A Performance Study of TCP variants (Tahoe, Reno, New-Reno, SACK, Vegas, and Westwood) in terms of Energy Consumption and Average Goodput within a Static Ad Hoc Environment

Alaa Seddik-Ghaleb*, Yacine Ghamri-Doudane*, and Sidi-Mohammed Senouci**

*Networks and Multimedia Systems Research Group (LRSM),
Computer Engineering Institute (IIE-CNAM)
18 Allée Jean Rostand, 91025 Evry Cedex, France.
seddik@iie.cnam.fr , ghamri@iie.cnam.fr

** France Telecom R&D
2 Av. Pierre Marzin, 22307, Lannion, France.
sidimohammed.senouci@francetelecom.com

Abstract— **TCP was mainly developed to be implemented within wired networks where the main cause for packet loss is network congestion. Conversely, in wireless ad hoc networks there are many other reasons to lose packets (such as fading, interferences, multi-path routing …). In front of these various loss types, TCP reacts by triggering its congestion control algorithm (i.e. considering all these losses as due to congestions). This reaction can be considered as an aggressive behavior in some cases, which may lead to network performance degradation. On the other hand, since ad hoc nodes are battery operated, they need to be energy conserving so that battery life is maximized. Thus, in our work we tried to find the effect of TCP variants' congestion control algorithms on TCP performance (energy consumption and average goodput[1]) in ad hoc networks. Obviously, this study takes into considerations different loss types that may occur in ad hoc environment in order to find the best adapted TCP variant for such networks. The results of our current work are intended to be used as a guideline for the design of specific TCP enhancements for ad hoc networks.**

I Introduction

TCP has gained its place as the most popular transmission protocol due to its wide compatibility to almost all today's applications. However, TCP as it exists nowadays may not well fit to ad hoc networks. It was designed for wired networks where the medium Bit Error Rate (BER) is very low and network congestion is the primary cause of packet loss. Unlike wired links, wireless radio channels are affected by many factors that may lead to high levels of BER. Additionally, TCP does not have the ability to recognize whether packet loss is due to network congestion, channel errors, or link failure. In this paper, we focus our attention on studying the impact of wireless ad hoc environment characteristics on the energy efficiency of the six major TCP variants (Tahoe, Reno, New-Reno, SACK, Vegas, and WestwoodNR) as well as the obtained goodput.

The remainder of this paper is organized as follows: after presenting the motivation behind our current work in section 2, section 3 presents each TCP variant as well as their performances in term of energy consumption and average goodput. Finally, we summarize the main results of this work and give some ideas for future work.

II Motivation

In the last few years, many researchers have studied TCP performance in terms of energy consumption and average goodput within wireless mobile networks [1][2][3]. Due to the specific issues related to wireless ad hoc networks, it is expected that the performance of TCP will be affected considerably in these environments. In wireless ad hoc networks, reasons for packet losses are more sophisticated than traditional wireless (cellular) networks. Those reasons include the unpredictable wireless channel characteristics due to fading and interference (implying a high BER), the vulnerable shared media access due to random access collision, the hidden and exposed terminal problems, path asymmetry, multi-path routing, and so on. Undoubtedly, all of these pose great challenges on TCP to provide reliable end-to-end communications in such environment.

Many research projects were specifically interested in studying the TCP performances (energy consumption and/or goodput) within such environments [3][4]. 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. This comparative study

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

takes place under different error loss situations. We take into consideration wireless channel effects and link failure cases. We make our simulations using a large number of nodes, in order to realize the effect of losing a non-adjacent node on energy consumption of the other nodes in the network.

The aim of this study is to help understanding the impact of the different TCP loss recovery mechanisms on TCP performance in ad hoc environments. Thus, obtained results can be used as a guideline for efficient design of new specific TCP enhancements for ad hoc networks.

III Comparative Study of TCP Performance in a Static Ad hoc Network

In this section we study the performances of different TCP variants in terms of energy consumption and average goodput within a static ad hoc network. We study the performances regarding two common situations: (i) the link loss scenario and (ii) different BER level scenarios.

Note that, for each TCP variant a short overview of its loss recovery mechanisms. For more details on the behavior of these variants, the reader can refer to the corresponding references.

III.1 Simulation Scenarios and Topologies

Our simulations are realized using the Network Simulator version 2 (NS-2) [6]. 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. 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, reception, 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. When this 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. In our simulations, we consider the use of the Optimized Link State Routing (OLSR) [7] as routing protocol. 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.

