National Conference on Emerging Trends in Computer, Electrical & Electronics (ETCEE-2015)
International Journal of Advance Engineering and Research Development (IJAERD)
e-ISSN: 2348 - 4470 , print-ISSN:2348-6406,Impact Factor:3.134

A Comprehensive Study of Layered Approaches for Improving TCP Performance in Wireless Networks

Hardik K. Molia[1] , Prof. Rashmi Agrawal[2]

[1,2] *Department of Computer Engineering*
Atmiya Institute of Technology & Science,
Rajkot- 360005,
Gujarat, India

[1]hardik.molia@gmail.com
[2]rashmi.agrawal@aits.edu.in

Abstract- **TCP- Transmission Control Protocol provides connection oriented, reliable, process to process communication. TCP was initially implemented with wired networks where the main cause of packet drops is congestion rather than channel issues like interference, noise, attenuation etc. when TCP is implemented with wireless networks, a significant amount of performance degradation can be noticed specially because of TCP's misinterpretation between congestion losses and any other losses. Various schemes have been proposed to differentiate various losses to improve overall performance. These approaches are broadly classified into two categories, cross layered approaches and layered approaches. In cross layered approaches, lower layer provides feedback – decision making information to the TCP to act accordingly. In layered approaches, TCP itself decides and acts without any feedback – decision making information from the lower layers. This paper focuses on layered approaches to improve TCP's performance. Approaches are classified into three categories for route failures, channel losses and channel contention issues.**

Keywords **– TCP, MANET, Congestion, Route Failure, Channel Loss, Contention, Layered Schemes, ACK Thinning**

I. INTRODUCTION

TCP performs well in wired networks where the reason of the packet loss is the network congestion in most of the cases. The standard TCP has no capability to distinguish congestion loss and non congestion and act accordingly. TCP's such misinterpretation of non congestion loss as congestion loss is the reason behind the performance degradation in wireless mobile adhoc networks – MANETs [1].

MANETs allow nodes to be mobile during the communication. Because of the node mobility, some of the nodes may move out of the transmission ranges of neighboring nodes with which they are sharing a route – causing a route failure. Route failure may split the network into two or more network partitions. When a route failure occurs, all the packets at the intermediate nodes are dropped [1]. Wireless nodes have limited ranges of transmission. Transmission range is even lesser than the carrier sensing range. There is a possibility that a wireless node may not detect any other ongoing communication because it is out of its sensing range. Simultaneous transmissions may collide and introduce loss of packets [1]. Wireless communication affects from various transmission impairments like attenuation, interference, noise which may corrupt or loss signals. This loss is known as random wireless loss or channel loss. The probability of such losses - measured as Bit Error Rate (BER) is much higher in MANETs as compared to wired networks. The next sections show various proposed schemes based on the specific loss they address [1].

II. APPROACHES FOR ROUTE FAILURE

These proposals are non feedback based layered approaches which focus on differentiation between route failure losses and other losses like congestion and channel losses.

A. Fixed RTO: -

Standard TCP is based on exponential back off algorithm for RTO- Retransmission Time Out. On every successful acknowledgement, RTO is recalculated as explained in 2.8. As per the back off algorithm, on every occurrence of RTO times out, TCP doubles RTO value and starts timer again. The upper limit for RTO depends upon specific TCP implementation. In fixed RTO based scheme, RTO increases exponentially until two consecutive time outs occur. On expiration of two consecutive RTOs without receiving any acknowledgement in between, TCP assumes possibility of route failure and doesn't increase RTO for second time. Fixed RTO has shown improvements over AODV – Adhoc On demand Distance Vector and DSR – Dynamic Source Routing algorithms [2].

B. TCP DOOR: -

DOOR stands for Detection of Out-of-Order and Response. DOOR is based on observing the pattern of delivery of segments.

1. Out-of-Order Delivery: - Out-of-Sequence delivery refers to the situation when a retransmitted segment reaches late. It causes violation of strict ordered delivery because some next segments are received successfully in 1st attempt. OOO - Out-of-Order delivery refers to the situation when a segment reaches late because of following a different path. It causes violation of strict ordered delivery because next segments may follow faster / shorter paths and reach earlier. TCP DOOR assumes a route failure when out-of-order delivery occurs. OOO delivery may trigger duplicate acknowledgements which may cause fast retransmission by the sender. DOOR detects such event and avoids unnecessary congestion control [3].

