Journal of Theoretical and Applied Information Technology
31st December 2019. Vol.97. No 24
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TCP AT-ER: TCP ADAPTIVE TIMEOUT BASED-ON EARLY RETRANSMISSION OVER MANET

NJOUD O. Al-MAAITAH[1]

Lecturer, Computer Science Department of IT College, Mutah University, Jordan

E-mail: [1] njoudmaitah@mutah.edu.jo

ABSTRACT

Transmission control protocol (TCP) is a transport layer protocol that provides many services to the network, such as reliable transmission and flow control. TCP provides reliability either by using timeout retransmission that based on timeout calculation or by fast retransmit mechanism that depends on receiving three successive duplicated acknowledgments. However, waiting for along period of timeout or waiting for three duplicated acknowledgments, could increase total delay and decrease the data transmission in a wireless network, which in turn negatively affects its performance. Since TCP was originally designed for wire networks, it suffers from low performance over Mobile Ad hoc network (MANET). TCP interprets packet loss as congestion and blindly doubling timeout value by ignoring any other reasons such as link failures due to mobility or high bit error rates. Therefore, we need calculating a suitable value of timeout which reflect the real reason of loss. This paper modifies the timeout calculation approach in order to obtain more accurate retransmission time based on MANET environment state. It uses the difference time between the first and second acknowledgments as indicator to network state. Also it proposes a reduction of the number of required duplicated acknowledgments to two rather than three to solve a case when there are no sufficient number of duplicated acknowledgments to fire fast retransmit. Using a discrete simulator, a set of experiments with various traffic load and mobility speeds have been conducted to assess the effect of proposed method (AT-ER) over TCP New Reno. Results have shown that AT-ER improves TCP New Reno throughput by 20% over MANET.

Keywords: *TCP, New Reno, Timeout, Fast Retransmit, adaptive timeout, MANET.*

1. INTRODUCTION

1.1 Wireless Network and Mobile Ad hoc Network

The emergence of mobile devices, which connected wirelessly with each other to share information, is now playing a very important role in our life. Wireless networks can be arbitrary classified into two categories: infrastructure based networks and infrastructure-less based networks. While the former relays on central point to control the communication, each host in the latter plays an important role of client and server in the same time. This type commonly called "ad hoc wireless network" where the devices connected in dynamic nature [1]. Ad hoc can used to connect buildings, vehicles, ships, etc. It has a huge number of applications especially when the environment is hard or danger, for example in battle fields or in festival sites. Therefore, ad hoc is considered the best way to communicate computers in harsh circumstances due to its flexibility [2]. That comes because all hosts in it move freely which means that no specified topology is exists. Figure 1 shows how ad hoc wireless communicates between four devices.

Each node in ad hoc network sends and receives data, which means every node plays a role of host and destination in the same time. If one node needs to communicate with another node out of its coverage, it would rely on its neighbor's node(s) as intermediate hop until reaching the final destination. Ad hoc dynamicity also provides many ways of communicate, in some cases source can choose to make broadcasting the message to all next nodes for saving transmission time. While in another circumstances, the path between source and destination is chosen carefully to save power consumption or to avoid threats [3]. Figure 2 compares between infrastructure and infrastructure less networks.

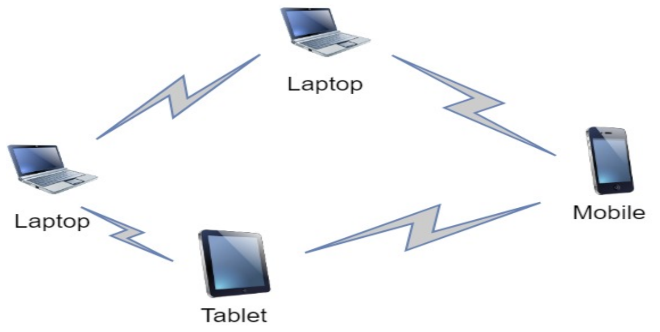

Figure 1: Ad Hoc Wireless Network

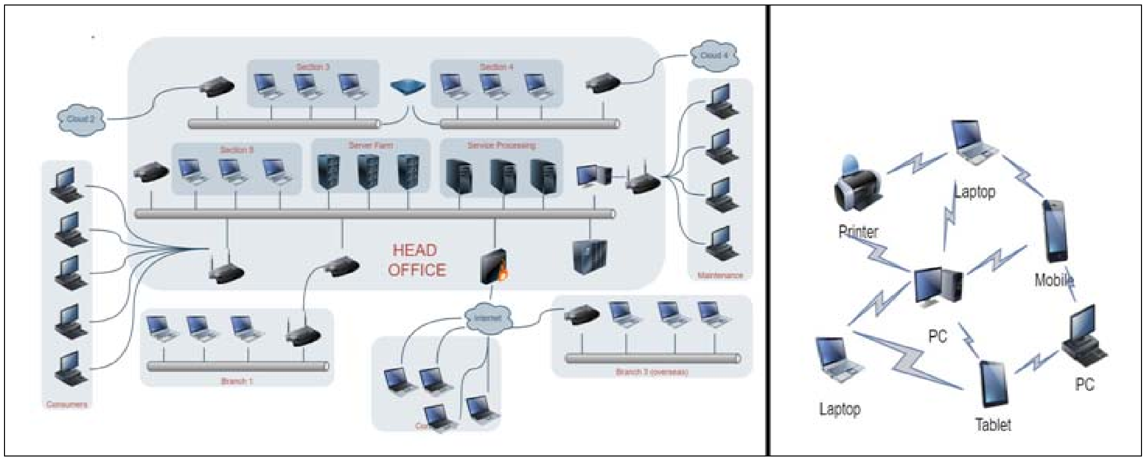

Figure 2: Network Structures (Infrastructured (On Left) And Infrastructure-Less (On Right)

Many routing protocols for mobile ad hoc (MANET) have been designed. Some of routing protocols designed also to provide security requirements, for example encryption and authentication. The most routing protocol designed for MANET called AODV which stands to "Ad hoc on- demand distance vector" protocol [13].