We use different loss model scenarios with several values of BER (5%, 10%, and 15%) and a lost link (LL) scenario. 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 [8] 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 goodput of TCP.

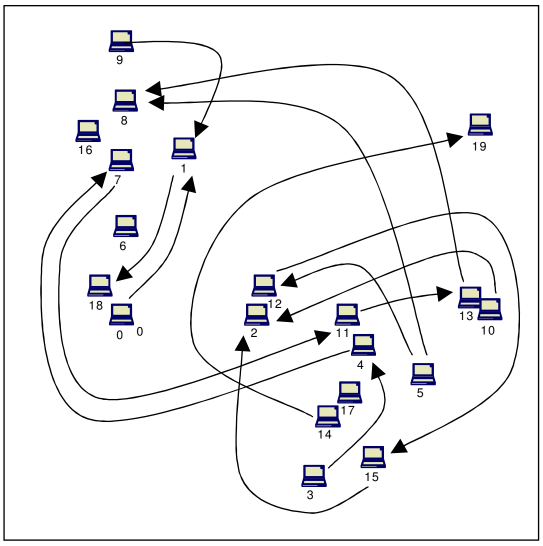

Figure 1 Network topology and FTP sessions.

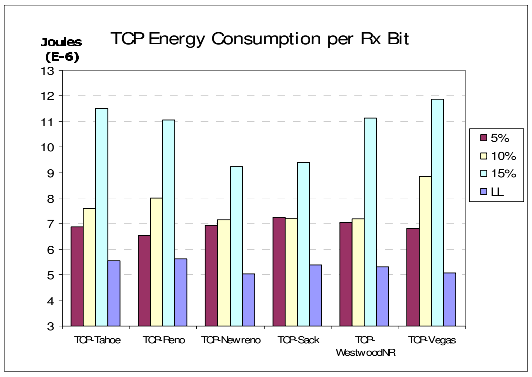

Figure 2 Comparison of TCP energy consumption per received bit.

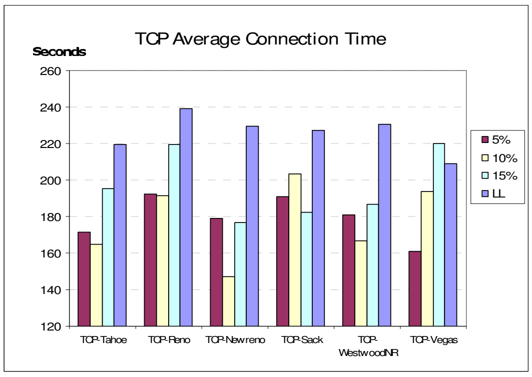

Figure 3 Comparison of TCP sessions Average connection time.

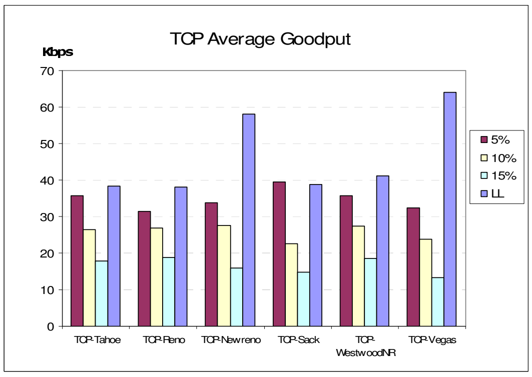

Figure 4 Comparison of TCP Average Goodput.

III.2 TCP Tahoe Performance

III.2.a Overview

TCP Tahoe implementation added a number of new algorithms and refinements to earlier implementations. It is the first TCP variant that incorporates congestion control mechanisms: Slow-Start, Congestion Avoidance, and Fast Retransmit [5][9]. 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 acknowledgments, 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. The slow start phase is then triggered.

III.2.b Performances Regarding to Link-Losses

From Figure 2, we can conclude that our results confirm the results obtained in [10]: TCP Tahoe performs better in the case of burst errors (lost link) than the case of random BER (as can be seen in Figure 2). Indeed, each time a lost link in the network breaks a route handling a TCP flow, this latter experiences several consecutive packet losses (burst losses). TCP Tahoe saves energy consumption per received bit because it backs off in the presence of burst losses. This confirms the fact that: "at burst error case or when there is a lost link in the network, it is better to stop data transmission until a new route is found".