2. Detection of OOO of ACKs by Sender: - Every TCP ACK carries a sequence number to indicate highest sequence number of bytes received consecutively so far. ACKs are not retransmitted (Duplicate ACKs are not retransmissions). So every successive ACK carries a higher sequence number than the one previous ACK carried. This inherent non decreasing property justifies that a receiver never sends a lower numbered ACK than any of the previously sent ACKs. Sender stores the sequence number of the most recently received ACK. On receiving an ACK, if its sequence number is lower than the stored highest sequence number so far, OOO delivery is considered [3].

Detection of OOO delivery across the same set of duplicate ACKs is tricky as they all carry the same sequence number. DOOR adds one byte of TCP Option in TCP header called ADSN – Acknowledgement Duplicate Sequence Number with every duplicate ACK. TCP receiver increments ADSN by 1 with every duplicate ACK of same instance. Sender can detect OOO delivery across the duplicate ACKs by comparing new ADSN with the highest ADSN received so far [3].

3. Detection of OOO of Data by Receiver: - There is no inherent ordered property involved with the data segments delivery. A segment with the lower sequence number may arrive later because of the out-of-sequence delivery. DOOR adds two bytes of TCP Option in TCP header called TPSN – TCP Packet Sequence Number. Sender increments TSPN by 1 with every data packet sent (including retransmissions). DOOR-TS supports Time-stamps to be sent as TPSN [3].

4. Responding to OOO Events: - OOO delivery indicates there was a disturbance with the route. As a part of route failure, sender may start congestion control because of the time outs. As soon as sender detects OOO event, TCP restores its state to the one before it started congestion control. As OOO events are detected when ACKs are received, sender assumes that new route has been established successfully. When receiver detects OOO delivery, it can also inform sender by modifying one of the reserved bits of TCP header [3].

III. APPROACHES FOR CHANNEL LOSS

These proposals are non feedback based layered approaches which mainly focus on differentiation between congestion losses and channel losses.

A. TCP WELCOME: -

WELCOME stands for Wireless Environment, Link losses and COngestion packet loss ModEl. WELCOME has LDA - Loss Differentiation Algorithm to differentiate congestion, channel and route failure losses based on analyzing consecutive RTTs and LRA – Loss Recovery Algorithm to take necessary actions. LDA has following rules to differentiate various losses [4].

1. Identify Channel Loss: - If subsequent values of RTT are not fluctuating and remain almost steady around the average value, it indicates there is no congestion. In this case if a packet loss is detected by duplicate acknowledgements, it is considered as channel loss.

2. Identify Route Failure Loss: - If subsequent values of RTT are not fluctuating and remain almost steady around the average value, it indicates there is no congestion. In this case if a packet loss is detected by retransmission time out, it is considered as route failure loss. If route reestablishment time is less than RTO, TCP detects the loss by duplicate acknowledgement and considers as a channel loss.

3. Identify Congestion Loss: - If subsequent values of RTT are fluctuating drastically, it indicates that the network's load is getting changed rapidly. In this case if a packet loss is detected by duplicate acknowledgements or retransmission time out, it is considered as congestion loss.

Loss Recovery Algorithm updates two values as per the type of loss identified [4].

4. RTO - Retransmission Time Out Recalculation: - RTO recalculation is not needed if the loss is congestion loss or channel loss. If loss is because of a route failure, RTO should be recalculated as per the status of the reestablished new route. The equation is,

$$RTO\ New = (RTT\ New / RTT\ Old) * RTO\ Old \quad (1)$$

5. Cwnd - Congestion Window Recalculation: - Cwnd recalculation is not needed if the loss is channel loss. If loss is because of a route failure, TCP can continue with the old transmission rate and adjust the rate as per the status of new route with its congestion control. TCP can also decrease transmission by a fixed factor. Cwnd is recalculated when congestion loss is detected. Cwnd can also be recalculated when route failure loss is detected. The equation is,

RTO should be recalculated as per the status of the reestablished new route. The equation is,

Cwnd New = (RTT Old / RTTNew) * Cwnd Old (2)

WELCOME uses 10 Consecutive RTT values for classification, 5 Consecutive Growing RTT values for congestion detection. RTT value which is within 10% range of the average is considered as stable.

