Evaluating and Improving TCP Performance against Contention Losses in Multihop Ad Hoc Networks

Ehsan Hamadani, Veselin Rakocevic
Information Engineering Research Centre
School of Engineering and Mathematical Sciences
City University, London EC1V 0HB, UK
{E.Hamadani, V.Rakocevic}@city .ac.uk

Abstract- **Default TCP treats all losses as congestion since it was designed and optimized to work in networks with low non-congestion losses. In this paper first it is shown that the dominant cause of packet drops in multihop ad hoc network is medium access contention and congestion losses, are very rare. Then the most prevalent packet contention drops are analyzed and based on that a modification to both TCP and IEEE802.11 MAC is proposed and evaluated. It is also shown that this modification can eliminate TCP throughput instability which is regarded as one of main TCP problems in multihop networks.**

Keywords- **Contention drop, Hidden Terminal, IEEE802.11 MAC, Multihop Ad hoc Network, TCP**

I. Introduction

Wireless ad hoc networks are multihop wireless networks without any fixed infrastructure. An ad hoc network is formed solely by its terminals so that each terminal connected to the network provides also relaying service for others, i.e. acts as a router.

Transmission Control Protocol (TCP) [1] was designed to provide reliable end-to-end delivery of data over unreliable networks. In theory, TCP should be independent of the technology of the underlying infrastructure. In particular, TCP should not care whether it is running over wired or wireless connections. In practice, this does matter because most TCP parameters have been carefully optimized based on assumptions that are specific to wired networks. More precisely, in a multihop ad hoc networks, the main problem of TCP lies in treating all packet losses as network congestion and hence performing congestion control unnecessarily [2].

Similarly, the 802.11 MAC standard [3] was primarily designed and optimized for the infrastructure and single hop wireless networks.

Intuitively, the poor performance of TCP in multihop ad hoc networks results from the fact that TCP is unable to distinguish the reason of packet loss in wireless networks and runs its congestion control mechanism regardless of the cause of error. The more frequently the congestion control is called, the worst will be the throughput of TCP. However, as it has been revealed in [4], losses due to link-layer contention due to hidden terminals are the dominant source of packet drops in multihop networks.

In this paper, we first extend and confirm the results observed in [4] by considering also the flow of TCP acknowledgments (ACKs) in reverse direction in our simulation. Then the most prevalent scenarios which result to MAC contention packet drop and TCP congestion window drop are reviewed and based on that a new algorithm to reduce the number TCP congestion window drop as a result of contention packet drop will be proposed.

The rest of this paper is organized as follows: in section 2 and 3, we will give an overview of TCP and IEEE 802.11 MAC, respectively. In section 4, the main cause of packet contention drops which result to TCP retransmission is discussed. In section 5, the simulation model and key results obtained from it are given. Finally, in section 6, we conclude the paper with some outlines towards future work.

II. Description of TCP response to packet loss

To estimate the number of packets that can be in transit without causing congestion, TCP maintains a *congestion window (cwnd)*. New packets are only sent if allowed by both this window and the receiver's *advertised window*. The TCP sender can detect a packet loss in two ways: (1) a timeout, and (2) duplicate ACKs. A time-out occurs when the sender does not receive any acknowledgment from the receiver within a Retransmission Time Out (RTO). When a time-out occurs, TCP interprets this as severe congestion in the network and sets the *slow start threshold* (*ssthresh*) to 1/2 of the minimum of the current *cwnd* and the receiver's *advertised window*. Then it decreases the *cwnd* to 1 segment and performs a *slow start*. During the slow start, the TCP sender increases *cwnd* by 1 segment every time an ACK is received. When the *cwnd* reaches the *ssthresh*, it leaves the slow start phase and switches to the *congestion avoidance*, where new ACK increments the cwnd by 1/*cwnd*. On the other hand, when duplicate ACKs are received (duplicate ACKs signify that the receiver received out-of-order packets, TCP interprets this as less severe congestion. When the third duplicate ACK is received, TCP goes into the *fast retransmit.* The *ssthresh* is set to 1/2 of the minimum of

cwnd and the receiver's advertised window and the TCP sender retransmits the lost packet immediately without waiting for the retransmission time-out. During the fast retransmit stage, each duplicate ACK is considered as an acknowledgment of an out-of-order segment and the *cwnd is increased by one* correspondingly and continues to send packets.

