Effect of Ad Hoc Routing Protocols on TCP Performance within MANETS

Alaa Seddik-Ghaleb
Networks & Multimedia Systems Research Group (LRSM), Institut d'Informatique d'Entreprise (IIE)
18 allée Jean Rostand, 91025 Evry, CEDEX - France.
seddik@iie.cnam.fr

Yacine Ghamri-Doudane
Networks & Multimedia Systems Research Group (LRSM), Institut d'Informatique d'Entreprise (IIE)
18 allée Jean Rostand, 91025 Evry, CEDEX - France.
ghamri@iie.cnam.fr

Sidi-Mohammed Senouci
France Telecom R&D
2 Av. Pierre Marzin, 22307,
Lannion, France
sidimohammed.senouci@francetelecom.com

Abstract- **TCP was mainly developed to be deployed within wired networks. Recently, many researches have studied its performance within Mobile Ad hoc Networks (MANETs). The research results found that TCP performance are highly influenced by the characteristics of such networks. This is due to the manner that TCP uses to implement reliability and its inability to distinguish between different packet loss reasons. Indeed, unlike wired networks, where packet loss is mainly caused by network congestion, in MANETs we could have many other reasons to lose a data packet. However, TCP considers that all packet losses are due to network congestion. This enforces TCP to be aggressive in front of certain types of packet loss. Packet losses in MANETs can be either related to the wireless communication environment (i.e. the effect of fading, interferences, multi-path routing, etc.) or to the dynamic nature of such networks (i.e. link failures, network partitioning). This latter could be due to the node mobility or to the node battery depletion. This could lead to frequent route re-computation within the network. In this work, we intend to study the effect of ad hoc routing protocols on TCP performance (energy consumption and average goodput[1]) within MANETs. We consider in our study different types of ad hoc routing protocols having different characteristics: reactive vs. proactive, distance vector vs. link state, and source routing. Our study results show that; DSDV as a proactive distance vector routing protocol leads to the most acceptable TCP performance results and this is confirmed at different mobility levels.**

1. Introduction

Wireless ad hoc networks have been proposed as the networking solution for those situations where the network set up time is a major constraint and/or a networking infrastructure is either not available or not desirable. Ad hoc networks allow mobile devices to exchange information using their wireless infrastructure without the need of a fixed infrastructure. Every device in wireless ad hoc networks takes the role of an end system, a server, a router, a gateway, etc., or all of them at the same time. It is expected that the performance of TCP will be affected considerably in ad hoc networks not only due to the effects of the wireless environment but also due to specific issues like mobility, routing, and energy constrains [1]. In mobile networks, such as MANETs, TCP displays some undesirable patterns of behavior in the context of efficient energy expenditure because of its reliability feature [2]. As an example, high channel delays in a mobile network, causing the TCP timer to expire, will force TCP to unnecessarily retransmit the delayed packet and to consume more time and energy resulting in network performance degradation. The performance of mobile ad hoc networks depends on many factors such as node mobility model, traffic pattern, network topology, obstacle positions, and so on. To better understand the effect of these factors, we classify them into two categories: those inherited from the wireless environment and others due to the mobile ad hoc characteristics itself. In a previous work [3], a complete analysis of the main TCP variants behavior has been realized in regards to the wireless environment characteristics. In the current work, our objective is to analyze the effect of the routing protocol on TCP performances in MANETs.

In the following, we present some of the most important issues related to the routing protocol in a mobile ad hoc network. When an old route is no longer available, the network layer at the sender attempts to find a new route to the destination. It is possible that discovering a new route may take significantly longer than the retransmission timeout interval (RTO) at the sender. As a result, the TCP sender times out, retransmits a packet and invokes congestion control. Thus, when a new route is discovered, the throughput will continue to be small for some time because TCP at the sender grows its congestion window using the slow start and congestion avoidance algorithms. This is clearly undesirable behavior because the TCP connection will be very inefficient. If we imagine a network in which route computations are done frequently (due to high mobility), the TCP connection will never get an opportunity to transmit at the maximum negotiated rate (the congestion window will always be significantly smaller than the advertised window size from the receiver) [4]. In addition, it is likely that the ad hoc network may periodically get partitioned for several seconds at a time. If the sender and the receiver of a TCP connection lie in different partitions, all the sender's packets get dropped by the network resulting in the sender invoking congestion control. If the partition lasts for a significant amount of time (say several times longer than the RTO), the situation gets even worse because of phenomena called serial timeouts. A serial timeout is a condition wherein multiple consecutive retransmissions of the same segment are invoked while the receiver is disconnected from the sender. All these retransmissions are thus lost. Since the retransmission timer at the sender is doubled with each

[1] The amount of data correctly received during a given time of period.

unsuccessful retransmission attempt (until it reaches 64 sec), several consecutive failures can lead to inactivity lasting for a while even when the sender and receiver get reconnected [4]. All the above situations can be a result of the dynamic nature of MANETs (i.e. nodes mobility or battery depletion). In order to study TCP performance over MANETs, it is important to analyze the effect of most common ad hoc routing protocols in such networks. Ad hoc routing protocols are responsible of re-establishing lost links within the network and reconnecting partitioned ones. Then, it is obvious that the behavior of these routing protocols will affect TCP performance. In this paper, we focus our attention on studying the impact of different ad hoc routing protocols on the energy efficiency as well as the obtained goodput of the six major TCP variants (Tahoe, Reno, New-Reno, SACK, Vegas, and WestwoodNR). More precisely, we will consider the effect of four ad hoc routing protocols those following different design principals. These protocols are: AODV (Ad-Hoc On-Demand Distance Vector) which is a reactive distance vector protocol, DSR (Dynamic Source Routing) a reactive source routing protocol, DSDV (Destination-Sequenced Distance Vector) a proactive distance vector protocol, and finally OLSR (Optimized Link State Routing) a proactive link state protocol.