In [10] the authors leave the important issue of the energy efficiency tradeoffs involved when backing off increases delays, and hence the overall connection time. Figure 3 can explain the effect of "backing off" algorithm of TCP Tahoe on the average connection time and hence on the idle energy consumption of TCP. Although that backing off saves energy consumed per received bit in the burst error case, we have found that it introduces an extra delay in the network. And as mentioned before, the longer the connection time, the greater the total energy consumed due to the idle energy consumption at the node [8]. At idle times, although that the node does not send data, it will listen to the radio waves in order to receive the acknowledgement. There is also time consumption at the CPU unit in which it executes the used TCP algorithms (Timeouts, CWND threshold adjustments, etc.). Figure 4 illustrates that in terms of average goodput, TCP Tahoe has a good performance the event of a lost link compared to BER cases. This is due to backing off algorithm, as the connection stays inactive during this time. This action saves unnecessary retransmissions until a new route is found. Then, inversely to the BER case, where it triggers its congestion control algorithm each time there is a lost packet, TCP Tahoe will enter the Slow-Start phase only once. As a result, the transmission rate does not stay low leading to a somewhat good utilization of bandwidth.

III.2.c Performances Regarding to Different Levels of BER

Figure 2 shows that at random loss BER, we notice that the performance of TCP Tahoe degrades in term of energy consumption per each received bit when the channel error rate increases, which is an expected result. This is due to the fact that TCP will trigger the congestion control algorithm more frequently (the higher the

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

BER, the more frequently TCP triggers its congestion control algorithm). This leads, also, to important decrease in the efficiency of the bandwidth utilization (as can be verified from Figure 4). Frequent triggering of TCP Tahoe's congestion control means entering the Slow-Start phase in succession. This will reduce the transmission rate many times (i.e. the transmission rate stays low leading to under utilization of bandwidth). We must mention here that Slow-Start algorithm considers all packets in the same window that caused the Slow-Start, as losses.

As a consequence, even that TCP Tahoe saves energy due to its backing off algorithm in lost link case; it does not show a good performance facing different BER levels.

III.3 TCP Reno Performance

III.3.a Overview

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 [11]. 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 [12]. 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.

III.3.b Performances Regarding to Link-Losses

The simulation results depicted by Figure 2 proves that in the case of burst packet loss (lost link), TCP Tahoe may have lower energy consumption per received bit, since it backs off in front of the burst errors, which may increase the chance of successful retransmission after that. For example, if the burst packet loss is due to a bad connection or a link failure, backing off for a while, will help avoiding the unnecessary retransmissions. As can be seen in Figure 4, TCP Tahoe has better goodput than TCP Reno in the case of lost link. On the other hand, Figure 3 shows that TCP Reno has a long average connection time compared to almost all other TCP variants especially at high BER and link loss cases.

All these results are due to the fact that TCP Reno is unable to recover from more than one packet loss at a time (i.e. is unable to recover from consecutive losses). When there are many packets lost within a transmission window, TCP Reno decreases its transmission rate by half each time there is a lost packet. Then after two trials of loss recovery, TCP Reno reaches almost the same transmission rate as in TCP Tahoe. After three trials of recovery, TCP Reno has to wait to RTO expiration that leads to backing off and entering slow-start phase (exactly as in TCP Tahoe). The above process leads to more time consumption in the first two trials of recovery, while that TCP Tahoe is backing off and goes through slow-start directly. Furthermore, there is an extra energy consumed in the two first trials which do not reach the destination due to the lost link (Figure 2). This led us to conclude that TCP Reno will probably consume more total energy than TCP Tahoe without leading to better goodput (Figure 4).

III.3.c Performances Regarding to Different Levels of BER

Figure 2 demonstrates that the energy consumption of TCP Reno increases with BER level as it can not manage more than one lost packet per window of data (as explained earlier). TCP Reno encounters several problems with multiple packet losses in a window of data. This usually happens when invoking fast retransmit and fast recovery in succession. Additionally, as expected, Figure 4 shows that the average goodput of TCP Reno is getting worse when the BER increases. Invoking loss recovery algorithms several times leads to multiplicative decreases of cwnd and SSThresh, which in turn impacts the goodput. TCP Reno was developed in order to enhance the goodput of TCP within wired networks, especially when there is only one lost packet from a window of data. This enhancement can not be noticed in some studied cases (which correspond to the ad hoc environment) in terms of average goodput or energy consumption per received bit, as we may have more than one lost packet from a window of data even at low BER (5%).