III. Overview of IEEE 802.11 MAC standard

MAC layer in IEEE 802.11 offers two different types of service: a contention-based service provided by the Distributed Coordination Function (DCF) and an optional contention-free service implemented by the Point Coordination Function (PCF). Since PCF is based on polling and it should be done by a centralized coordinator, here we just describe DCF.

DCF provides the basic access method of the 802.11 MAC based on the CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) scheme to access the channel. According to this scheme, when a node receives a packet to be transmitted, it first listens to the channel to ensure no other node is transmitting. In order to detect the status of the medium, IEEE 802.11 performs carrier sensing at both the physical layer, referred to as the *physical carrier sensing* and at the MAC layer, referred to as the *virtual carrier sensing*. A virtual carrier-sensing mechanism is done via the use of two control packets (RTS/CTS) as follows: A node ready to transmit a packet, sends an RTS (Request to Send) packet towards the destination. All nodes that hear the RTS then update their Network Allocation Vector (NAV), which indicates the amount of time that must elapse until the current transmission session is complete and the channel can be sampled again for idle status. The destination, upon reception of the RTS responds with another short control packet CTS (Clear to Send). All nodes that hear the CTS packet also defer from accessing the channel for the duration of the current transmission. This means that, the channel is marked busy if either the physical or virtual carrier sensing mechanisms indicate the channel is busy.

The reception of the CTS packet at the transmitting node acknowledges that the RTS/CTS dialogue has been successful and the node starts the transmission of the actual data packet after a specified time, called the Distribution InterFrame Space (DIFS) and then transmits the packet. Otherwise, it chooses a random back-off value and retries later. Hereafter the node listens to the medium using carrier sensing to determine whether there is any activity in medium or not. Each time the medium is sensed to be idle, the backoff time is decremented. When the backoff counter reaches zero, the node waits for another DIFS and if the channel is still idle, it transmits the packet. If the receiver gets the packet without error, it initiates the transmission of MAC ACK (hereafter is called MACK to be distinguished from TCP ACK) after a time interval called Short InterFrame Space (SIFS) towards the sender. The SIFS is shorter than DIFS in order to give priority to the receiving station to over other possible stations waiting for transmission. If the MACK (or CTS when we have sent RTS) is not received at the sender within a certain time, the data frame is presumed to have been lost and a retransmission is scheduled if allowed by maximum retransmission counter called MAC_Retry_Limit. In fact there are two MAC_Retry_Limits: Short_Retry_Limit which is used for short packets (e.g. RTS, CTS), and Long_Retry_Limit for long packets (e.g. TCP DATA). If the packet cannot be sent successfully to the next hop within these limits, it would be dropped and the MAC repeats the same procedure to send the next frame in its buffer queue. If consecutive packet(s) experience the same problem, TCP timeout will happen. However, if some of consecutive packets can get through and reach the destination, a duplicate ACK would be sent towards sender. This leads to triggering TCP Fast Retransmit before timeout happens.

IV. Problem definition

MAC protocols have been shown to significantly affect TCP performance [5, 6]. It is shown that hidden and the exposed terminal problem [6] are two major reasons for preventing one node from reaching the other when the two nodes are in each other's transmission range. In [9] simulation results indicate that TCP throughput decreases exponentially as the number of hops due to the hidden terminals problem, which increases the packet collision. Similar problems were evaluated in [7] where the authors show that using smaller values for both *packet size* and *maximum window size* in TCP setup can mitigate such problems to some extent. As mentioned in the last section, if a node cannot reach its adjacent node within MAC_Retry_Limit, it will drop the packet and trigger a route failure, which in turn will cause the source node to start route discovery. Before a new route is found, no data packet can be sent out. During this process, generally TCP sender timeout and invokes its congestion control algorithm. Serious oscillation in TCP throughput will thus be observed.