The remainder of this paper is organized as follows: after presenting the motivation behind our current work in section 2, section 3 overviews each TCP variant as well as the different ad hoc routing protocols, section 4 presents TCP performance in terms of energy consumption and average goodput with respect to different ad hoc routing protocols. Finally, we summarize the main results of this work and give some ideas for future work.

2. Motivation

Recently, many researchers have studied TCP performance in terms of energy consumption and average goodput within wireless mobile networks [5][6][7]. It was proved that the performance of TCP degrades considerably in mobile ad hoc networks environment, due to the specific issues related to those environments [8]. Many research projects were specifically interested in studying TCP performances (energy consumption and/or goodput) within such environments [7][9]. However, none of them compared more than three TCP variants over a widest set of realistic scenarios. In this paper, we aim to make a clear comparison between the most common TCP variants. The behavior of ad hoc routing protocols in face of the dynamic nature of ad hoc networks can have an effect on these performances. Thus, this comparative study takes place using different ad hoc routing protocols. This study concentrates on analyzing the effects of these protocols whilst the mobility rate is varied. We make our simulations using a large number of nodes, in order to realize the effect of losing a non-adjacent node on both (i) the energy consumption of the other nodes in the network and (ii) the goodput obtained by TCP connections. The aim of this study is to help understanding the impact of the different ad hoc routing protocols and node mobility on TCP performance in MANETs. Thus, obtained results can be used as a guideline of the interactions between TCP and ad hoc routing protocols in MANETs.

3. Background

As our study relates to the main TCP variants and ad hoc routing algorithms these are described briefly in the following sub-sections:

3.1. Overview of TCP Variants

3.1.1. TCP Tahoe

TCP Tahoe is the first TCP variant that incorporates congestion control mechanisms. Indeed its implementation added a number of new algorithms and refinements to earlier implementations. Mainly these algorithms and refinements are: Slow-Start, Congestion Avoidance, and Fast Retransmit [10][11]. The goal of slow-start and congestion avoidance is to keep the congestion window[2] size around optimal size as much as possible. Slow-start increases the congestion window (CWND) size rapidly to reach maximum safety transfer rate (SSThresold) as fast as possible and congestion avoidance increases the CWND slowly to avoid packet losses as long as possible. If a packet is not acknowledged after a predefined timeout, Retransmission TimeOut (RTO), it is regarded as lost and is retransmitted. On the other hand, at the reception of three duplicate[3] ACKs, the first unacknowledged packet is also considered as lost. In this case, the Fast Retransmit algorithm is in charge of retransmitting the lost packet without waiting for the RTO. This latest speeds-up the lost packet retransmission. Finally, note that in both situations, the packet retransmission is followed by a reduction of both the SSThresh, to CWND/2, and the CWND to its minimum (CWND= 1 segment). The slow start phase is then triggered.

3.1.2. TCP Reno

The congestion control mechanisms of TCP Reno, the most popular TCP implementation, retains the enhancements incorporated into TCP Tahoe, but modifies the Fast Retransmit operation to include Fast Recovery [12]. The Slow-Start and the Congestion Avoidance algorithms are used by a TCP Reno sender to control the amount of data injected into the network while the Fast Retransmit and the Fast Recovery are used to recover from packet losses without the need for RTOs [13]. Fast Recovery algorithm reacts after a packet loss discovered by a three duplicate ACKs. Then it halves the congestion window instead of decreasing it to minimum as in TCP Tahoe.

3.1.3. TCP New-Reno

TCP New-Reno includes a small change to the Reno algorithm at the sender [14]. This change concerns the sender's behavior during Fast Recovery when a partial ACK is received. A partial ACK is the acknowledgment that can be received in response to a lost-packet retransmission. This one do not acknowledges all the packets that were outstanding at the start of the Fast Recovery period but acknowledges only some of them. This means that there are multiple losses in the same window of data. In TCP Reno, partial ACKs take TCP out of Fast Recovery by deflating the usable window back to the size

2 Maximum amount of data that can be sent out over a connection without being acknowledged.

3 TCP receivers generate a duplicate ACK when out-of-sequence segment is received.

of the congestion window. In TCP New-Reno, partial ACKs do not take TCP out of Fast Recovery. Instead, partial ACKs received during Fast Recovery are treated as an indication that the packet immediately following the acknowledged packet in the sequence space has been lost, and should be retransmitted. Thus, when multiple packets are lost from a single window of data, New-Reno can recover without a retransmission timeout, retransmitting one lost packet per round-trip time (RTT) until all of the lost packets from the window have been retransmitted. TCP New-Reno remains in Fast Recovery until all of the data outstanding when Fast Recovery was initiated has been acknowledged [13].

3.1.4. TCP SACK

Traditional implementations of TCP use an acknowledgement number field that contains a cumulative acknowledgement, indicating that the TCP receiver has received all of the data up to the indicated byte. A selective acknowledgement option allows receivers to additionally report non-sequential data they have received. The SACK option is used in an ACK packet to indicate which packets were received precisely [13] and thus allows to deduce which packets had been lost. This option aims to speed up the the retransmission of lost packets. This will in turn avoid retransmitting the whole window of data.

