A Review: TCP Variants with MANET

Jaydevsinh B. Vala
Master of Computer Engineering,
Department of Computer Engineering
Atmiya Institute of Technology & Science,
Rajkot- 360005,
Gujarat, India

Hiren V. Mer
Assistant Professor,
Department of Computer Engineering
Atmiya Institute of Technology & Science,
Rajkot- 360005,
Gujarat, India

ABSTRACT
TCP- Transmission Control Protocol is a connection oriented and reliable transport layer protocol of TCP/IP protocol suite. TCP provided process-to-process, stream and full duplex communication. TCP also provides flow control, error correction and congestion control. Congestion is the traffic jam of the packets in the network. It occurs when the load in the network is higher than of its capacity to handle. This paper explains basic congestion control mechanisms used by TCP. This paper also discuss some of the possibilities of future research work in TCP congestion control. This all TCP variants have been proposed to improve TCP congestion control mechanisms. This paper explores some of the most widely used TCP variants conceptually.

Keywords
TCP, Tahoe, Reno, NewReno, Vegas, BuS, ELFN

1. INTRODUCTION
Congestion is situation in a computer network when the number of outstanding packets becomes difficult to handle by the internetworking devices. An intermediate device like router, switch has a limited amount of memory-buffer and processing capabilities. Congestion occurs when we force a network and its devices to work beyond their capacities. When a router is supplied more than of its capacity to process, router suffer from an traffic jam kind of situation which is called congestion. As a result, Router may discard few packets which is the side effect of it[1].

TCP variants can be broadly classified into two major categories based on their strategies of congestion control.

Table 1 Categories of TCP Variants

Reactive TCP Variants	Proactive TCP Variants
Based on Congestion detection.	Based on congestion avoidance.
Detect congestion after causing it.	Detect congestion before causing it.
Packet loss is the feedback signal.	Packet delay is the feedback signal.
Corse – grained timers.	Fine-grained timers.
TCP Tahoe, TCP Reno, TCP NewReno are examples.	TCP Vegas is an example.
Less accurate Round Trip Time estimation.	More accurate Round Trip Time estimation.

This paper explore reactive TCP-Tahoe, Reno, New Reno and proactive TCP-Vegas. Some of the most widely used TCP variants.TCP variants for adhoc networks are also discussed.

2. TCP CONGESTION CONTROL
TCP performs congestion control in three phases:- Slow Start, Congestion Avoidance, Congestion Detection. TCP uses acknowledgement to check packet loss and find packet delay. An Acknowledgement method can be either cumulative or selective. A cumulative acknowledgement with 2001 informs the sender that the bytes around 2000 sequence number have been received successfully. Receiver may send a cumulative acknowledgement for few segments together to reduce acknowledgement overhead. A selective acknowledgement informs the sender about outorder delivery of segments and so sender can further send only the missing segments[2].

2.1 Sender Sliding Window
TCP maintain two sliding windows variables - C_window-congestion window and R_window-receiver's advertised window. Sender estimate and change the size of C_window as per the congestion situation of the network. Receiver advertises the capacity at which it can receive segments in the form of R_window. Receiver sends R_window as a part of TCP header field called window size. Sender selects the minimum of C_window and R_window to reduce possibility of congestion as well as the possibility of overwhelming the receiver.

$$Window = Min (C\_Window, R\_Window) \quad (1)$$

TCP sliding window is a byte oriented. Sender side it supports open, close operations while receiver side it supports open and close operation based on basis of sliding window concept. In our discussion we assume that R_window>>C_window for all the cases.

So Window = C_Window[2].

2.2 Slow Start – Exponential Increase
TCP starts the transmission with initially very small C_window,1-2 Sender Maximum Segment Size. The purpose of slow start is not to overwhelm the network without knowing the current situation. Slow start increases the size of C_window by 1 with every successfully received ACK-acknowledgement. So after every RTT-Round Trip Time, C_window gets doubled. TCP sets ssthresh – Slow Start Threshold value. TCP continues in Slow Start until C_window ≥ ssthresh[2].