The objective of our work is to minimize the probability of incorrect triggering of the congestion control window reduction in TCP. For that, an in-depth detail of the scenarios which result in congestion window reduction as a result of contention packet drop in a chain topology are reviewed in this section. Here, we assume RTS/CTS handshake is used only for TCP DATA packets and not for TCP-ACK. Regarding IEEE802.11 MAC timing specification, the DCF protocol ensures that CTS frame transmission will be successful if the RTS frame is

successfully delivered to the destination. This is because successful RTS frame transmission silences all the nodes in the neighborhood of the source either for a duration specified in the duration field of the RTS or for EIFS (Extended InterFrame Space) time (if collision occurs) which is large enough to transmit a CTS. Therefore, the CTS frame cannot collide with any frame at the source. Using a similar argument, it can be concluded that a successful TCP DATA frame transmission ensures a successful MACK frame transmission. Thus, we cannot have CTS and MACK frames drop because of medium contention. However, there can be RTS, TCP DATA, and TCP-ACK frame drop according to the presence of hidden terminals in the multi-hop scenario. Section *a*, *b*, and *c* reviews the scenarios in which these packets can be dropped in a chain topology. Here, node A is sending TCP DATA flow to node G using 5 intermediate nodes (nodes B to F). The transmission range of nodes is shown by a circle around them and as an initial value, all stations are assumed to have an NULL NAV field.

a) RTS drop:

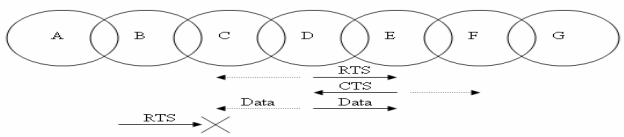

Figure 1- TCP-DATA drop due to RTS drop

1) Station D has TCP DATA to send to E. Therefore, it initiates RTS transmission towards node E
2) Station E transmits CTS back to D and consequently D starts transmitting TCP DATA to E
3) Meanwhile B has a TCP DATA to send to C, thus starting its own RTS handshake.
4) Due to ongoing TCP DATA transmission between D and E, the B's RTS is dropped at C.
5) B resends the RTS after an exponential backoff; however due to large size of TCP DATA, in most of the cases all RTS retransmissions (7 by default) are collided at C, resulting in TCP DATA drop at B.

b) TCP-ACK drop:

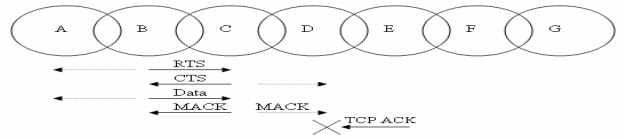

Figure 2- TCP-ACK drop

1) Station B has TCP DATA to send to C. So, it starts initiating RTS transmission.
2) Station C transmits CTS back to B and consequently B starts transmitting TCP DATA to C
3) Station B sends an MACK at the end of successful reception of TCP DATA.
4) Meanwhile TCP ACK is flowing back from E to D since E is not aware of ongoing transmission between B and C.
5) Collision happens at D between TCP-ACK and unintended MACK received by D. TCP-ACK is dropped.

c) TCP DATA drop:

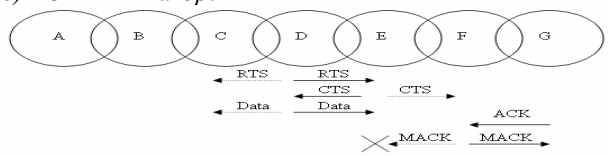

Figure 3- TCP-DATA drop

1) Station D has TCP DATA to send to E. so, it starts initiating RTS transmission.
2) Station E transmits CTS back to D and consequently D starts transmitting TCP DATA to E
3) Meanwhile station G sends a TCP ACK to F.
4) Upon reception of TCP ACK, F triggers an MACK transmission to G. Note that in general F can not send anything as it has updated its NAV by D's CTS. However, regarding 802.11 standard, the MACK should be send immediately irrespective of NAV duration.
5) Collision happens at E between the transmitted MACK and ongoing TCP DATA. So TCP DATA will be dropped spite of successful RTS/CTS transmission and need to be retransmitted.