3.1.5. TCP WestwoodNR

TCP Westwood is a sender-side modification of the TCP congestion window algorithm that is intended to improve upon the performance of TCP New-Reno and TCP Reno in wired as well as wireless networks. In fact, there are two variants of TCP-Westwood, one is based on TCP Reno and the other is based on TCP New-Reno. Our study uses this latter (depicted as TCP WestwoodNR). The improvement is also targeted to be most significant in wireless networks with lossy links. Indeed, TCP WestwoodNR [15] relies on end-to-end bandwidth estimation to discriminate the cause of packet loss (congestion or wireless channel effect). This discrimination is based on measured RTT values.

3.1.6. TCP Vegas

TCP Vegas extends Reno's retransmission mechanisms. It relies on measured RTT values of sent packets. According to this measurement, Vegas may decide to retransmit a packet [16] before reaching RTO. When a duplicate ACK is received, Vegas checks to see if the difference between the current time and the timestamp recorded for the first un-acknowledged segment (i.e. its RTT) is greater than the timeout value. If it is, then Vegas retransmits the segment without having to wait for three duplicate ACKs. Also, TCP Vegas uses RTT values to calculate the actual transmission rate in the network. Hence, by comparing that value by the expected goodput in the network, TCP Vegas decides how to modify its transmission rate.

3.2. Overview of Ad Hoc Routing Protocols

3.2.1. DSDV

The Destination-Sequenced Distance-Vector Routing protocol (DSDV) described in [17] is a table-driven proactive algorithm based on the classical Bellman-Ford routing mechanism [18]. The improvements made to the Bellman-Ford algorithm include freedom from loops in routing tables. Every mobile node in the network maintains a routing table in which all of the possible destinations within the network and the number of hops to each destination are recorded. Each entry is marked with a sequence number assigned by the destination node. The sequence numbers enable the mobile nodes to distinguish stale routes from new ones, thereby avoiding the formation of routing loops. Routing table updates are periodically transmitted throughout the network in order to maintain table consistency. To help alleviate the potentially large amount of network traffic that such updates can generate, route updates can employ two possible types of packets. The first is known as a full dump. This type of packet carries all available routing information and can require multiple network protocol data units (NPDUs). During periods of occasional movement, these packets are transmitted infrequently. Smaller incremental packets are used to relay only that information which has changed since the last full dump. Each of these broadcasts should fit into a standard-size NPDU, thereby decreasing the amount of traffic generated. Every node keeps a route table (Destination-address, Metric, and Sequence-number) for every possible destination.

3.2.2. OLSR

The Optimized Link-State Routing Protocol (OLSR) is a proactive link-state routing protocol, employing periodic message exchange to update topological information in each node in the network, so the routes are always immediately available when needed. While having some commonalities with OSPF, OLSR is specially designed to operate in the context of ad hoc networks, i.e. in bandwidth-constrained, dynamic networks. The OLSR protocol applies an optimized flooding mechanism, called MPR-flooding, (Multipoint Relays) to minimize the problem of duplicate reception of message within a region. The MPR flooding mechanism is directly used by OLSR for diffusing topological information to the network [19].

3.2.3. AODV

The Ad Hoc On-Demand Distance Vector (AODV) routing protocol described in [20] builds on the DSDV algorithm previously described. AODV is an improvement on DSDV because it typically minimizes the number of required broadcasts by creating routes on an on-demand basis, as opposed to maintaining a complete list of routes as in the DSDV algorithm. The authors of AODV classify it as a pure on demand route acquisition system, since nodes that are not on a selected path do not maintain routing information or participate in routing table exchanges [20]. When a source node desires to send a message to some destination node and does not already have a valid route to that destination, it initiates a path discovery process to locate the other node. It broadcasts a route request (RREQ) packet to its neighbors, which then forward the request to their neighbors, and so on, until either the destination or an intermediate node with a "fresh enough" route to the destination is located. Once the RREQ reaches the destination or an intermediate node with a fresh enough route, the destination/intermediate node responds by unicasting a route reply (RREP) packet back to the neighbor from which it first received the RREQ. This RREP is than unicasted from neighbor to another until reaching the source node. Hence, all the nodes participating to the route will have the ability to update their routing tables accordingly.

3.2.4. DSR

The Dynamic Source Routing (DSR) protocol presented in [21] is an on-demand routing protocol that is based on the concept of source routing. Mobile nodes are required to maintain route caches that contain the source routes of which the mobile is aware. Entries in the route cache are continually updated as new routes are learned. The protocol consists of two major phases: route discovery and route maintenance. When a mobile node has a packet to send to some destination, it first consults its route cache to determine whether it already has a route to the destination. If it has an unexpired route to the destination, it will use this route to send the packet. On the other hand, if the node does not have such a route, it initiates route discovery by broadcasting a route request packet. This route request contains the address of the destination, along with the source node's address and a unique identification number. Each node receiving the packet checks whether it knows of a route to the destination. If it does not, it adds its own address to the route record of the packet and then forwards the packet along its outgoing links. A route reply is generated when the route request reaches either the destination itself, or an intermediate node which contains in its route cache an unexpired route to the destination [22]. DSR has the advantage that no routing tables must be kept to route a given packet, since the entire route is contained in the packet header. The caching of any initiated or overheard routing data can significantly reduce the number of control messages being sent, reducing overhead. The primary disadvantages of DSR are that it is not scalable to large networks, and that it requires significantly more processing resources than most other protocols. In order to obtain routing information, each node must spend much more time processing any control data it receives, even if it is not the intended recipient [23]. DSR does not use any periodic routing advertisements, link status sensing, or neighbor detection packets. Also, it does not rely on these functions from any underlying protocols in the network.