1.2 Transmission Control Protocol

Transmission control protocol (TCP), Transport layer protocol, provides reliable transfer service. It guarantees that each byte sent will be received. When some bytes (i.e. segments or packets) are dropped through many circumstances (i.e. contains errors) , TCP recovers them by acknowledgments (ACKs) and retransmission procedures. In case that data packets are delayed which make it received after another packets, reordering applied by using sequence numbers.

In addition, flow control and congestion control are two different and amazing functions provided by TCP. Using flow control TCP ensures that sender will not pump too much packets into network which cannot be absorbed by receiver. This can be by allowing some kind of feedback about current state from receiver to sender.

Congestion control procedures keep network light and running well as possible. TCP, at the beginning of transmission, takes precaution by pumping data packets slowly into network. The sequence can be summarized as follow: Firstly, the sender specifies the number of packets for transmit using some kind of buffer named as congestion window (cwnd). The value of cwnd for first time is one segment (i.e. packet or message). When receiving ACK that confirms receiving of this segment the sender responses by sending two segments at a once. The sender waits until ACK that confirm the receiving of the two segments arrived. When ACK arrived correctly, cwnd sets to be four which means double of previous cwnd value. The exponential increasing of cwnd continues while cwnd is still less slow start threshold (ssthresh). When cwnd reaches ssthresh slow start phase ends and another phase called congestion avoidance starts. Congestion avoidance increases cwnd by only one segment each time of receiving ACK. Figure 3 clarifies the slow start and congestion avoidance algorithms. From Figure 3 it is clear that cwnd value keeps increasing linearly until packet loss detected [14].

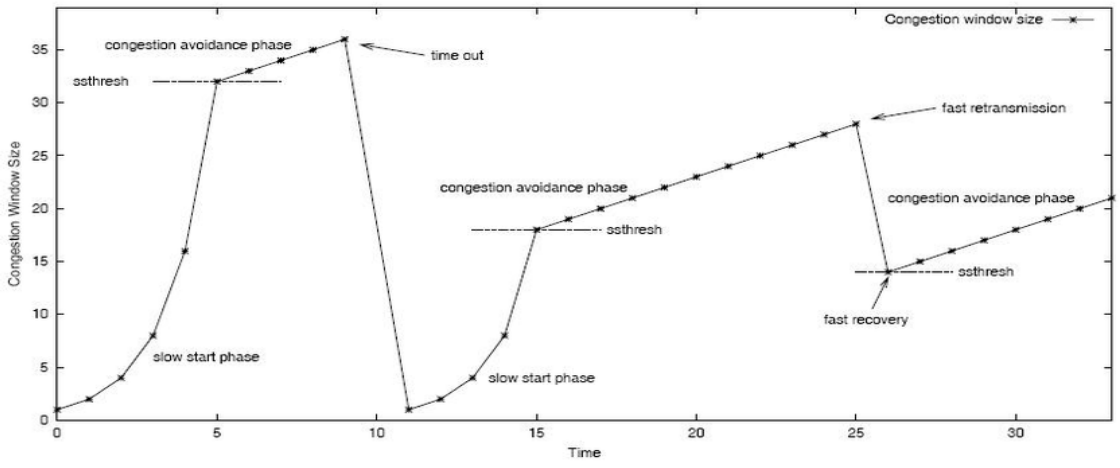

Figure 3: Slow Start, Congestion Avoidance, Fast Retransmit And Fast Recovery Algorithms

TCP is the embodiment of reliability for end-to-end data transmission. It takes a decision to recover the detected lost packets using two ways, the first one uses a timer that starts after sending packet and rests when acknowledgment back to provide guarantee about packet reception. It is particularly obvious that timer fires when no acknowledgment backed to the sender who will retransmit the packet again. in such case, TCP also reduces cwnd to one segment and makes slow start begins again. in one of TCP versions called TCP Tahoe (1988) mainly relays on this way to deal with loss of packet(s), that main problem was in TCP Tahoe was the high delay which comes as a consequence of doubling RTO each time loss detected. this means that network be idle for a significant period of time [23].

TCP Reno suggested as a modified version of TCP Tahoe in 1990 by Jacobson, it provides fast retransmit and fast recovery algorithms [23].

In fast retransmit, TCP retransmits lost packet before timer fires but after receiving three duplicated acknowledgement that all ask for a particular packet, cwnd is reduced to half and increased linearly by receiving new ACK., Figure 3 and figure 4 explain how this work when packet loss is detected. Although TCP Reno was really helpful to detect loss of one packet, it was act as Tahoe with multiple packets losses. TCP New Reno uses a modified Fast retransmit and fast recovery algorithms to solve this situation.

Both Fast retransmit and fast recovery are congestion control algorithms that come to recover lost data in TCP Reno and TCP new Reno versions [9]. TCP New Reno is more powerful than Reno to recover multiple lost packets. while Reno exits from Fast recovery phase, as shown in figure 3, when three duplicated acks are received.TCP New Reno does not exit from fast recovery until all outstanding packets are acknowledged or timeout event happened.

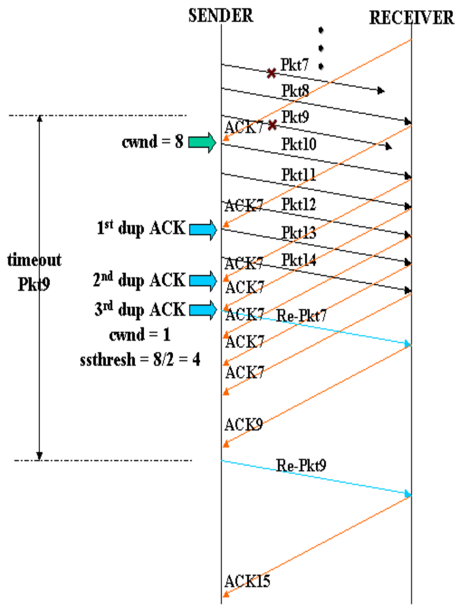

Figure 4: Fast Retransmit And Fast Recovery Algorithms.