V. Simulation Model and Results

A) Simulation Model

To validate the analysis given in the previous section, we use the same network topology in OPNET simulator. The distance between two neighbors is 80m while the transmission range is set to 100m according to the 802.11b testbed measurements presented in [8]. In all scenarios, the application operates in asymptotic condition (i.e., it always has packets ready for transmission) and scheduling of packet transmission is FIFO. Nodes use DSR as the routing protocol. TCP New Reno flavor is deployed since as reported in [10], it is currently the most common TCP version used in the Internet. TCP packet size is assumed to be fixed at 1460B and the TCP advertised window is set to infinity so the receiver buffer size does not affect the TCP congestion window size. In physical layer Direct Sequence Spread Spectrum (DSSS) technology with 2Mbps data rate is adopted. The channel uses free-space with no external noise and each node has a 20 packet MAC layer buffer pool. RTS/CTS message exchange is used for packets larger than 256B. This means that for TCP ACKs, no RTS/CTS exchange is done due to the large overhead they impose to system. The number of retransmission at MAC layer is set to 4 for packets greater than 256B (Long_Retry_Limit) and 7

for other packets (Short_ Retry_Limit) as has been specified in IEEE 802.11 MAC standard [3]. Three scenarios have been implemented and tested in the simulation. In scenario 1, the current version of 802.11 MAC and TCP New Reno is used. In scenario 2, an extended version of recent and promising modification to TCP for single hop wireless communication called TCP-DCR [11] is used. The basic idea behind DCR is when both congestion and non congestion losses can occur, a simple solution would be to let the link layer mechanisms recover from the losses due to transmission errors and the transport protocol to recover from the losses due to congestion. In other words, the TCP-DCR modifications aim to change the time at which the Fast Retransmit/Recovery algorithms are triggered. Therefore, in TCP-DCR the receipt of duplicate ACKs is assumed to be caused by non-congestion errors, for a bounded delay period (one RTT). If within this time the packet is recovered by the link level retransmission, TCP would continue its normal operation. However, if the packet could not be recovered by the end of that time, the packet would be recovered through Fast Retransmit/Recovery. In Scenario 2, an extended DCR (E-DCR) implementation for multihop ad hoc network is implemented. Intuitively, in E-DCR approach if the MAC can not succeed in accessing the medium in limited tries at any hop, the packet should be retransmitted by TCP from the sender and start contending for the medium from the first hop again. Obviously, this should results into more packet contention during simulation. However, as showed in the last section, since the contention drop is the most dominant cause of packet drop in multihop networks, we expect poor performance of E-DCR in multihop ad hoc network.

To alleviate the above problem, a new method is proposed and implemented in scenario 3. In this approach, the Fast Retransmit/Recovery is delayed for one RTT as in scenario 2. However, in the event of unsuccessful packet transmission more link level retransmission is allowed to be performed by that hop. More precisely, after reaching the specified MAC_ Limit_Retry, the node is given the chance to try for another MAC_Limit_Retry before giving up and dropping the packet. We expect this will result to less TCP retransmission as a higher chance is given to MAC to shield the packet contention drop from TCP.

B) Results:

First, in order to support our discussion about the main cause of packet drop in multihop ad hoc networks, the dropped packets were traced in the default TCP (scenario 1). During the simulation time on average, 2% of packets were dropped due to the contention and less than 0.05% packets were dropped because of congestion. This observation confirms the results presented in [4]. In addition, the model used here is more realistic than the model used in [4][1]. In the next step, the three mentioned scenarios are simulated and evaluated from TCP perspective. Figure 4 and 5 depict the total number of TCP timeout and TCP Fast Retransmit invocation, respectively.