4. Ad Hoc Routing Protocols' Effects On TCP Performance In Manet

In this section we study the performance of different TCP variants in terms of energy consumption and average goodput within wireless mobile ad hoc networks. We take into consideration the effect of different ad hoc routing protocols by implementing the most common ones and analyzing TCP performance regarding to the behavior of each ad hoc routing protocol when this one have to recover a link failure and how this action could affect TCP performance. We study the performances regarding two common factors that may affect TCP performances within such environments: (i) the choice of ad routing protocol (ii) the nodes' mobility rate.

4.1. Simulation Results

Our simulations are realized using the Network Simulator version 2 (NS-2) [24]. Each simulation consists of a network of 20 nodes confined in a (670 x 670) m² area. These nodes are randomly positioned in the simulation area. 14 TCP connections were established (ftp traffic used with a packet size of 512 bytes) between the nodes. The source-destination pairs for FTP sessions were chosen randomly. They are shown in Figure 1.

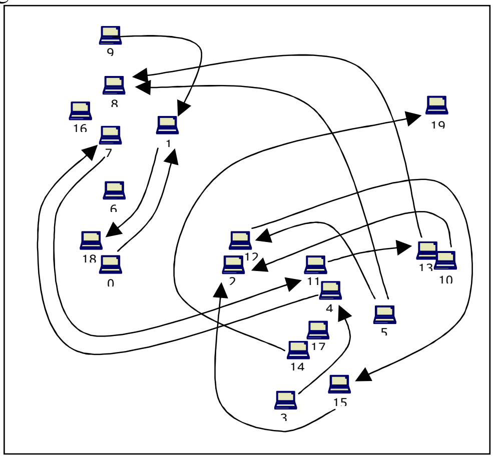

Figure 1. Network topology and FTP session.

The simulation time is set to 400 seconds. The initial battery capacity of each node is 10 joules. This initial energy is reduced progressively by data transmission and reception (which includes also the retransmission and forwarding). We consider the simple case where the transmission and reception of a packet consumes a fixed amount of energy from the node's battery. We fix the amount of energy to be consumed per each transmitted packet at 0.6 watt, and that for each received packet at 0.3 watt. When the initial energy reaches zero joules, the corresponding node cannot take part anymore in the communication, and is regarded as dead. Note that a node death can lead to routes reorganizations in the network. To study the effect of different routing algorithms on TCP performances, we aim to study both reactive and proactive routing protocols. In order to study the effect of nodes' mobility in MANETs on TCP performance, we use three different values of mobility rates (5m/s, 15m/s, and 30m/s). All nodes communicate with identical wireless radios using the standard MAC 802.11 which have a bandwidth of 2Mbps and a radio propagation range of 250 meters. In this work, we study three TCP performance parameters: the first one is the energy consumed in transmission, reception, forwarding and retransmission of packets. This energy is calculated proportionally to the amount of received data. Thus, it is defined as energy consumed per received bit. The second one is the average connection duration of TCP sessions. Note that, it was demonstrated in the literature [25] that this connection duration is proportional to the energy consumed at each node listening to the radio channel plus that consumed to execute the recovery mechanisms associated to each TCP (Timeouts, CWND threshold adjustments, etc.). This sum is called idle energy in the following. The third parameter studied is the average TCP goodput.

4.2. Effect of Routing Protocols

4.2.1. TCP Performances at 5m/s mobility rate

Figure 2 shows that, at 5m/s mobility rate, proactive protocols help TCP to have best performance in terms of energy consumption per received bit compared to reactive routing protocols. The fact that proactive protocols include all the available routes towards any destination in the network into routing tables, helps reducing the route recovery time spent in the network. We must note here that, the longer the route recovery time is the most probable that TCP sender's RTO timer expires. That enforces triggering TCP congestion control algorithm. Thus, decreasing the CWND, and entering "slow-start" phase. On the other hand, reactive protocols do not start a route discovery process unless it's needed by the source node. That may lead to higher latency in route recovery process. In other words, TCP sender's RTO timer might expire before that the routing protocol can recover from lost link. Here, TCP sender will not be able to know about the route recovery and it may stay in the "back off" state even that there is a recovered route towards the destination. Moreover, the case may become worse if the route recovery process longs for more than RTO timer. TCP enters into a "serial timeouts phenomena". A serial time out is a condition wherein multiple consecutive retransmissions of the same segment are transmitted to the receiver while it is disconnected from the sender. All these retransmissions are thus lost. Since the retransmission timer at the sender is doubled with each unsuccessful retransmission attempt (until it reaches 64 sec).

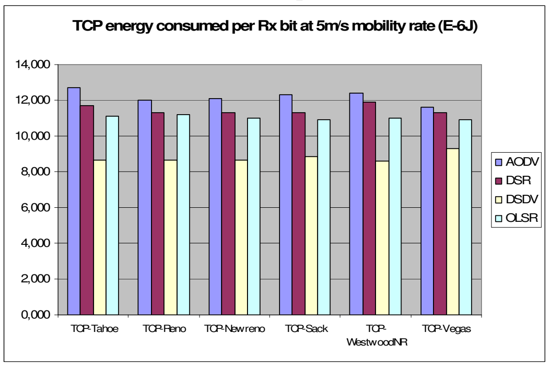

Figure 2. Energy consumed per received bit at 5m/s speed.