TCP new Reno calculates retransmission timeout using Jacobson study [6]. The equations provided optimal value of waiting time on wired networks which are considered as stable network. Moreover, the packet losses in wired networks often happen due to congestion.

The original simple algorithm equations compute timeout as a function of estimated round trip time (ERTT). ERTT value based on a weighted value of trip time called smoothed round trip time (SRTT) which calculated as the difference between sending time of a packet and receiving time of acknowledgment [7].

By using variable r to denote receiving time of ACK, and **p** to denote previous time of packet sending. The equations below show how timeout value computed:

First, computing SRTT as

$$SRTT = r - p \qquad (1)$$

Then find ERTT as

$$ERTT = \alpha * ERTT + (1-\alpha) * SRTT \qquad (2)$$

Where α is a constant weight ranging between [0.8 to 0.9]

Which leads to $Timeout = 2 * ERTT \qquad (3)$

Jacobson and Karle introduced their equations to compute timeout with ability to avoid congestion in the network [7]. Their way allows the sender to compute a new sample of round trip time (SampleRTT) then reflect this value on timeout calculation. This comes as a solution of original approach's problem which does not take a consideration of SampleRTT variance. Equations below observe that:

$$Difference = SampleRTT - ERTT \qquad (4)$$

$$ERTT = ERTT + (\delta * Difference) \qquad (5)$$

where δ from 0 to 1

$$Deviation = Deviation + \delta * (|Difference| - Deviation) (6)$$

then computes Timeout as follow :

$$Timeout = \mu * ERTT + \theta * Deviation. (7)$$

Based on experiments and idle results, μ sets to 1 and θ sets to 4.

1.3 TCP over MANETs

TCP is still used as transport layer protocol for MANET, but along many years TCP suffers from degradation in performance when applied over such wireless networks. that because MANET nature is full of challenges such as:

I. Nodes Mobility

The main feature of mobile Ad hoc network is the free nodes movement. Therefore, a node can enter or exit from the network at any time without any restrictions. As a consequence of two problems arise, first one is frequent link breakages which means a considerable time will be spent waiting for acknowledgments of data which will never reach until new path is discovered. Many timeout events will occur and many packets need retransmission, where each timeout event followed by duplication of retransmission timeout value (i.e. causes more time to wait) and also reducing of congestion window to one packet to activate the slow start phase. All this leads to reduce overall performance of network. The second problem, is network partition which means that source and destination of data (i.e. sender and receiver) are located in two different parts of network which also leads to route failures and many losses of packets [16].

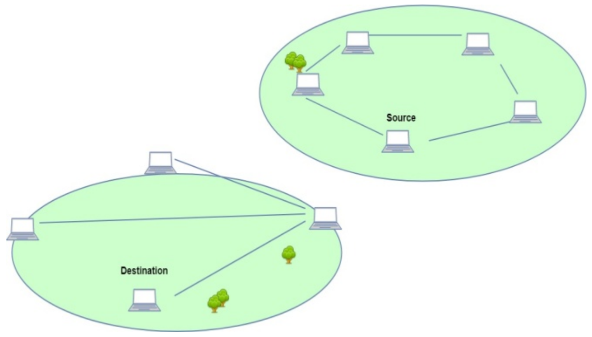

Figure 5: Wireless Network Partition

II. High Bit Error Rate

MANETs communicate over radio frequencies. As a result of this type of communication channels, many transmitted bits of any packet could be altered due to many reasons such as distortion, noise and interference. The number of altered bits per time unit called bit error rate (BER). Any packet arrived with bit error rate is corrupted and considered as lost packet by TCP. the receiver will not generate acknowledgment while sender is waiting for a period of time (RTO). when timer of RTO fires cwnd of sender closed to one segment, slow start begins again from scratch and new RTO value assigned to double of previous value. Repeated errors means repeated corrupted packets which causes that cwnd at sender remains small, and low throughput will be gained [18].

Any packet loss either due to link failures, high BER, handoff, fading or taking wrong path with delay ability are considered as congestion by TCP. Under this wrong consideration, TCP procedures to recover lost packets are taken. TCP either wait for a period of time before retransmit the lost packet(s) or the retransmission is triggered when three or more duplicated acknowledgments are received. When TCP indicates the loss by a timeout, half of the current window size is saved in slow start threshold, congestion window is set to one segment through slow start phase and the new timeout value takes double of previous one [5]. Duplication of timeout value means more time to wait in the next loss which causes more delay in the network. The second indication of loss called "fast retransmit" which resends the packet after three or more duplicated acknowledgments delivery even if the timeout for that packet has not expired yet and a half of the current window size is saved in slow start threshold. In some cases, the congestion window is small and not enough to generate the required number of duplicated acknowledgments to invoke fast retransmit. The only last solution in such case is waiting timer to fire to resend the packet again [6]. Depending on that, retransmission timeout value must be calculated carefully.

One of main reasons of low performance of TCP over wireless networks, is that by moving to wireless technology many considerable changes are carried out on communication stack that exists in lower layers (i.e. Network, data link and physical layers) without taking in consideration their effect on upper layers, Figure 6. As a result TCP, which belongs to transport layer, acting over wireless still same of its acting over wired network. Route partition and breakage, hidden node, channel contention and high bit error rate are the biggest challenges associated with the nature of the wireless network that TCP will face if it did not modified [16].

Ad hoc network inherits all properties of wireless networks with possibility of motion. Hence, TCP challenges become bigger over ad hoc. TCP will not unable to detect the packet loss reason in such environment, It limits any loss cause to congestion. The timeout is being unavoidable when there are no enough number of duplicated acknowledgments captured. This study aims to help the TCP to be more adaptive with mobile ad hoc network state to be able to assign appropriate timeout value based on the state of network.

This paper proposes a modification of fast retransmit mechanism and RTO calculation to ensure TCP adaptively. TCP AT-ER should make TCP able to sense current Ad hoc network state to assign an adaptive value of RTO.