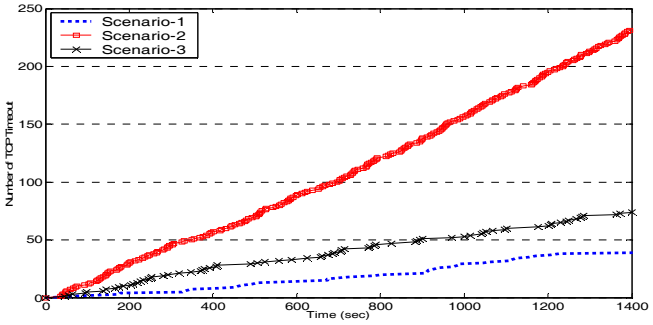

Figure 4- TCP Timeout

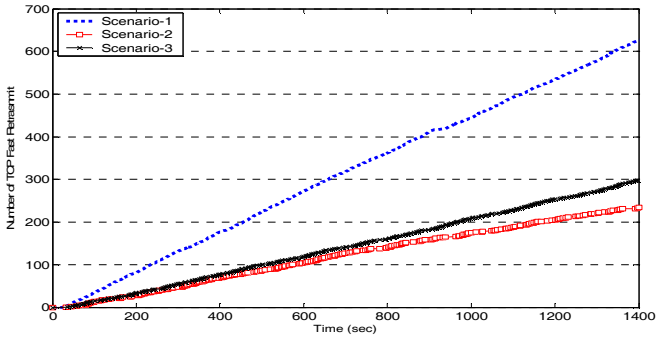

Figure 5- TCP Fast Retransmit

As it is clear from these graphs, in the default TCP most dropped packets due to medium contention are recovered through TCP Fast retransmit. The reason behind this is in the high sensitivity of the current TCP to duplicate ACKs. In other words, while setting the duplicate threshold to 3 in wired network is reasonable due to low probability of packet reordering, this threshold is very small in wireless networks where packet reordering can happen very frequently because of packet contention drops. On the other hand, decreasing the sensitivity of TCP to duplicate ACKs in TCP-DCR approach can bring a very serious and severe problem to TCP performance. As shown in figure 5, though the number of TCP fast retransmit has decreased in E-DCR comparing to default TCP, many TCP timeouts is occurring. This mostly happens due to wrong retransmission timing between TCP and MAC. Indeed, in E-DCR in most of the cases the TCP timeout is triggered earlier than TCP fast retransmit while the opposite happens in default TCP.

[1] - The contention drop ratio given in [4] is 1%. However, TCP ACK flows in reverse path have not been considered in deriving the results.

Figure 6 shows a comparison of achieved goodput. Clearly, scenario 2 is experiencing a poor TCP performance again regarding its large number of timeouts. Scenario 1 and 3 almost follow the same TCP goodput pattern while the TCP throughput instability that was reported in [6] and can be also observed in scenario 1 (for instance between time 900 to 930 s), has been eliminated in scenario 3 (figure 7). This is indeed due to less number of TCP congestion window drop as result of less TCP retransmissions (both from fast retransmit and timeout) in scenario 3 than other scenarios.

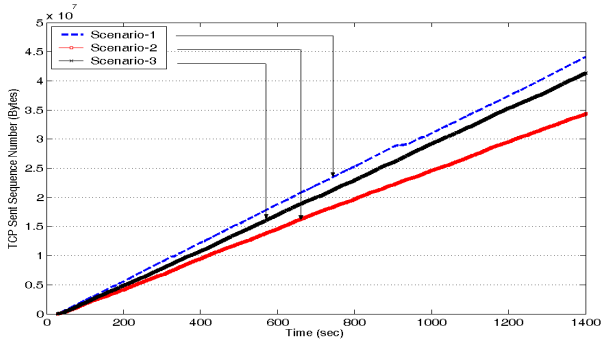

Figure 6 - TCP Goodput

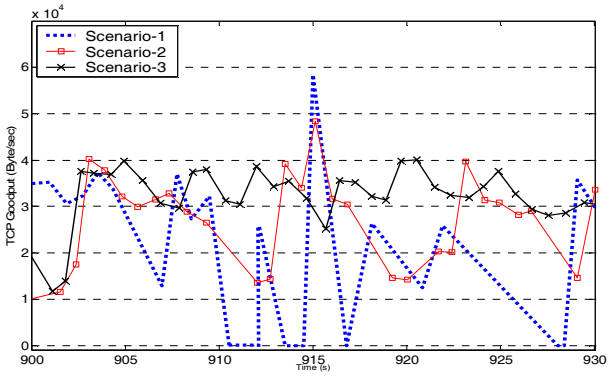

Figure 7- TCP Throughput instability

To also evaluate the effect of doubling the number of MAC retransmission on power consumption in other nodes, the total number of short and long retransmissions in all the nodes is given in table 1.