Thus, the energy consumed trying to retransmit the lost packets while there is no valid route between the source and the destination will be wasted. Besides, the sender might stay in "back off" state for a while after re-establishing the lost link due to lack of knowledge that there is a valid route to continue the communication. As a result, TCP connection bandwidth will be underutilized. Although that in both proactive routing protocols, routing table updates are periodically transmitted throughout the network in order to maintain table consistency. DSDV tries to help alleviate the potentially large amount of network traffic that such updates can generate. In DSDV, route updates can employ two possible types of packets; the first is known as a full dump. This type of packet carries all available routing information. During periods of occasional movement, these packets are transmitted infrequently. Instead, smaller incremental packets are used to relay only that information which has changed since the last full dump. Thereby, decreasing the amount of traffic generated in the network. We must note that, the way the routing protocol updates its routing information may affect TCP connection data flow. In other words, heavy routing updates or control messages might lead to network congestion, provoking packet loss for some TCP flows. Consequently, leads to extra energy consumption in retransmitting lost packets.

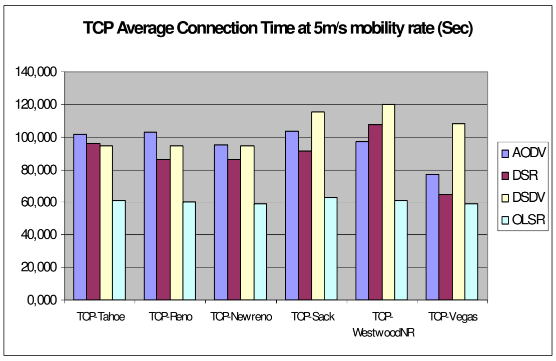

Figure 3. TCP Average connection time at 5m/s speed.

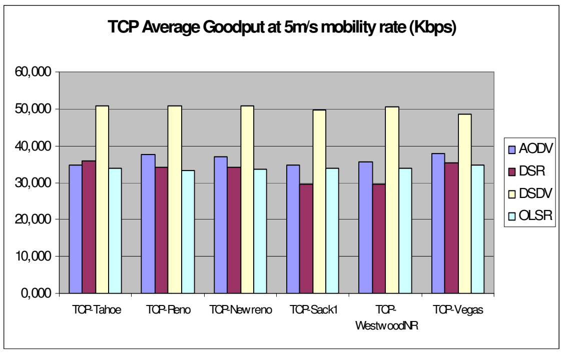

Figure 4. TCP Average Goodput at 5m/s speed.

That explains why DSDV routing protocol results in TCP less energy consumption per received bit compared to OLSR [as can be seen in Figure 2]. Distance vector protocols have always less routing messages overhead than link state ones. In addition, the bandwidth taken by routing update messages in the network is found to be higher in the case of link state routing protocols. This can be verified from Figure 4. We see from the Figure that OLSR has lower goodput than DSDV. This is due to the high routing messages overhead caused by OLSR and on the numerous layer 2 contentions and congestion periods that is led by this overhead. When comparing OLSR and DSDV performances we can notice two surprising aspects: Figure 3 illustrates that OLSR has the shortest average connection time compared to the other routing protocols while Figure 4 depicts that, all TCP variants have the highest average goodput when combined with DSDV routing protocol. This is explained by the fact that simulations' results show that the number of dead nodes in OLSR case is more than those in DSDV simulation case. From that, we can affirm that TCP nodes are run off battery much faster in the case of OLSR which made TCP connections to live shorter.

When analyzing the effect of reactive routing protocols, one can expect that with nodes' mobility; both DSR and AODV would have heavy route discovery processes to maintain network connectivity. This is confirmed by Figure 2 where we can see that both AODV and DSR have the worst performances compared to the proactive routing protocols. From Figure 2, we also notice that DSR routing protocol causes all TCP variants to consume less energy per received bit compared to AODV. DSR is a source routing protocol where the sender implies the entire forwarding path within the packet header. Intermediate nodes forward data packets based on source route. This in turn decreases the route discovery process overhead at intermediate nodes. While in AODV (hop-by-hop routing protocol), intermediate nodes maintain routing tables and makes autonomous forwarding decisions. Then, the use of route cache in DSR can speed up route discovery, and reduce propagation of route requests. Thus, limiting congestion conditions in the network and having the chance to recover from link loss before sender's RTO times out. In the mean time, route cache in DSR also reduces the time spent in link loss recovery (in the case when there is a valid route to the destination in its cache). This may lead, sometimes, to shorter average connection time than AODV, as can be verified from Figure 3. We have to mention here, that even route cache in DSR has a positive effect on TCP performances; stale routes in the cache may cause TCP performance degradation. This effect is not noticeable for small scale mobility rates as can be verified by Figures 2 and 3 however we expect a decrease of DSR performances with the increase of the mobility rate. This assumption will be verified in the next sections.

4.2.2. TCP Performances at 15m/s mobility rate

Figure 5 shows that all TCP variants have less energy consumption with proactive protocols than with reactive protocols for the same reasons explained above. Additionally, we notice that in most cases, AODV leads to high TCP energy consumption per received bit. AODV has to trigger route discovery process each time there is a broken link between any two communicating nodes. Figure 5 depicts an interesting result regarding TCP Vegas performances. In the case of this TCP variant, we find that DSR increases TCP energy consumption behind that of AODV. TCP Vegas tries to resend any lost packet at the reception of the first duplicate acknowledgement (does not wait for the third duplicate ACK as most TCP variants). When the mobility of network nodes increases, DSR cache would contain stale routes (as the routes become invalid faster due to nodes mobility). TCP Vegas tries to use the cached route. This might lead to more losses in the network. Thus, more TCP energy consumed to recover from frequent packet losses caused by using the cached stale routes. We will see in the following that stale routes within DSR routing cache would affect all TCP variants as nodes mobility rate increases. This is an expected result. The number of stale routes in DSR routing cache increases due to high dynamic changes of network topology. Also, we mention here that the way used to establish DSR's routing cache might complicate the problem of stale routes. Intermediate nodes can reply to route requests with routes from their caches. Thus, a stale route could be propagated through the network leading to frequent link failures and consequently more packet losses.