2.3 Congestion Avoidance – Additive Increase
TCP enters into congestion avoidance phase once C_window becomes greater than or equal to ssthresh. TCP still continue to increase the rate of sending by increasing C_window but not as fast as it does in slow start. In Congestion avoidance, TCP increments C_window by 1 with every RTT, So it is called additive increase. C_window= C_window +

1/C_window per Round. TCP continues in this phase until retransmission times out. On retransmission times out, TCP sets ssthresh to half of the C_window, sets C_window to 1 and starts slow start phase again.

The above policies were purposed in the initial standard of TCP; later on many TCP variants have modified them while keeping the basis intact[2].

3. TCP TAHOE

TCP Tahoe has introduced a fast retransmission phase over the slow start and congestion avoidance. Tahoe states that it is possible to detect congestion even before RTO-Retransmission Timer times out. Whenever receiver receives an out of order segment, it sends a duplicate ACK immediately. Sender Tahoe process counts the number of such duplicate ACKs. On receiving 3 same duplicate ACKs, Tahoe considers a packet loss and switches to slow start phase. This early retransmission is called fast retransmission.

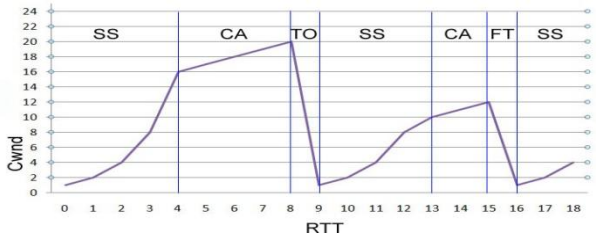

Figure 1. TCP Tahoe

SS-Slow Start, CA-Congestion Avoidance

TO-Time Out, FT- Fast Retransmission

Initially ssthresh=16 and so on round 4, SS phase completes and CA phase is started. On round 8, retransmission time out occurs which sets ssthresh to half of the current C_window. so ssthresh becomes initially 10. On round 13, SS phase ends and CA round starts. On round 15, three duplicate ACKs cause fast retransmission in which ssthresh is set to 6 and SS phase starts. Tahoe treats time outs as well as 3 Duplicate ACKs same way.

4. TCP RENO AND TCP NEW RENO

Tahoe switched to slow start phase in both the cases, retransmission time outs and fast retransmission. Retransmission time out is stronger possibility of congestion and moving to slow start phase is required. But receiving 3 duplicate ACKs is a weaker possibility of congestion. With 3 duplicate ACKs, it is necessary to slow down the rate, but not completely because there are still packets are delivered in the network and so duplicate ACKs are received. TCP Reno focuses on this logic. Reno introduced a Fast Recovery phase other than of slow start, congestion avoidance and fast retransmission[3].

In case of Retransmission time out occurrence, ssthresh is set to half of the current C_window, C_window is set to 1 and new slow start phase starts. In case of 3 duplicate ACKs, ssthresh is set to half of the current C_window, C_window to ssthresh+3 and starts congestion avoidance phase. Here C_window is set to ssthresh+3 because 3 out of order segments are delivered which caused 3 duplicate ACKs. This phase is known as congestion detection with multiplicative decrease.

Once Reno detects 3 duplicate ACKs, it immediately halves the C_window and starts fast recovery phase. Reno stays in the fast recovery phase until a fresh ACK, acknowledges some of the sent data and then switches to congestion avoidance phase. Fast recovery is something between slow start and congestion avoidance where Reno stays until it detects that receiver has started receiving something.

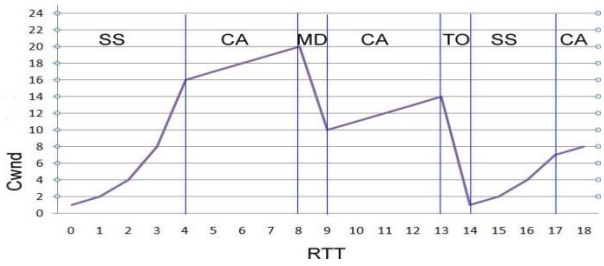

Figure 2. TCP Reno

SS-Slow Start, CA-Congestion Avoidance

TO-Time Out, FT- Fast Retransmission, MD-Multiplicative Decrease

Figure 2 shows behavior of Reno, Initially ssthresh=16 and so on round 4, SS phase completes and CA phase is started. On round 8, Reno receives 3 duplicate ACKs, which sets ssthresh to 10, C_window to 10 and starts congestion avoidance phase. On round 13, a conventional time out is detected which starts slow start phase and then congestion avoidance phase from round 17[3].