Table1- Number of Mac Retransmissions in the whole network

	Total Short Retransmission Retry	Total Long Retransmission Retry
Scenario 1	197140	28130
Scenario 2	183492	26031
Scenario 3	204285	29285

VI. Conclusion and future work

Default TCP treats all losses as congestion since it was designed and optimized to work in networks with low non-congestion losses. On the other hand, as showed here, the dominant cause of packet drops in multihop ad hoc network is medium access contention and congestion losses happen very rare. Consequently, most contention drops are misinterpreted as congestion causing congestion window reduction. More interestingly, the congestion window drop is mostly occurring because of Fast Retransmit and not Timeout. TCP-DCR which was designed and tested in single hop wireless communication, tries to alleviate this problem by postponing the congestion window reduction by one RTT. However, as discussed, this approach further deteriorates the TCP performance in multihop networks. To solve the problem, we first investigated the cause of packet drop contention in chain topology. Based on that, a new technique was proposed to overcome the large number of TCP packet retransmissions. The basic concept behind our proposal was that in order to minimize the packet contention drop, packets should be recovered at each hop rather than from TCP sender. It was shown that our modification has less number of TCP retransmissions and hence less incorrect triggering of the congestion control window reduction. This in turn resulted to more stable TCP throughput. However the main drawback of our scheme is that this might result to more number of MAC retransmission which leads to more power consumption in the network. We believe the fact that 802.11 uses a fixed number of retransmissions in all the hops for lost packets, is causing this problem. In future we will investigating how to adaptively change the number of MAC retransmissions according to cross layer information such as number of backoffs the node is experiencing. This effectively means that rather than using a similar and constant MAC_Limit_Retry in different nodes, more flexible and optimized method should be adopted in our approach to keep the total network power consumption low.

References:

[1] J. Postel, "Transmission Control Protocol," RFC 793, Sep.1981

[2] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," IEEE/ACM Trans. on Net, 1997.

[3] IEEE std. 802.11. "Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) specifications", 1999.

[4] Z. Fu, H. Luo, P. Zerfos, and M. Gerla, "The Impact of Multihop Wireless Channel on TCP Performance" IEEE Trans. on Mobile Comp,2005

[5] S. Xu and T. Saadawi, "Does the IEEE 802.11MAC protocol work well in multihop wireless ad hoc networks?" *IEEE Communications Magazine*, vol. 39, no. 6, pp.130-137, June 2001

[6] S. Xu and T. Sadawi, "Revealing the Problems with 802.11 MAC Protocol in Multihop Wireless Networks", Comp. Net., Vol. 38, 2002

[7] S. Xu, T. Saadawi and M. Lee. "On TCP over Wireless Multi-hop Networks" In proceedings of IEEE MILCOM2001, October 2001

[8] G. Anastasi, E.Borgia, M.Conti,E. Gregori, "IEEE 802.11b Ad Hoc Network:Performance Measurement",*Bechtel Telecom Technical Journal 2002*

[9] E. Altman and T. Jimenez, "Novel delayed ACK techniques for improving TCP performance in multihop wireless networks", in proc. of the Personal Wireless Communication, Italy, Sep2003, pp237-253

[10] A. Medina, M. Allman, and S. Floyd "Measuring the Evolution of Transport Protocols in the Internet", To appear in ACM CCR, April 2005

[11] S. Bhandarkar, N. Sadry, A. L. Reddy and N. Vaidya, "TCP-DCR: A novel protocol for tolerating wireless channel errors". IEEE Trans. on Mobile Computing, February 2004