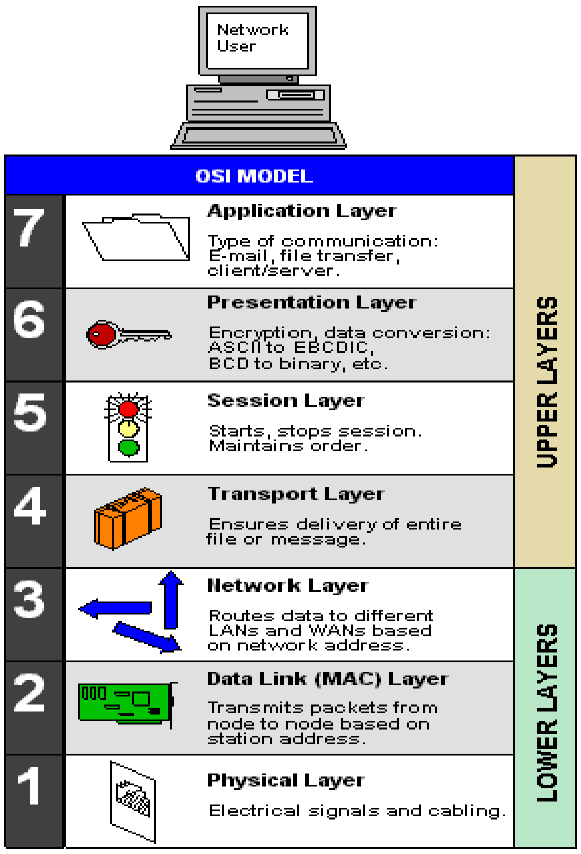
Network User
OSI MODEL
7 Application Layer
Type of communication: E-mail, file transfer, client/server.
6 Presentation Layer
Encryption, data conversion: ASCII to EBCDIC, BCD to binary, etc.
5 Session Layer
Starts, stops session. Maintains order.
4 Transport Layer
Ensures delivery of entire file or message.
3 Network Layer
Routes data to different LANs and WANs based on network address.
2 Data Link (MAC) Layer
Transmits packets from node to node based on station address.
1 Physical Layer
Electrical signals and cabling.
UPPER LAYERS
LOWER LAYERS

Figure 6: ISO Model For Describing A Network Protocol Stack.

2. RELATED STUDIES

There are many previous works studied and analyzed TCP performance. N. A. E. Farah,

A. Yasir and Y. Elhadi in [1] investigate the impact of reducing the number of duplicated acknowledgments in fast retransmission method to be two rather than triple. They captured various statistics like drop rate and latency through many scenarios. Their results mention that by reducing duplicated acknowledgments threshold TCP throughput is better than TCP Reno but not good enough when compared by TCP new Reno.

A study, focused specifically on updating the retransmission timeout (RTO) calculation, is introduced in [2] where the authors aim to provide more adaptive timeout value. They take in consideration the properties of wireless ad hoc environment to study TCP. In such environment, TCP faces many factors that degrade it's performance. The modification made timeout value affected by the deviation value divided by weighted congestion window (cwnd) value and added to estimated round trip time (ERTT). The proposed equation tested and compared with original calculation in regards to number of nodes. They found that their modification maintained a relatively good level with doubling the number of nodes each time to consider their work as improving the retransmission time over ad hoc network.

Another research focused on studying TCP performance on wireless network. The Authors in [3] analyzed and compared between TCP Reno and TCP new Reno performance over Ad hoc network. They aim to study TCP for congestion problem under DSDV routing protocol. They carried out different experiments to determine which is more suitable Reno or new Reno for mobile adhoc environment. The authors in their work mention that TCP Reno has a good performance when significant amount of packets been lost, but Reno behavior becomes worse with increasing loss. They conclude from their experiments That TCP New Reno is the most suitable transfer protocol in such rich challenges environment.

$$\boxed{\text{NSTATE} = \begin{cases} \text{true, newRTO} <= \text{oldRTO} \\ \text{false, otherwise} \end{cases}}$$

another modification on RTO calculation presented in [4]. The proposed adaptive algorithm, called TTAF, relays on fast retransmission method by computing the difference between the recorded time of receiving the third duplicated acknowledgement and the end of RTO period. The computed difference is subtracted from timeout value to compute a new adaptive RTO value. Their results show that TTAF achieves better than TCP new reno in regards to throughput with mobility and bit error rate existence.

Authors in [15] proposed another way to improve TCP performance over MANET. They suggests an dependant channel for sending acknowledgment packets that contains no data. these packets will be sent as highest priority to make it reach in very small period of time or immediately. the suggested approach reduced delay and in hence improve the overall network performance. number of hops, ratio of congestion and transmission rates are measured to promote their approach.

Allman, Mark, Avrachenkov, K , Ayesta, U & , J.Blanton & Hurtig in [6] proposed a solution for triggering fast retransmit when congestion window is small. they suggest to lowering the threshold of duplicated acknowledgments that required ti invoke fast retransmit to two rather than three. they suppose that in case of small congestion window there are no enough number of packets to generate duplicated acknowledgments. besides that, they found that TCP SACK variant (selective acknowledgments) is preferred to use for facing the loss of acknowledgment packets.

Another work [7] presented by Dongkyun Kim, C.-K. Toh, and Yanghee Choi for enhancing TCP performance over MANET. Their work allows each node in MANET network to buffering packets when route is disconnected. buffered packets are retransmitted later. They proposed a way to determine packet loss reason using two control messages, ERDN ("Explicit route disconnect notification") generated and sent back to sender when inner node (hop) between source and destination discovers route failure. the sender in its turn stop transmission until the ERSN ("Explicit route successful notification") arrives. using ERDN and ERSN makes distinguish between network congestion and route failure. TCP-BUS assigns double timeout value for buffered packets. Authors mentioned that their approach overcomes other approaches especially in term of throughput with mobility existence.

Mesut Gunes and Donald Vlahovic proposed "the restricted congestion window enlargement (TCP/RCWE)" in [1]. their work inspired from TCP/ELFN [9]. TCP/RCWE focused to solve TCP problems over ad hoc network. Firstly, they seek to make TCP able to distinguish between high bit error rate and link failure. They suggest to make congestion window adaptive with network state. To achieve this goal, a basic network estimator, called NSTATE, is used. NSTATE is a boolean variable takes either true or false value. it is assigned to false when the new retransmission timeout (RTO) value is greater than previous RTO value. As a response, it keeps congestion window value without changing. on another hand if new RTO is not greater than previous RTO, it is assigned to true and congestion window size is increased. following equation explains that.