Hence from the above results, we found from the above that TCP Reno does not fit well within an ad hoc environment, where it is frequent that many packets could be lost at a time.

III.4 TCP New-Reno Performance

III.4.a Overview

TCP New-Reno includes a small change to the Reno algorithm at the sender [13][14] The 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 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 [12]

III.4.b Performances Regarding to Link-Losses

In this variant of TCP, there are noticeable savings in the energy consumption per received bit in the case of burst packet losses. This is due to the used partial ACKs. For burst packet loss, TCP Tahoe performs better than TCP Reno (as explained earlier), and TCP New-Reno outperforms TCP Tahoe (Figure 2). This is due to the fact that TCP New-Reno is not obliged to wait for RTO before retransmitting the lost data and in the mean time its congestion window increases faster than that of TCP Tahoe (Fast Recovery algorithm that exists in TCP New-Reno aims to halves the congestion window instead of minimizing it as in TCP Tahoe). We can see from Figure 4 that TCP New-Reno outperforms both TCP Tahoe and TCP Reno in case of burst error due to its ability to recover from multiple losses at a single window of data (partial acknowledgements).

Finally, Figure 3 shows that TCP Tahoe has shorter average connection time than TCP New-Reno. Indeed, TCP New-Reno can not resend more than one lost packet per RTT. TCP New-Reno retransmits a lost packet after receiving a partial ACK that indicates that the next packet in sequence is lost. Thus, the recovery time of lost packets is equal to the number of these lost packets multiplied by RTT value. Hence, the more the number of lost packets is, the longer the recovery time and consequently the longer the average connection time. That explains the long average connection time of TCP New-Reno in case of link failure case.

III.4.c Performances Regarding to Different Levels of BER

At low BER (5%), we can notice no difference in the energy consumption between TCP Reno and TCP New-Reno which is not surprising. The development of TCP New-Reno was mainly concerned by its behavior in front of multiple packet loss within a single window of data to overcome this problem in TCP Reno. Thus, it is expected to recognize that effect at higher loss rates which is verified in Figure 2. Thus, at high BER, we found that TCP New-Reno has a good performance in terms of energy consumption per received bit. Also, we notice, from the same Figure, that the energy consumption increases with the BER. Figure 3 shows that TCP-New-Reno has shorter average connection time compared to TCP Reno, at all studied cases, due to the partial ACKs used which does not exist within TCP Reno. Also, Figure 4 shows that the goodput of TCP New-Reno degrades with the BER, which is an expected notice as the channel losses increase. We can notice also that, TCP Reno outperforms TCP New-Reno at high BER. Although that the simulation results show that TCP New-Reno has the ability to send and receive more TCP data than TCP Reno and at a shorter connection time, we found that some bad connections (which have bad performance) had affected the whole simulation performance. In fact, we found that some connections had not sent much data compared to other connections. This can be explained by the fact that the number of dead nodes with TCP New-Reno is higher than those with TCP Reno. This has an effect on the whole simulation as the average goodput had been calculated and averaged over all the TCP connections in the simulations' scenario.

TCP New-Reno performs generally well within a static ad hoc network. Partial ACKs in TCP New-Reno helps it to better manage the recovery of consecutive packet losses. Its main drawback is the time spent to recover from multiple losses.

III.5 TCP SACK Performance

III.5.a Overview

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[12].

III.5.b Performances Regarding to Link-Losses

Although that it was expected that TCP SACK outperforms TCP New-Reno in terms of energy consumption per received bit. This is mainly due to the Selective Acknowledgements feature that allows TCP SACK to terminate the retransmission of lost data more quickly than TCP New-Reno (which has to wait for all partial ACKs to know which segments are missing at the receiver). However, in terms of energy consumption, this gain is neutralized (Figure 2). We think that this is due to the overhead that is introduced by the SACK option. Indeed, SACK packets[3] can in certain cases reach the double of normal TCP ACK packet size[4]. This leads to more energy consumption per sent SACK. Figure 3 is showing that TCP SACK has a shorter average connection time in case of burst loss (lost link) as it retransmits the lost packets consecutively without waiting for RTTs. The packets that may use the recovered route (by the routing protocol) are not retransmitted. In the mean time, TCP New-Reno spends more time in order to recover from this type of losses (as mentioned early). We can also notice that the average goodput obtained when using TCP SACK is higher than the one obtained by both TCP Tahoe and TCP Reno (Figure 4). This is also due to its ability to retransmit only the lost packets. Inversely, we found from the same Figure that TCP New-Reno has better goodput than TCP SACK. Actually, the simulations results show that TCP SACK simulation scenario has transmitted less data bytes than TCP New-Reno. Additionally, the results show also that there were more dropped data bytes than TCP New-Reno. These observations lead us to conclude that the high energy consumption of TCP SACK is the main cause of having lower goodput. The high energy consumption of TCP SACK has influenced the whole simulation performance in the way that the simulation nodes' were going out of battery more quickly than in the case of TCP New-Reno.