Reno performs good when it is one packet loss in a window of outstanding packets. This is because Reno comes out of the fast recovery phase once it receives a fresh ACK. Reno doesn't care whether the new ACK, acknowledge all the outstanding packets or not. So if there are multiple packet losses in a single window,it not perform well. NewReno has modified fast recovery phase. NewReno stays in the fast recovery phase until all the outstanding packets are acknowledged successfully. Each ACK which acknowledges some of the packets in middle of the window is known as Partial ACK. An ACK which acknowledges all the outstanding packets of a window is a full ACK. Reno comes out of the fast recovery on receiving a partial ACK while NewReno considers partial ACK as a possibility of further loss of packets and keeps staying in the fast recovery phase until a Full ACK comes.

Reno and NewReno perform very well as compared to Tahoe but they are able to detect only one packet loss per RTT. This limitation can be overcome with TCP SACK and TCP FACK[3].

5. TCP VEGAS

Vegas is a proactive TCP variant, which detects congestion before congestion occurs. It uses packet delay as a primary feedback signal. Vegas performs 40% to 70% better than Reno in throughput as well as one half to one fifth of reduction in retransmission requirements. Vegas check the beginning of congestion by observing the difference between the expected rate and actual rate. Vegas is based on five techniques to improve performance by increasing throughput and decreasing spurious retransmissions[4].

Accurate RTT Calculation

Course grained timers are used one per connection and they are not accurate with reference of individual segments. It is based on using fine-grained timers. Whenever it sends a segment, it stores current system clock as a time stamp for a segment. So it is possible to calculate exact RTT for each successfully acknowledged segment. its RTT calculation

provides more accurate time out calculation which can be used to decide whether to retransmit a segment or not more precisely.

Retransmission Requirement Detection
If Vegas specific retransmission detection fails, it follows the conventional time out and 3 duplicate ACKs based retransmission scheme. It extends the 3 duplicate ACKs based retransmission scheme as follow.

1. On receiving 1st duplicate ACK, Vegas find the difference between the current time and the sent time stamp of a segment which is requested with duplicate ACK. If the difference is more than the coarse grained time out value, Vegas immediately retransmits the requested segment by a duplicate ACK without waiting for 2 more duplicate ACKs. When losses are too high or window is too small, sender will never receive 3 duplicate ACKs which will cause time out. This is one of the major issues of Reno which is solved by Vegas.

2. on receiving a fresh ACK – non duplicate ACK, if it is 1st or 2nd ACK after a retransmission, Vegas checks whether the time interval since the segment was sent is larger than the time out or not. If it is,it retransmit the segment. This catches any other lost segment previous to the retransmission without waiting for a duplicate ACK. This strategy is used to identify multiple segment losses in a window.

Later, sender sends segment 13. It gets lost and after some time sender gets ACK asking for segment 13. At this point the time difference between sent time of segment 13 and arrival time of acknowledgement asking for segment 13 is larger than time out time and so sender immediately resends segment 13. After retransmission of segment 13,Vegas checks the time difference between sent time of segment 15 which was lost and arrival time of acknowledgement asking for segment 15. As it is larger than time out time, Vegas immediately resends segment 15 too[4].

Reduction in Window Size
The windows size should be decreased only if the losses have occurred due to current sending rate not because of any higher previous sending rate. Vegas compares the time of retransmission of last segment and time at which windows was modified last. When the retransmitted packet was sent before the decrease, it will not decrease window size on receiving any duplicate ACK for that segment because packet loss was with reference of previous window size. This scheme reduces unnecessary slow down of the sending rate[4].

Modified Congestion Avoidance
Vegas compares measured throughput rate with the expected throughput rate. it believes that the number of outstanding bytes is directly proportional to the throughput. So if it increases windows size, number of outstanding bytes increases and subsequently throughput should.The primary goal of Vegas is to manage right amount of extra bytes. If Vegas sends too much extra bytes, it may cause congestion.