All previous researches, which discussed above, aim to improve the performance of TCP when applied to MANET. But none of them used the difference in time between two successive duplicated acknowledgments before fast retransmit event. Our proposed work should improve TCP New Reno performance through make TCP able to sense network state and consequently compute retransmission timeout in accurate adaptive manner. The difference between two successive duplicated acknowledgments will be used as an indicator of network state.

3. AT-ER ALGORITHM

AT-ER comes to achieve the following objectives and requirements:

First, Make TCP more adaptive with network state: TCP interprets any loss happed as a consequence of congestion. This wrong assumption affects negatively on TCP performance because TCP will invoke TCP congestion avoidance algorithms which degrade data transmission and causing more delay.

To achieve this point AT-ER based on fast retransmit method to use elapse time between receiving ACKs as indicator. The value of difference between time between first and second ACKs gives TCP

Secondly, solve a case when window size is small and insufficient to invoke fast retransmit: fast retransmit mechanism in new Reno retransmits the lost packet after the receiving three of dupAcks. In some cases, like small congestion window, returning three successive dupAcks seems hard. To solve such case, we reduce the required number of dupAcks to two rather than three.

In last and as expected result, Improving TCP performance over MANET: As a consequence of satisfying of two objectives above the TCP performance should be improved.

Jacobson and Karle's timeout equation inherently designed to solve congestion problem in wired network. With the fast - paced digitization of the world including the growth of wireless networks, a need for update equation in (5) rises. Our study modifies the timeout calculation depending on fast retransmit mechanism that used in TCP new Reno. Fast retransmit helps TCP to distinguish the reason of loss from congestion. While returning multiple duplicated Acks (dupAcks) means that the network is not congested but the loss happens due to other causes such as (link failures, high bit error rate,etc..), the time between the duplicated Acks must be taken into account also. The long period between receiving first and second duplicated Acks means that the network is not in congestion, but it trends to be soon. While short period between 1st and 2nds dupAcks definitely means that no congestion and network in perfect state. In this study we record the time difference between the successive Acks, only the first and second Acks, and then reflect this difference on timeout calculation. Equations below explain:

$$D = |ACK\_02\_time - ACK\_01\_time| \quad (8)$$

where ACK_01_time means the recorded receiving time of first duplicated acknowledgment while ACK_02_time means the recorded receiving time of second duplicated acknowledgment.

$$newTimeout=\mu*ERTT+\theta*Deviation - (D/\sigma) \quad (9)$$

where σ is the adaptive variable that its value ranged between [2,4].

we subtract the time difference between the two acks from the timeout to detraction the waiting period time when the timeout event is unavoidable. Furthermore, fast retransmit mechanism in new Reno retransmits the lost packet after the number of dupAcks reaches three. In some cases like small congestion window, returning three successive dupAcks seems hard. To solve such case, we reduce the required number of dupAcks to two rather than three.

The following pseudo code explains :

```
1. set duplication threshold to 2
2. recording arrival time of first duplicated acknowledgment and save in ACK_01_time variable
3. recording arrival time of second duplicated acknowledgment save in ACK_02_time variable
4. compute the absolute difference time between ACK_02_time and ACK_01_time and save in D variable
5. subtract the half of D from pervious timeout value.
```

σ value plays main role in lowering waiting time. It is assigned to adapt with the status of network (congested, light congested, and not congested). Our approach measures whether network is crowded or not by the difference time. The second pseudo code simply explains:

```
1. Compute the absolute difference time between ACK_02_time and ACK_01_time and save in D variable
2. IF double of D value is still smaller than current timeout value -> no congestion , then assign σ to 2
3. Else if double of D is greater or equal than current timeout value -> congestion could happen soon, then assign σ to 4
4. Else if D is greater than current timeout -> congestion happened, then assign D to zero.
```

It should be clear now that multiplying difference by 2 and comparing it with current timeout value gives us a predicate about the status of network. If the duplication is still lower than current timeout value that means two things: first, the time difference between arriving first duplicated Ack and second duplicated is too small. This means the Acks returned so fast and the network is in a great state in which there is no congestion. Second, the value of waiting time (timeout value) is baggy and could cause delay when timeout event is unavoidable. A suitable response for this situation is subtracting only the half of Difference (σ = 2) from current timeout. It is necessary to subtract the half not the whole because subtracting the whole value exposes timeout value not to be adequate to absorb network delay which could happen by many circumstances such as link breakage or reroute of packets.

Second case when the double of time difference is greater or more equal than current timeout value in which the second duplicated Acks arrived after considerable time. This gives a predicate that the network is not seriously congested since the Acks back, but the congestion could happen soon. Since the network trends to congestion, the suitable response subtracts only the quarter of current time. subtracting this small value reduces waiting time and gives opportunity to the network to deal with congestion that could happen or not. Final case compares the value of difference with timeout value. If (D) is greater than current timeout, congestion will happen with high probability in which it is so soon to happen. The network needs more time to deal with its state, so timeout value should not be reduced, Different value is assigned to zero and timeout is calculated by original equations if it is not happened yet. Otherwise, the timeout event definitely happened and timeout value will be doubled as usual.

5. SETUP AND CONFIGURATION OF SCENARIOS (SIMULATION AND MANET SETTING)

We have used a discrete event simulation to emulate wireless ad hoc network. Table 1 shows the Characteristics of the environment during the study. we execute a large number of experiments with different seed each time and by changing the number of nodes from (10 to 100). also we executed experiments and record the results by changing the number of packet to send from 1000 to 9000 when the number of nodes is 40.