Furthermore, one should note that the energy consumption (per time unit) due to the operation of the algorithms of TCP SACK (CPU units) is higher than for TCP New-Reno. This is due to the important additional overhead related to the timers and algorithms TCP SACK have to run. Then, even if, TCP SACK has a slightly low average connection time in the burst error case (link lost), the energy consumed by TCP SACK is certainly higher then the one consumed by TCP New-Reno.

[3] SACK packet size = IP Header + TCP ACK Header + SACK option = 20 bytes + 20 bytes + 40 bytes = 80 bytes. 40 bytes is the maximum size of a TCP Header option. SACK can use this entire size to transmit the *Selective Acknowledgement*. This size depends on the number of segments to be acknowledged.

[4] Normal TCP ACK packet size = IP Header + TCP ACK Header = 20 bytes + 20 bytes = 40 bytes.

III.5.c Performances Regarding to Different Levels of BER

Figure 2 shows also that, at high BER (15%), TCP SACK has a good performance in terms of energy consumption per received bit compared to the other variants. From that, we can conclude that using selective acknowledgements might be effective within high BER wireless ad hoc networks. On the other hand, we must mention that the processing overhead of TCP SACK would have a negative effect on TCP performance as the BER increases. From Figure 3 we can see that, at different BER cases, TCP New-Reno has a shorter average connection time than that of TCP SACK. Here, we find that, as TCP SACK nodes goes down more quickly than those of TCP New-Reno; and we had many link losses in the network. Thus, TCP SACK recognizes the packet loss by RTO. This in turn increases the average connection time. This explains why TCP SACK does not consume a lot of energy compared to other TCP variants. The energy consumed here is the idle energy. Regarding the average goodput of TCP SACK, Figure 4 demonstrates that the performance of TCP SACK degrades as the BER of the wireless channel increases for the same reasons as above. TCP SACK enters Slow-Start phase each time after RTO expiration. This leads to underutilization of the bandwidth.

From all these results, we can say that TCP SACK has not always the best performances. Sometimes TCP New-Reno outperforms TCP SACK. This is due to the higher energy consumption of TCP SACK.

III.6 TCP WestwoodNR Performance

III.6.a Overview

TCP WestwoodNR 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. 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 RTT values.

III.6.b Performances Regarding to Link-Losses

Figure 2 illustrates that TCP WestwoodNR has comparable energy consumption per received bit when compared with TCP New-Reno. At link loss case, both TCP variants recognize the packet loss with RTO expiration. Thus, both react the same way by backing off for a while then triggering the Fast Recovery and entering Slow-Start phase. Figure 3 shows that TCP WestwoodNR has always longer average connection time than that of TCP New-Reno. Furthermore, at the lost link case, the average goodput of TCP WestwoodNR (Figure 4) is less than that of TCP New-Reno. This is due to the lost ACKs. Indeed, in order to estimate the end-to-end bandwidth and discriminate among loss types, TCP WestwoodNR relies on the received ACKs. In a situation where there is several ACK losses, this may lead to wrong estimate of the end-to-end bandwidth and consequently to TCP WestwoodNR misbehavior.

III.6.c Performances Regarding to Different Levels of BER

From Figure 2, we found that TCP WestwoodNR has higher energy consumption per received bit than TCP New-Reno in most cases. In addition, it can be noticed (from the same Figure) that TCP WestwoodNR energy consumption is getting worst with BER increase. We think that its dependence on RTT measurements to calculate the estimated bandwidth is also responsible of this latter effect. Similarly to the lost link case, as BER increases over the wireless channels, the returned ACKs might be lost or corrupted. These lost or corrupted ACKs could cause mistaken estimated bandwidth calculations. From Figure 4, we recognized that TCP WestwoodNR has better performance in term of average goodput than TCP New-Reno at low BER, due to its ability to adjust its transmission rate according to the network bandwidth conditions (instead of blindly halving it as in TCP New-Reno).