IV. Approaches for Contention Issues

In a wireless network, the main issue is the spatial reuse: - the number of packets that can be in transmission at any time. There is a continuous competition among data and ACK packets of a same TCP connection to access the available bandwidth which may generate collisions and other contention based issues. The most suitable way to deal with this issue is to reduce the number of packets being transmitted. ACK-Acknowledgement Thinning is a process of reducing the flow of ACK packets to provide more bandwidth to the data packets. ACK Thinning is not possible when ACKs are always piggybacked. Even though exclusive ACK packets are very small, they consume same signaling overhead [5].

A. TCP DA: -

DA stands for Delayed Acknowledgement which is the cumulative acknowledgement nature of standard TCP. Here the delay parameter is fixed to 2 segments. TCP receiver starts an acknowledgement delay timer (100- 500 ms) on receiving an in-order segment when all the previously received segments have been already acknowledged. TCP doesn't send an ACK immediately but waits for another in-order segment or until the timer times out. So on arrival of an in-order segment, if acknowledgement delay timer is running, TCP sends a cumulative ACK immediately otherwise starts acknowledgement delay timer and delays sending ACK until another in-order segment arrives or timer times out. So at any moment, TCP will have no more than 2 unacknowledged in-order received segments [5].

B. TCP ADA: -

ADA stands for Adaptive Delayed Acknowledgement which tries to reduce number of ACKs to one per congestion window size. ADA calculates API – Average Packet Interval which is the average of inter-arrival times of packets to define AET - ACK Expiration Time. ADA postpones sending an ACK until a MDT Timer – Maximum Defer Time Timer times out provided that segments arrive within β * API time limit. If either MDT timer or AET timer expires, TCP sends a cumulative ACK immediately. Various events are explained with examples as below [5].

```
LPA = 0                 // Last Packet Arrival
API  = 0                // Average Packet Interval
α = 0.8, β = 1.2        // Weighing Constants
MDT = 500ms             // Maximum Defer Time

On_Data_Segment_Received( )
        If LPA = 0 Then
                LPA = Now
        Else
                API =  α * API + (1 − α) * (Now − LPA)
                LPA = Now
        End If
        If MDT Timer is not Scheduled Then
                Start MDT Timer for MDT Time
        End If
        Reschedule AET Timer to β * API
End

On_AET_Timer_Timeout( )
        Send a cumulative ACK
        Stop MDT Timer
End

On_MDT_Timer_Timeout( )
        Send a cumulative ACK
        Stop AET Timer
End
```

C. TCP DDA: -

DDA stands for Dynamic Delayed ACK. Delaying ACK is really beneficial only when there are enough segments in the transmission (flight data) to help cumulative scheme to work. For example, if the size of congestion window is 2 segments and receivers delays ACK up to 3 segments, ACK will be triggered only on ACK timer timeout because 3rd segment will never arrive until previous 2 are acknowledged. Size of flight data is unknown to the receiver. DDA has a delay coefficient d to define number of ACKs to delay before sending a cumulative ACK. When d=2, it works as standard TCP. DDA increases d as more and more segments are received in-order. Three thresholds l1, l2 and l3 are defined [6]. Algorithm is shown below.

d = 1

```
l1 = 2, l2 = 5, l3 = 9

N = ISN(0)
If N < l1 Then
        d = 1
        Else If l1 ≤ N < l2 Then
                d = 2
                Else If l2 ≤ N < l3 Then
                        d = 3
                        Else If l3 ≤ N Then
                                d = 4
End If
```

D. TCP DAA:-

DAA stands for Dynamic Adaptive ACK. If receiver delays sending an ACK excessively, sender may times out and retransmits unnecessarily. DAA mainly focuses on reducing amount of such spurious retransmissions. DAA adjusts the delay as per the channel condition to avoid spurious retransmissions at the sender. DAA is a sender and receiver side changing algorithm. Sender side, the number of duplicate ACKs required to trigger fast retransmission phase is changed to 2 from 3, the RTO is increased fivefold [7].