Table 1: MANET Simulation Settings

Property	value
SIMULATION-TIME	600S
TERRAIN - DIMENSIONS	(1200, 1200)
NODE-PLACEMENT	RANDOM
MOBILITY -MIN-SPEED	1 meter/sec
MOBILITY -MAX-SPEED	5 meter/sec
MAC-PROTOCOL	IEEE 802.11
NETWORK-PROTOCOL	IP
ROUTING-PROTOCOL	AODV
TCP VERSION	TCP New Reno
NUMBER-OF-NODES	10, 40, 70 and 100
NUMBER-OF-PACKETS-TO-SEND	1000,3000,6000,9000

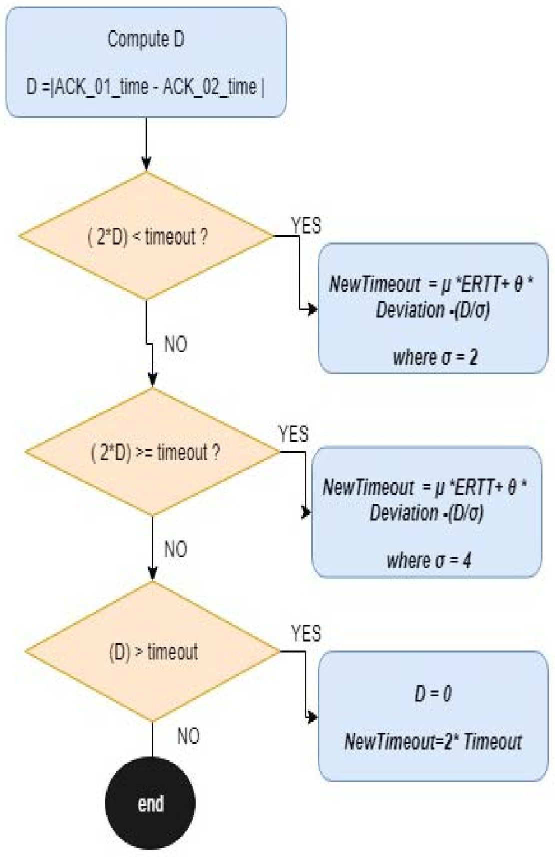

Figure 7: Flow Chart Of Proposed Algorithm

6. PROPOSED EQUATIONS AND RESULTS

Our proposed equation (9) for computing timeout is:

$$\text{Timeout} = \mu * \text{ERTT} + \theta * \text{Deviation} - (D/\sigma)$$

Where the value if σ determined based on the network status which sensed using the difference time between first duplicated acks and second duplicated acks. The flow chart in Figure 7 shows that σ takes the value of 2 when the network is far from congestion, while σ is set to 4 when congestion could happen soon. The difference of time between duplicated acks will be zero if the network is in a congestion, because the timeout already happened and cannot be avoided.

We conduct many experiments to test our approach by many parameters, such as Throughput, number of timeout events and packets ratio. while throughput provides a good indication about proposed approach, since it computes the amount of transmitted data in a time of unit, ratio of packets increases the results. Packet delivery ratio (PDR) is the ratio between the amount of packets transmitted and the amount of packets received. We also count the number of timeouts events during simulation experiments.

Figures 8,9,10 and 11 show the most important results have been gained.

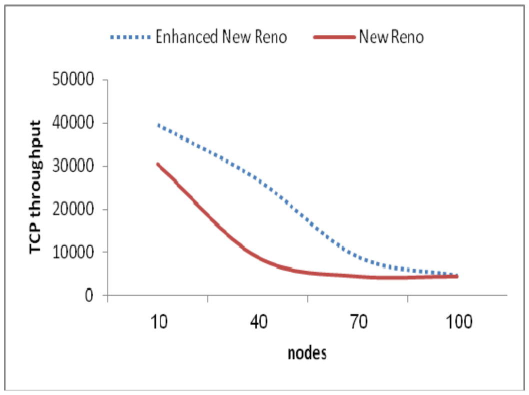

Figure 8: TCP Throughput Versus Number Of Nodes

Figure 8 illustrates the relationship between TCP throughput and the number of nodes. It can be seen that TCP throughput falls while the number of nodes rises. In other words, while the number of nodes increased, TCP throughput decreased in which it is affected by TCP new Reno. One of the main reason for performance degradation of TCP over MANET is increasing the loss of packets due to link breakages, fading, handoff or high bit error rate. We can note from figure 8 that TCP AT-ER deals better than TCP new Reno in MANET environment.

AT-ER enhances throughput by decreasing the waiting time after receiving the second duplicated ack. Moreover, figure 9 observes that number of timeout raised gradually by increasing the number of nodes. TCP AT-ER's behavior is same of TCP new Reno with lower number of timeout events that because AT-ER can sense the network state to assign adaptive value of timeout.

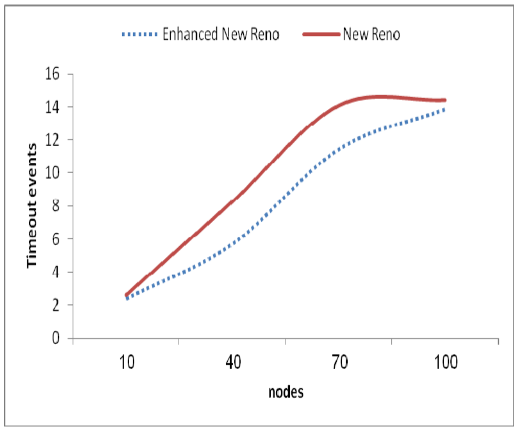

Figure 9: Timeout Events Versus Number Of Nodes

Another aspect of the performance is packet delivery ratio (PDR) which can give us a specification around packet delivery loss. Figure 10 observes how AT-ER enhances PDR of TCP new reno.

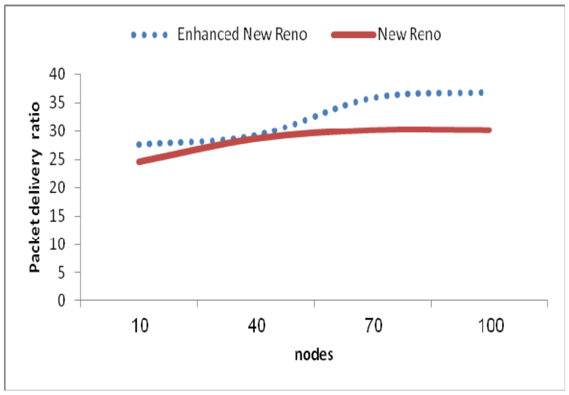

Figure 10: Packet Delivery Ratio Versus Number Of Nodes.

