Novel Delayed ACK Techniques for Improving TCP Performance in Multihop Wireless Networks," Lecture Notes in Computer Science, vol. 2775, Sept. 2003.

[28] A. K. Singh and K. Kankipati, "TCP-ADA: TCP with Adaptive Delayed Acknowledgement for Mobile Ad Hoc Networks, "Proc. IEEE WCNC 2004, vol. 3, Atlanta, GA, USA, 21–25 Mar. 2004, pp. 1685–90.

[29] Ammar Mohammed AI-Jubari and Mohamed Othman "A New Delayed ACK Strategy for TCP in Multi-hop Wireless Networks" 978-1-4244-6716-7/10/$26.00 © 2010 IEEE.

[30] Ammar Mohammed Al-Jubari, Mohamed Othman, Borhanuddin Mohd Ali and Nor Asilah Wati Abdul Hamid,"TCP performance in multi-hop wireless ad hoc networks: challenges and solution", EURASIP Journal on Wireless Communications and Networking December 2011.

[31] Nitin Vaidya, Miten Mehta, Charles Perkins, and Gabriel Montenegro, "Delayed Duplicate Acknowledgements: A TCP-Unaware Approach to Improve Performance of TCP over Wireless".

[32] Miten N.Mehta and Nitin H.Vaidya, "Delayed Duplicate Acknowledgements: A proposal to Improve Performance of TCP on Wireless Links", Feb, 24, 1998.

[33] A Roach, A negative acknowledgement mechanism for signaling compression, RFC 4077, IETF Network Working Group (May 2005)

[34]Guy Riddle "Method for Measuring Network Delay using Gap time" Patent No: US &, 012,900 B1, Mar, 14, 2006.

[35] "The network simulator – ns2," Available at http://www.isi.edu/nsnam/ns/.

[36] Prasanthi.S., Sang-Hwa Chung and Won-Suk Kim, An Enhanced TCP Scheme for Distinguishing Non-congestion Losses from Packet Reordering over Wireless Mesh Networks 2011 IEEE.

[37] Sreekumari and Chung "TCP NCE: A unified solution for non-congestion events to improve the performance of TCP over wireless networks" EURASIP Journal on Wireless Communications and Networking 2011, http://jwcn.eurasipjournals.com/content/2011/1/23.

[38] K. LakshmiNadh, Y.K.Sundara Krishna and K.Nageswara Rao "Improving TCP performance with delayed acknowledgments over wireless networks: a receiver side solution", Conference on Advances in Recent Technologies in Communication and Computing (ARTCom 2013), 20-21 Sept. 2013,DOI: 10.1049/cp.2013.2231, ISBN: 978-1-84919-842-4,Location: Bangalore, India.

[39] Quanjie Qiu, Zhinhuo Li ans Zhongfu Wu "An Algorithm for Avilable Bandwidth Estimation of IPv6 Network" DOI:10.1007/978-3-642-17313-4_41 In proceeding of: Advanced Data Mining and Applications-6th International Conference, ADMA 2010, Chongqing,China,November 19-21,2010,Preceediings,PartII Source:DBLP.

[40] Sumitha Bhandarkar and A. L. Narasimha Reddy Quanjie Qiu, Zhinhuo Li ans Zhongfu Wu "TCP-DCR: Making TCP Robust to Non-congestion Events" Networking 2004**,** Lecture Notes in Computer Science Volume 3042, 2004, pp 712-724**.**