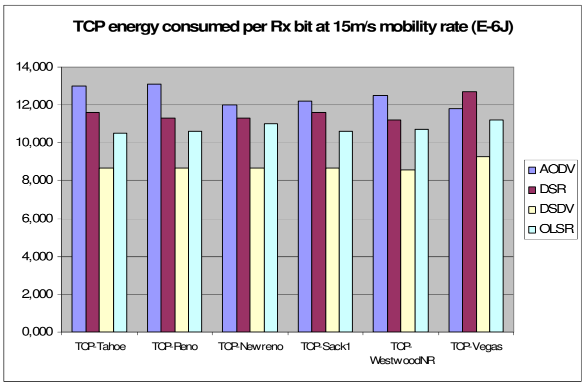

Figure 5. Energy consumed per received bit at 15m/s speed.

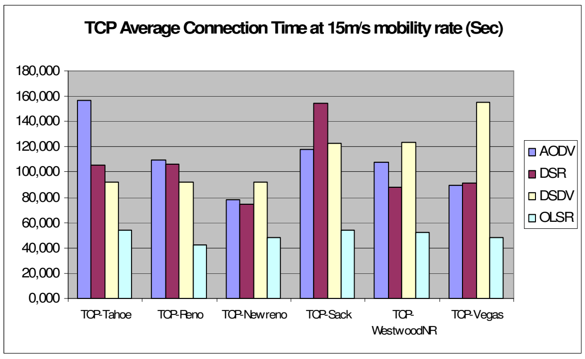

Figure 6. TCP Average connection time at 15m/s speed.

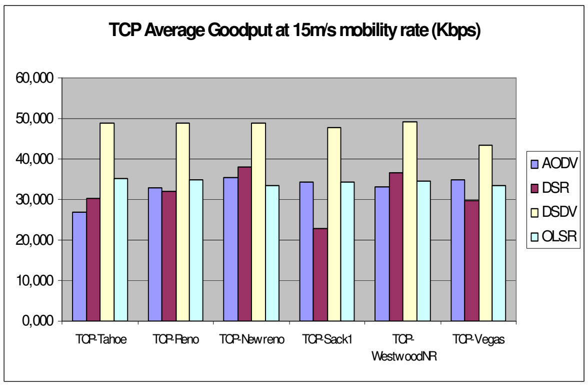

Figure 7. TCP Average Goodput at 15m/s speed.

Figure 6 illustrates a surprising result where each TCP variant has a different behavior in term of average connection time. In fact, in this Figure the effect of ad hoc routing protocols are hidden by the specificities of the analyzed TCP variants. Hence, it is worth to note that, TCP New-Reno has the best performance compared to other TCP variants. This is due the fact that TCP New-Reno is able to handle a consecutives packet loss more efficiently than TCP Tahoe and TCP Reno.

For TCP SACK, the time passed at the senders' side in order to deduce the lost packets from the SACK header increases as the number of lost packets might increase with mobility rate. Regarding TCP Vegas and TCP WestwoodNR, TCP New-Reno congestion control algorithm is less complex than these variants. In both variants, there will be more time consumed in order to calculate the transmission window according to network conditions. As there will be continuous network topology changes due to mobility. The calculations will be done frequently.

Figure 7 illustrates that, DSDV still outperforms other routing protocols in term of TCP average goodput. This can be explained by the behavior of proactive Distance Vector protocols that recover from link losses more rapidly than proactive one (cf. Section 4.2.1). In addition, it can be clearly seen that, TCP average goodput decreases with nodes mobility rate (as could be verified by comparing Figures 4 and 7). When nodes mobility increases, link failures occur more frequently leading to more data packet loss within the network. Thus, underutilizes the available bandwidth. The reader may refer to [3] for a more detailed study of TCP variants performance, regarding to link losses.

4.2.3. TCP Performances at 30m/s mobility rate

Figure 8 illustrates the effect of high mobility rate on TCP energy consumption per received bit when using different ad hoc routing protocols. We can see from this figure that, DSDV causes the least TCP energy consumption per received bit for all studied TCP variants. This is due to the same reasons as above (cf. Section 4.2.1). On the other hand, we notice that, DSR routing protocol causes more TCP energy consumption than AODV in most cases. This is due to the stale routes of DSR routing caches that instead of enhancing performance (as it is supposed to be), it degrades it. With many stale routes within the DSR's routing cache (that may propagate to other nodes within the network), the result is more losses in the network. Thus, TCP triggers its congestion control algorithm more frequently with the increase of the mobility rate. Let us now consider the particular case of TCP Vegas. Indeed, even that TCP Vegas consumes less energy per received bit when used with DSR than with AODV case. Simulations' results show that TCP Vegas has sent and received less data It is worth to note that, at high mobility rate, the frequent changes in network topology (due to nodes mobility) leads to unstable RTT values between the communicating nodes. TCP Vegas depends on measured RTT values in the network to adjust its RTO timer and its transmission rate. Thus, might provoke mistaken RTO timer or transmission rate calculations. If the calculated RTO timer value is much less than the actual RTT between the communicating nodes, that means sent packets will be resent without need (unnecessary retransmissions). On the other hand, if it is much longer than the actual RTT between the communicating nodes, we would wait for unnecessary long time to perceive that there is a lost packet. We have to mention that TCP Vegas still contains Reno's coarse-grained timeout code in case its proper mechanisms fail to recognize the loss. The low goodput of TCP Vegas, shown in Figure 10, confirms that its transmission rate window was not well dimensioned in this case, and we think that the estimated RTO value is much less than the actual RTT in most of the cases (as explained above). Regarding TCP average connection time, Figure 9 depicts that the average connection time of TCP sessions differs with the TCP variant.