Vegas calculates the BaseRTT which is RTT of a segment when there was no congestion. BaseRTT can be minimum of all the measured RTT times. Practically it is the RTT of the first segment sent on a connection. At any moment, expected throughput is,

Expected = WindowSize / BaseRTT (5)

WindowSize is the current congestion window size which we assume to be equal to the number of bytes in transit. Vegas calculates the actual – current throughput per round as,

Actual = WindowSize / Average Measured RTT (6)

Expected throughput represents available bandwidth in absence of congestion. Actual throughput represents current bandwidth being used by the connection. Vegas measures the difference between Expected and Actual throughputs and changes the congestion window C_window accordingly. Let,

Diff = Expected – Actual (7)

Modified Slow Start

The conventional slow start doubles the C_window every RTT which is quite aggressive way because of exponential increase. Being a proactive variant, Vegas doubles the C_window every other RTT only[4].

6. TCP for MANETs

TCP-F:
TCP-F means TCP-Feedback. If any intermediate node detects route failure, it immediately informs the source to avoid unnecessary starting of congestion control using a RFN – Route Failure Notification message. RFN message is propagated towards the source. Mean while if any intermediate node finds an alternative route, it diverts packets to new path and discards RFN message. If no alternative route is available, RFN message reaches to the source. On receiving a RFN message, source immediately enters into the freeze – snooze state. In freeze state, source stops further transmission, saves the transmission status (window, RTOs etc) and starts a RFT – Route Failure Timer. TCP remains in the freeze state until it receives a RRN – Route Re-establishment Notification message or RFT times out. TCP changes its state from freeze to active on receiving a RRN message and continue with the transmission status which was saved earlier. TCP also changes its state from freeze to active on RFT time out but it retransmits all the unacknowledged packets immediately which may cause burst of traffic [5].

TCP with ELFN:
ELFN - Explicit Link Failure Notification based scheme is similar to TCP-F but it involves real interaction between TCP and routing protocol. When a node detects route failure, it sends a ELFN message to the source. ELFN message is similar to "host unreachable" message of ICMP – Internet Control Message Protocol. On receiving ELFN message, source enters into freeze – standby mode by pausing transmission. Source periodically get information about route reestablishment. If acknowledgement of probe message is received, TCP leaves the standby mode and resumes transmission. Route failure message of DSR- Dynamic Source Routing algorithm is piggybacked to carry route failure message information for TCP. ELFN message contains source and destination addresses and port numbers as well as TCP segment's sequence number. ELFN performs poor when load is high because of probing based nature [5].

TCP BuS:
BuS Stands for Buffering Capability and Sequencing Information. TCP BuS uses a reactive ABR – Associative Based Routing protocol. TCP BuS is based on following four improvements [6].

1. Explicit Notification: - A node which detects route failure is called PN – Pivot Node. PN informs the source about route failure and route re-establishment with a ERDN - Explicit Route Disconnection Notification message and ERSN - Explicit Route Successful Notification message respectively.

On receiving ERDN message, source freezes transmission and on receiving ERSN message, source resumes transmission [6].

2. Extending Timeout Values: - Route re-establishment process is called RRC – Route Reconstruction Phase. During RRC phase, all the packets which are at any node between source and PN are buffered. Source TCP may get timeouts for buffered packets due to lack of acknowledgements. So on receiving ERDN message, source TCP increases RTO (mostly doubles) for these buffered packets [7].

3. Selective Retransmission Requests: - There may be loss of some packets in the path from source to PN. These packets can be selectively retransmitted to buffer missing packets. Buffered packets are not forwarded until a new route is established between PT and destination [7].

4. Avoid Unnecessary Fast Retransmission: - There may be loss of some packets in the path from PN to Destination. There are already few next packets which are buffered in the path from source to PN. On new route establishment, destination informs source about the lost packets. The buffered packets reach to the destination before those retransmitted lost packets. Because of out-of order delivery, destination generates duplicate acknowledgements for fast retransmissions. Source avoids such unnecessary fast retransmission [8].