Figure 11 observes the superiority of AT-ER over new Reno in regards to throughput when number of packets to send increased from 1000,3000,6000 and 9000 when the nodes is 40.

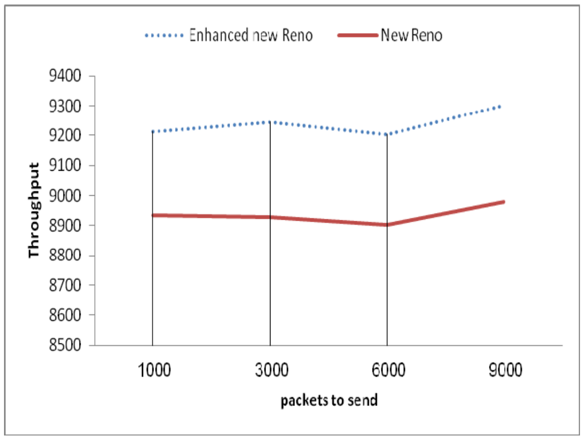

Figure 11: Throughput Versus Number Of Packets To Send

In regards to mobility, Figures 12 and Figure 13 compare between TCP AT-ER and New Reno with mobility existence when number of nodes is 70 . it is notable that TCP AT-ER more qualified with MANET dynamicity. these experiments conducted based on setting showed below in Table 2.

Table 2. Mobility Experiments Settings

Parameter	value
Number of nodes	70
Packet to send	3000 packets
Mobility speeds	5,10,15 and 20 m/sec

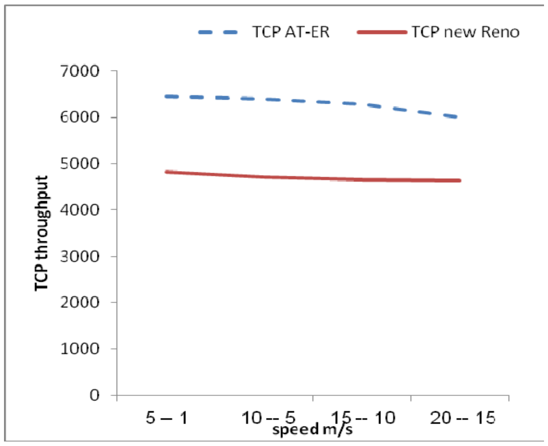

Figure 12 : TCP Throughput Versus Mobility.

Figure 12 above compares between the throughput of TCP New Reno and TCP AT-ER while mobility speeds vary between 1 to 20 meter per second. AT-ER provide higher throughput than New Reno especially when speed range between 1 to 10 m /sec. this comes as a consequence of reducing number of timeout events, as Figure 13 below shows, which affect on throughput by reducing the cwnd each time to be only one segment and doubling the RTO value.

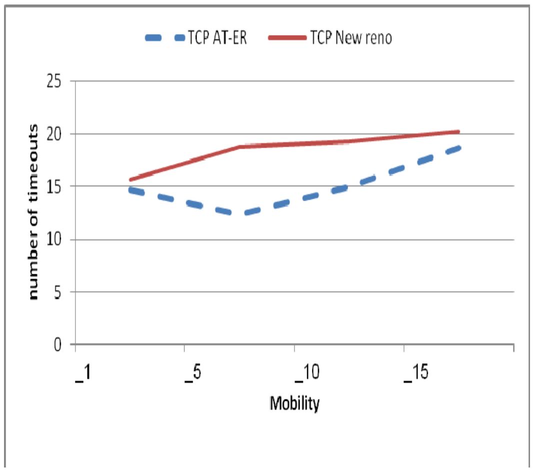

Figure 13 : Timeout Event When Number Of Nodes Is 70 And Packet To Send 3000 Versus Mobility.

The gained results from all experiments prove that AT-ER method makes TCP New Reno's performance better. It is clear from provided figures the superiority of AT-ER over new Reno in regards to throughput when number of packets to send is increasing.

TCP New Reno was in confusion through increasing mobility speeds, it interprets any loss as a congestion while it could have happened because link failures or out of range nodes. On another hand, it is notable that TCP AT-ER more qualified with MANET dynamicity.

7. CONCLUSION

This research proposes AT-ER algorithm which mainly aims to improve TCP New Reno performance over MANET. It comes to make TCP able to sense the network state and able to recognize the real reason of loss. Based on fast retransmit method, which invoked by receiving three duplicated acknowledgments back from destination through wireless media, the difference of time of receiving the first and second acknowledgment used as indicator from AT-ER to recognize network status. Based on this difference of elapsed time, a modification on timeout calculating equation is proposed to minimize wasted time in waiting. AT-ER uses only two duplicated acknowledgments for invoking fast retransmit in TCP New Reno. AT-ER solves two main problems by reducing duplicated acknowledgments threshold: firstly, when cnwnd size is small which making generating of three duplicated acks impossible. Secondly, TCP recognition of packet(s) loss reason.

Using discrete simulator and by analyzing results, we find that AT-ER provides accurate and adaptively retransmission timeout. It enhances TCP New Reno performance by 20% over Mobile Ad hoc network. This comes since New Reno with AT-ER algorithm becomes able to sense the state of network, in regards of congestion, through the calculated time difference. For future, we are looking to involve window size to be calculated also based on network state which makes the sender and receiver more aware about state of each other.

REFRENCES:

[1] GUNES, Mesut; VLAHOVIC, Donald. The performance of the TCP/RCWE enhancement for ad-hoc networks. In: Proceedings ISCC 2002 Seventh International Symposium on Computers and Communications. IEEE, 2002. p. 43-48.