Receiver manages a dwin – Dynamic Delaying Window with size varying from 2 to 4 segments with initial value of 2. Pending_ack_counts keeps track of number of in-order received unacknowledged segments. When Pending_ack_counts becomes dwin, receiver sends a cumulative ACK. With every successful arrival of an in-order segment, dwnd is incremented by 1 up to its maximum value 4. ACK Timeout value is determined by statistical analysis of inter-arrival times across the segments [7]. The algorithm is described below.

```
Pending_ack_count = 0, dwin = 2

If an in-order segment is received Then

        If Pending_ack_count > 0 Then
                Record inter-arrival time to calculate ACK
        Timeout Value
        End If

        Pending_ack_count = Pending_ack_count + 1


        If Pending_ack_count = dwin Then
                Produce a cumulative ACK
                Pending_ack_count = 0
         End If

         If dwin < 4 Then
                dwin = dwin + 1
        End If

Else                    // out-of-order segment is received
        Produce a cumulative ACK
        ack count = 0
        dwin = 2
End If
```

E. TCP COPAS:-

COPAS stands for COntention based PAth Selection. COPAS is a routing algorithm based technique which is not based on ACK thinning process. COPAS is based on using different routes for forward (DATA) and backward (ACKs) traffic as shown in figure 1.

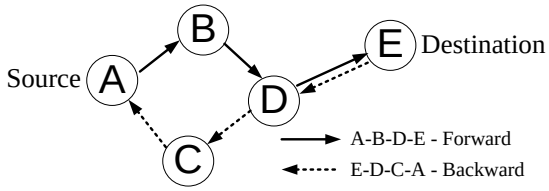

Fig. 1 COPAS

COPAS can be implemented on any on-demand routing algorithm which supports probing based routing with source floods a RREQ – Route Request packet to discover a route to the destination. Destination collects multiple RREQ packets and chooses two routes based on which are disjoint and having less contention. The destination replies with two RREP – Route Reply packet. Destination also sets a direction flag to specify which route belongs to which kind of flow (forward – source to destination – data and reverse – destination to source - ACK). Routes disjointness can be measured easily as each RREQ contains complete information of the route through which it reached a node. Contention is measured by monitoring MAC layer and routes are changed when contention becomes high [8].

V. CONCLUSION

Various approaches explained in this paper are additional features over the traditional TCP mechanism. If these approaches fail, the conventional scheme takes necessary actions. The main purpose of these approaches is to put some intelligence which reduces the amount misinterpretation between congestion loss and non congestion losses. Some latest approaches include split TCP where a single TCP connection is split across multiple small TCP connections. A novel inter piggybacking concept has been introduced where data and acknowledgements of different TCP connections are piggybacked together to resolve contention issues.

REFERENCES

[1]Subir Kumar Sarkar, T G Basavaraju, C Puttamadappa , "Ad Hoc Mobile Wireless Networks - Principles, Protocols, and Applications" - Auerbach Publications Taylor & Francis Group

[2]TD Dyer, RV Boppana, "A comparison of TCP performance over three routing protocols for mobile ad hoc networks", in Proceedings of the 2001 ACM International Symposium on Mobile Ad hoc Networking and Computing: MobiHoc 2001 56–66 (2001)

[3]Feng Wang, Yongguang Zhang, "Improving TCP performance over mobile ad-hoc networks with out-of-order detection and response" Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing Pages 217 – 225

[4]Seddik-Ghaleb, A.,Ghamri-Doudane Y.SenouciS.M. "TCP WELCOME TCP variant for Wireless Environment, Link losses, and COngestion packet loss ModEls", Communication Systems and Networks and Workshops, 2009. COMSNETS 2009.

[5]AK Singh, K Kankipati, "TCP-ADA: TCP with adaptive delayed acknowledgement for mobile ad hoc networks", in Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE. 3, 1685–1690 (2004)

[6]E Altman, T Jiménez, "Novel delayed ACK techniques for improving TCP performance in multihop wireless networks", in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2775, 237–250 (2003)

[7]de Oliveira, T Braun, "A dynamic adaptive acknowledgment strategy for TCP over multihop wireless networks", in INFOCOM, 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE. 3, 1863–1874 (2005)

[8]CDA Cordeiro, SR Das, DP Agrawal,"COPAS: dynamic contention-balancing to enhance the performance of TCP over multi-hop wireless networks", in Computer Communications and Networks, 2002. Proceedings Eleventh International Conference on, 382–387 (2002)