ECIA based TCP:
ECIA stands for Exploiting Cross-layer Information Awareness. ECIA is an improvement over TCP – ELFN. Loss of data packets and acknowledgements may cause retransmission time outs. ECIA suggests two mechanisms called EPLN – Early Packet Loss Notification and BEAD – Best Effort Acknowledgement Delivery. Conceptually ECIA is similar to TCP BuS but it doesn't focus on buffered packets. When an intermediate node detects a route failure, it informs the sequence number of every lost packet to the sender via EPLN. Source disables RTO for these packets and retransmits from lowest sequence numbered packet once new route is established. Similarly, intermediate node informs the destination regarding lost acknowledgements via BEAD. Destination resends the acknowledgement with the highest sequence number by following cumulative acknowledgement concept. The DSR – Dynamic Source Routing protocol is modified to implement ECIA based scheme [8].

Preemptive Routing Based Schemes:
Preemptive routing tries to predicate route failure based on the signal strength variations. This prevents all of sudden disconnection and loss of packets. When an intermediate node detects signal strength which is below a primitive threshold, it informs source to start route discovery phase. Ping-pong based small messages are proposed to measure signal strength of a transmission between two nodes. DSR and AODV protocols are modified for preemptive routing. In signal strength based link management, a node can try to increase transmission range of a node so that the packets in transit can reach to the destination. RFP – Route Failure Prediction mechanism maintains history of signal strengths to find the speed at which two nodes are moving away from each other. This information is used to predicate by which time a route may get failed. To avoid all of sudden disconnection, source is informed in advance to start route discovery [8].

TCP-F and ELFN based TCP both are based on route reestablishment after route failure. TCP – F allows intermediate node to continue with any alternative path if it knows while ELFN doesn't. In both the schemes, it is possible that the new path is longer and so time out occurs. It is also possible that the bandwidth of new path is not suitable with the old value of congestion window which was calculated at the time of old path. TCP-F is non-probing based while ELFN is probing based. ELFN is simulation based on modification of DSR routing protocol while TCP-F scheme is emulation with any of the existing the routing protocol [10].

TCP BuS and ECIA based schemes try to avoid unnecessary fast retransmissions. TCP BuS focuses on managing the buffered packets while ECIA focuses on sharing information about lost packets and acknowledgements. While TCP-F and ELFN inform only source about the route failure, TCP Bus and ECIA inform both source and destination. Preemptive Routing Based Schemes focus on informing source about the route failure before it actually occurs [10].

C3 TCP:
C3 stands for Cross-layer Congestion Control. In a wireless multi-hop network, source's link layer buffers packets while other transmissions are going on. Source has to wait for the channel access until the medium becomes free. Available wireless channel bandwidth is shared by all the nodes which are located in the transmission range of the source as well as of the destination. A node follows medium access mechanism with CSMA/CA protocol with addition of RTS-CTS signaling packets based MACA protocol [10].

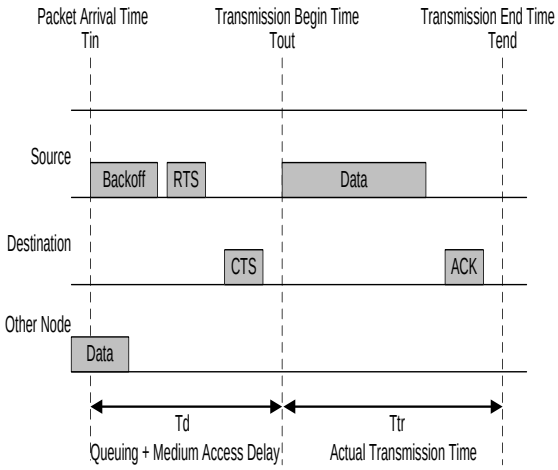

Fig. 3 Medium Access and Data Delivery Process

Fig 3 shows medium access and data delivery process with reference of the link layer. Every node has a buffer queue where it stores incoming packets until medium becomes free to access. Packets may be originated by the same node or received for the forwarding purpose from the neighboring nodes. Suppose at time Tin, a new packet becomes ready to send. Source senses the energy of the channel and finds it busy because of other node's communication. Source enters into the waiting state by following exponential backoff mechanism of CSMA/CA. after back off time out; Source finds the medium free and sends RTS-Request to Send to the destination. Destination sends CTS – Clear to Send back to the source. On arrival of CTS, source sends data frame and on receiving of data frame, destination acknowledges source. Based on this concept, C3 TCP estimates bandwidth and delay. This information is used for the congestion control purpose [9].