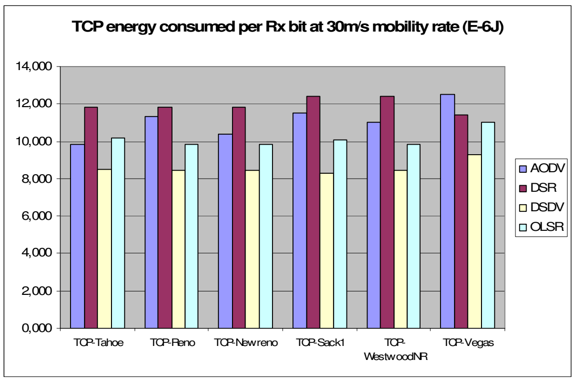

Figure 8. Energy consumed per received bit at 30m/s speed.

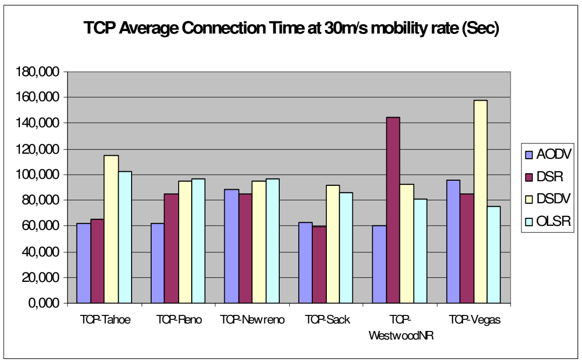

Figure 9. TCP Average connection time at 30m/s speed.

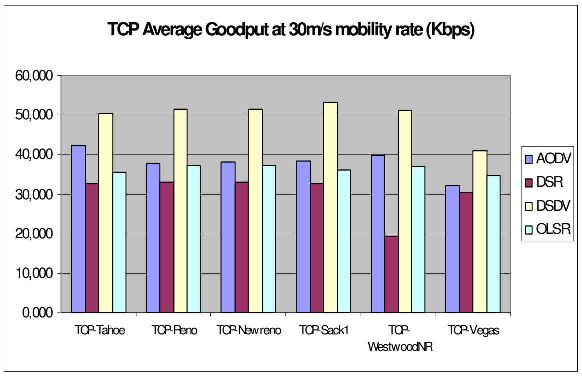

Figure 10. TCP Average Goodput at 30m/s speed.

For example, with some variants (i.e. TCP Tahoe, TCP Reno, TCP New-Reno, and TCP SACK), both reactive protocols lead to less average connection time than proactive ones. On the other hand, Figure 10 illustrates that these variants have better goodput at high mobility rate than at low mobility rates. This can be viewed as an interesting result. Actually, if the communicating nodes are moving in such a way that there is always a direct communication path between them; this would improve the bandwidth utilization over TCP connection.

Consequently, leads to higher goodput over the connection. We must mention here, that a similar result was found by the authors of [8]. In [8] the authors studied TCP Reno performance over a mobile ad hoc network using DSR as routing protocol. From our observations, we might generalize this result for almost all TCP variants with different ad hoc routing protocols. From Figures 8, 9 and 10, we conclude that both proactive protocols (DSDV and OLSR) have certain stability within the network (in terms of energy consumption per received bit and average goodput) over the reactive protocols (AODV and DSR). Since the control traffic of both proactive protocols is continuous and periodic, it keeps the network links more stable, where reactive protocols, with bursty flooding for route discoveries and repairs, may cause numerous collisions on network links.

5. Conclusion And Future Work

It was proved that TCP performance is highly influenced by the dynamic nature of mobile ad hoc networks, due to nodes mobility. Also, the choice of ad hoc routing protocol to be implemented within the network affects TCP behavior within this network. Indeed, on the one hand, the overhead of control messages caused by the routing protocol might lead to network congestion. Thus, enforces TCP to trigger its congestion control algorithm. On the other hand, the route recovery approach used by a particular ad hoc routing protocol can either have a good or a side effect on TCP behavior. Hence, if the route recovery time needed by the routing protocol to re-establish a broken link is shorter than RTO timer of TCP, TCP will not experience packet loss. Otherwise, TCP sender recognizes the packet loss through RTO timer. This latter will lead TCP to react inadequately to the packet loss by triggering its congestion control algorithm.