[2] Krishan, Tareq T., Adnan M. Al-Smadi and Jordan. "A Novel Algorithm for TCP Timeout Calculation over Wireless Ad Hoc Networks." (2015).

[3] GUNANTARA, Nyoman; PUTRA, Nurweda; NYOMAN, I. Dewa. "The Characteristics of Metaheuristic Method in Selection of Path Pairs on Multicriteria Ad Hoc Networks". Journal of Computer Networks and Communications, 2019, 2019.

[4] Almobaideen, Wesam A. and Njoud O. Al-maitah. "TTAF: TCP Timeout Adaptivity Based on Fast Retransmit over MANET." JNW 9 (2014): 3321-3326.

[5] Stevens, W. Richard. "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms." RFC 2001 (1997): 1-6.

[6] Allman, Mark & Avrachenkov, K & Ayesta, U & , J.Blanton & Hurtig, Per. (2010). Early Retransmit for TCP and Stream Control Transmission Protocol (SCTP). Internet RFCs, ISSN 2070-1721, RFC 5827.

[7] KIM, Dongkyun; TOH, C.-K.; CHOI, Yanghee. TCP-BuS: Improving TCP performance in wireless ad hoc networks. Journal of Communications and Networks, 2001, 3.2: 1-12.

[8] N. A. E. Farah, A. Yasir and Y. Elhadi, "Investigation and Estimation of Fast Retransmission Method for Reliable Data Transmission in TCP," 2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), **2018**, pp. 1-5.

[9] MONKS, Jeffrey P.; SINHA, Prasun; BHARGHAVAN, Vaduvur. Limitations of TCP-ELFN for ad hoc networks. MOMUC'00, 2000.

[7] Peterson, L.L. and Davie, B.S., 2007. Computer networks: a systems approach. Elsevier.

[8] Loo, J. (Ed.), Lloret Mauri, J. (Ed.), Ortiz, J. (Ed.). (2012). Mobile Ad Hoc Networks. Boca Raton: CRC Press.

[9] Padhye, Jitendra, et al. "Modeling TCP Reno performance: a simple model and its empirical validation." IEEE/ACM Transactions on Networking (ToN) 8.2 (2000): 133-145.

[10] Al Hanbali, Ahmad, Eitan Altman, and Philippe Nain. "A survey of TCP over ad hoc networks." IEEE Communications Surveys and Tutorials 7.1-4 (2005): 22-36.

[11] Weniger, K. and Zitterbart, M., 2004. Mobile ad hoc networks-current approaches and future directions. IEEE network, 18(4), pp.6-11.

[12] Bhatt, Priyang. (2013). Performance Analysis of TCP variants in Wired and Wireless scenario. 2. 2321-613.

[13] ZAPATA, Manel Guerrero; ASOKAN, Nadarajah. "Securing ad hoc routing protocols". In: Proceedings of the 1st ACM workshop on Wireless security. ACM, 2002. p. 1-10.

[14] ROSE, Marshall T.; MCCLOGHRIE, Keith. Structure and Identification of Management Information for TCP/IP-based internets. 1990.

[15] SUNITHA, D.; NAGARAJU, A.; NARSIMHA, G. Cross-layer based Smart Acknowledgment Scheme for TCP in MANETs. International Journal of Computer Science and Network Security (IJCSNS), 2016, 16.7: 124.

[16] MAST, Noor; OWENS, Thomas J. A survey of performance enhancement of transmission control protocol (TCP) in wireless ad hoc networks. EURASIP Journal on Wireless Communications and Netw orking, 2011, 2011.1: 96.

[17] HAARTSEN, Jaap. Bluetooth-The universal radio interface for ad hoc, wireless connectivity. Ericsson review, 1998, 3.1: 110-117.

[18] LIU, Jian; SINGH, Suresh. ATCP: TCP for mobile ad hoc networks. IEEE Journal on selected areas in communications, 2001, 19.7: 1300-1315.

[19] K. Sasaki, M. Hanai, K. Miyazawa, A. Kobayashi, N. Oda and S. Yamaguchi, "TCP Fairness Among Modern TCP Congestion Control Algorithms Including TCP BBR," 2018 IEEE 7th International Conference on Cloud Networking (CloudNet), Tokyo, 2018, pp. 1-4.

[20] S. U. Shenoy, M. S. Kumari, U. K. K. Shenoy and N. Anusha, "Performance analysis of different TCP variants in wireless ad hoc

networks," 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, 2017, pp. 891-894.

[21] X. Sun, Y. Zhang, S. Zhang, X. Zhou, K. Niu and J. Lin, "HHS-TCP: A novel high-speed TCP based on hybrid congestion control," 5th IET International Conference on Wireless, Mobile and Multimedia Networks (ICWMMN 2013), Beijing, 2013, pp. 271-275.

[22] ROSE, Marshall T. The simple book: an introduction to management of TCP/IP-based internets. Englewood Cliffs, NJ;: Prentice Hall, 1991.

[23] FALL, Kevin; FLOYD, Sally. Simulation-based comparisons of Tahoe, Reno and SACK TCP. ACM SIGCOMM Computer Communication Review, 1996, 26.3: 5-21.

[24] KAZMI, Madiha, et al. Comparison of TCP tahoe, reno, new reno, sack and vegas in IP and MPLS networks under constant bit rate traffic. In: International Conference on Advanced Computational Technology and Creative Media (ICACTCM). 2014. p. 33-39.

[25] SHENOY, Sharada U.; KUMARI, Sharmila; SHENOY, Udaya Kumar K. Comparative Analysis of TCP Variants for Video Transmission Over Multi-hop Mobile Ad Hoc Networks. In: International Conference on Computer Networks and Communication Technologies. Springer, Singapore, 2019. p. 371-381.

[26] LAXMI, Vijay, et al. JellyFish attack: Analysis, detection and countermeasure in TCP-based MANET. Journal of Information Security and Applications, 2015, 22: 99-112.

[27] LIEN, Yao-Nan; YU, Yi-Fan. Hop-by-Hop TCP over MANET. In: 2008 IEEE Asia-Pacific Services Computing Conference. IEEE, 2008. p. 1150-1155.