Adhoc TCP:
The congestion window defines the tolerable transmission rate for a connection based on a route status through which it is associated. Route failures damage the relationship between congestion window and tolerable transmission rate. Adhoc TCP is a thin layer between IP and TCP [10]. Various states of Adhoc TCP are shown in figure

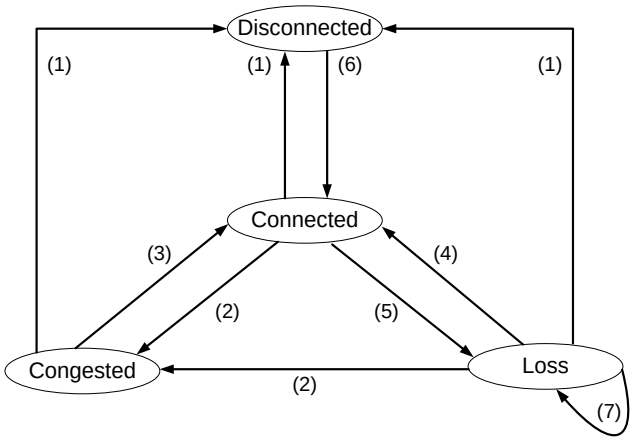

Fig 4. Adhoc TCP

1) Receive Destination Unreachable ICMP Message
2) Receive ECN
3) TCP Transmits a packet
4) New ACK
5) 3 Duplicate ACKs or RTO expiration
6) Receive Duplicate or New ACK
7) Retransmission

Adhoc TCP has four states: Normal (Connected), Congestion Control (Congested), Persistent (Disconnected) and Retransmit (Loss). Adhoc TCP listens to ICMP – Internet Control Message Protocol messages to put sender TCP in persistent state (freeze state until a new route is established). Adhoc TCP listens ECN – Explicit Congestion Notifications to put sender TCP in congestion control state. On occurrence of 3 duplicate acknowledgements or RTO time out, sender TCP enters into the retransmission state [10].

7. CONCLUSION

In the client –server era, most of the research work is going on towards improving the performance at the sender side. Sender and receiver communicate using network. Sender focused improvement is easy to adopt by doing necessary changes at the servers without expecting clients-receivers to be upgraded. A novel scheme can be introduced in which user can use the four reserved bits of TCP header to send messages to the sender. Receiver assistant congestion controls schemes can be more advantageous in interactive applications. A perfect combination of all the three congestion control feedbacks – packet loss, packet delay and explicit notifications by intermediate routers improves the TCP performance drastically.here can do work on congestion control on MANET by different variants.

8. REFERENCES

[1] M. Allman, V. Paxson and W. R.Stevens, "TCP congestion control", in IETF RFC, 2581, 1999.

[2] V. Jacobson, "Modified TCP congestion avoidance algorithm. nd2endinterest mailing list", in Tech. Report in IEICE Transactions on Communications, 2007, E90-B(3):516-52, 1990.

[3] T. Bonald, "Comparison of TCP Reno and TCP Vegas: efficiency and fairness", in Performance Evaluation, Vol. 36-37, pp. 307-332, 1999..

[4] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno, and SACK TCP", in ACM Computer Communication Review, Vol 26, pp. 5-12, 1996.

[5] V. Jacobson, "Congestion avoidance and control", in SIGCOMM Symposium on Communications Architectures and Protocols, pp. 314– 329, 1999.

[6] J. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP", in Proceedings of SIGCOMM Symposium, pp. 270-280, 1996.

[7] Ha, S., Rhee, I. and Xu, L. (2008). CUBIC: A new TCP-friendly ighspeed TCP variant. ACM SIGOPS Operating Systems Review. 42(5), 6474.

[8] Y. -C. Lai and C.-L. Yao, "Performance comparison between TCP Reno and TCP Vegas". Computer Communication, 25: 1765-1773. 2002.

[9] Molia,Hardik k and Rashmi Agrawal,"A conceptual exploration on TCP Variants",2014 2nd international conference on emerging technology trends in Electrionics communication and Networking,2014

[10] Molia,Hardik k and Rashmi Agrawal,"A comprehensive study of cross layer approaches for improving TCP performance in wireless networks",2015 international conference on computing and communication technologies,2015