Regarding mobility of nodes at mobile ad hoc networks, it was interesting to find that nodes' mobility is not always a degradation factor of TCP performance. Sometimes, mobility might help ad hoc routing protocols to re-establish broken links faster. This prevents TCP sender's congestion control algorithm to start. The importance of choosing the right mobility rate still needs some research. As a general result, we found that DSDV could be the best choice to be implemented when using TCP within mobile ad hoc networks. Indeed, it proved to be the best performing routing protocol with all TCP variants within mobile ad hoc environment. DSDV, as proactive and distance vector routing protocol, includes all available routes towards any destination in the network in its routing table. In the presented work, we studied the behavior of TCP variants in mobile ad hoc networks by varying the ad hoc routing protocol and nodes' mobility rate. In the above simulations, we were not able to measure accurately the idle energy of TCP variants. NS-2 does not apply the idle energy consumption of TCP. In order to find the total energy consumption of each TCP variant (including both idle energy and communication energy), we intend in our future work to study the idle energy consumed by each TCP variant by the means of real test-bed experiments.

References

[1] V. Ramarathinam, M. A. Labrador, «Performance Analysis of TCP over Static Wireless ad hoc networks», *In ISCA 15th International Conference on Parallel and Distributed Computing Systems, PDCS'02*, Sep. 2002.

[2] V. Tsaoussidis, A. Lahanas and C. Zhang, «The Wave and Probe Communication mechanisms», *The Journal of Supercomputing Kluwer Academic Publishers*, Vol. 20, No 2, Sep. 2001.

[3] A. Seddik-Ghaleb, Y. M. Ghamri Doudane, and S.-M. Senouci, « A Performance Study of TCP variants in terms of Energy Consumption and Average Goodput within a Static Ad Hoc Environment », *To appear in the 2nd ACM International Wireless Communications and Mobile Computing Conference, IWCMC'06*,Jul. 2006.

[4] J. Liu, S. Singh, «ATCP: TCP for Mobile Ad Hoc Networks», *IEEE Journal on Selected Areas in Communications*, Vol. 10, No 7, Jul. 2001.

[5] M. Zorzi and R. Rao, «Energy Efficiency of TCP in a local wireless environment», *Mobile Networks and Applications*, Vol. 6, No. 3, July 2001.

[6] S. Agrawal and S. Singh, «An Experimental Study of TCP's Energy Consumption over a Wireless Link», *4th European Personal Mobile Communications Conference, EPMCC'02*, Feb. 2002.

[7] H. Singh and S. Singh, «Energy consumption of TCP Reno, New Reno, and SACK in multi-hop wireless networks», *In ACM SIGMETRICS'02*, Jun. 2002.

[8] [HOL 99] G. Holland and N. Vaidya, «Analysis of TCP performance over mobile ad hoc networks», *in 5th annual ACM/IEEE International Conference on Mobile Computing and Networking, ICMCN'99*, Aug. 1999.

[9] H. Singh, S. Saxena, and S. Singh, «Energy Consumption of TCP in Ad Hoc Networks», *J. Wireless Networks*, Vol. 10, No. 5, Sep. 2004.

[10] M. Allman, V. Paxon, W. Stevens, «TCP Congestion Control», *RFC 2581, IETF*, Apr. 1999.

[11] V. Jacobson., «Congestion avoidance and control», *In ACM SIGCOMM'88 symposium on communications architectures and protocols*, Vol. 18, No. 4, Aug. 1988.

[12] V. Jacobson., «Modified TCP Congestion avoidance Algorithm», end2end-interest mailing list, 30 Apr. 1990. (ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail).

[13] K. Fall and S. Floyd., «Simulation-based comparison of Tahoe, Reno, and sack TCP*», in ACM Computer Communications Review*, Jul. 1996.

[14] J. Hoe., «Start-up Dynamics of TCP's Congestion Control and Avoidance Scheme », *Master's thesis, MIT*, Jun.1995.

[15] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, «TCP Westwood: Bandwidth *Estimation for enhanced transport over wireless links», 7th annual International Conference on Mobile Computing and Networking, ICMCN'01*, Jul. 2001.

[16] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson «TCP Vegas: New Techniques for Congestion Detection and Avoidance», *ACM SIGCOMM'94*, Aug. 1994.

[17] C. E. Perkins and P. Bhagwat, «Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers», *ACM Computer Communications Review*, Oct.1994.

[18] L. R. Ford Jr. and D. R. Fulkerson, «Flows in Networks», Princeton Univ. Press, 1962.

[19] T. Clausen, «Comparative Study of Routing Protocols for Mobile Ad-Hoc NETworks», *Research Report, RR-5135, INRIA*, Mar. 2004.

[20] C. E. Perkins and E. M. Royer, «Ad-hoc On-Demand Distance Vector Routing», *In 2nd IEEE Wksp. Mobile Comp. Sys. And Apps, WMCSA'99*, Feb. 1999.

[21] D. B. Johnson and D. A. Maltz «Dynamic Source Routing in Ad Hoc Wireless Networks». *In Mobile Computing*, T. Imielinski and H. Korth, editors, Chapter 5, pp. 153-181, Kluwer Academic Publishers, 1996.

[22] J. Broch, D. Johnson, and D. Maltz, « The dynamic source routing protocol for mobile ad hoc networks», *Internet draft, IETF Mobile Ad Hoc Networking Working Group*, Dec. 1998.

[23] A. Aaron and J. Weng, «Performance Comparison of Ad-hoc Routing Protocols for Networks with Node Energy Constraints», *EE 360 Class Project*, *Stanford University*, Spring 2000-2001.

[24] Network Simulator-NS-2. Available at www.isi.edu/nsnam/ns/

[25] V. Tsaoussidis et al., «Energy/Throughput Tradeoffs of TCP Error Control Strategies», *5th IEEE Symp. Computers and Communications, ISCC'00*, Jul. 2000.