III.7 TCP Vegas Performance

III.7.a Overview

TCP Vegas extends Reno's retransmission mechanisms as follows. First, Vegas reads and records the system clock each time a segment is sent. When an ACK arrives, Vegas reads the clock again and does the RTT calculation using this time and the timestamp recorded for the relevant segment. Vegas then uses this more accurate RTT estimate to decide to retransmit a lost packets [16] before reaching RTO. TCP Vegas uses RTT values to calculate the actual transmission rate in the network. Also, by comparing that value by the expected goodput in the network, TCP Vegas decides how to modify its transmission rate.

III.7.b Performances Regarding to Link-Losses

It is shown (Figure 3) that TCP Vegas has low average connection time in case of burst error (lost link) due to its ability to deduce a good estimation for the transmission rate compared to TCP New-Reno (that simply halves the congestion window size). This behavior also leads to less energy consumption as can be verified from Figure 2. TCP Vegas can be considered the best performing variant in the cases of link loss (burst error loss) as can be shown from Figure 4. Indeed, it is an expected result. TCP Vegas is a modified version of TCP New-Reno. It replies to packet losses faster than TCP New-Reno. The algorithm of TCP Vegas is based on the principal that there are signs prior to congestion in the network. For example, an increase in RTT values is a sign indicating that router's queue is building up and that congestion is about to happen. This will lead to faster recovery from packet losses and to a good utilization of the available bandwidth (in the lost link case).

III.7.c Performances Regarding to Different Levels of BER

At low BER (5%), TCP Vegas has shorter average connection time than all the other TCP variants (Figure 3). This is due to TCP Vegas that may retransmit a packet after the first duplicate ACK. Indeed, on the first duplicate ACK received TCP Vegas checks for the RTT value and compares it with the RTO value to recognize the packet loss (as explained earlier). By doing so, it leads to faster recovery than the other TCP variants that have to stay until the reception of the third duplicate ACK. On the other hand, at higher BER (10% and 15%), TCP Vegas has long average connection time and higher energy consumption per received bit, because of its dependence on the RTT measured values of the received ACKs. Hence, at high BER, we will have a high loss in received ACKs that in turn will force TCP Vegas not to have a good behavior (as can be seen from Figures 2 and 3).

Figure 4 shows that, in the cases of random error losses, TCP Vegas has a bad performance especially when the BER increases. This is also due to the dependence of TCP Vegas on the RTT measurements that may cause mistaken calculations when the BER increases. It thus leads to more unnecessary retransmissions instead of decreasing them.

III.8 Summary

To summarize simulation results obtained in this work, at high BER (15%) TCP New-Reno outperforms the other TCP variants followed by TCP SACK. In low and medium BER (5% and 10%), TCP New-Reno has also a moderate energy consumption per received bit ratio. In addition, TCP Vegas and TCP New-Reno have the best (the least) energy consumption when there is a lost link in the network.

When comparing the average connection time, we found that at medium and high BER (10% and 15%) TCP New-Reno outperforms the other TCP variants. In addition, TCP Vegas followed by TCP New-Reno, have the best (the least) average connection time when there is a lost link in the network. On the other hand, when we have a low BER (5%), the best TCP variant, in term of average connection time, is TCP Vegas. This is due to its ability, at this BER level, to adjust well the congestion window size. Regarding the average goodput, we conclude that most TCP variants perform better at link loss case than at random BER. At low BER we find that TCP SACK has the best average goodput as it resends only the lost packets. At higher BER (10%), our results show that TCP New-Reno would have the ability to achieve better goodput. On the other hand, we find that TCP Vegas has the best performance in the case of lost link due to its modified retransmission time-out algorithm.

Our results show that in almost all studied situations, TCP New-Reno is the one having the most acceptable performances in terms of energy efficiency and goodput in a static ad hoc network. Although that TCP New-Reno does not always have the best results, the partial ACKs that are included helps improving TCP New-Reno's performance in most cases.

According to the previous results, we can also say that the improvements that have been added to TCP SACK, WestwoodNR and Vegas de not well fit to all the situations that may happen in ad hoc networks. Hence, TCP SACK suffers from important energy consumption (due to the SACK option) which has consequences on the survivability of the ad hoc network. For its part, the performance of TCP WestwoodNR is strongly impacted when the number of lost ACKs increases, with the increase of the BER value or in the link loss case. This leads to TCP WestwoodNR misbehavior. Finally, even if TCP Vegas performs well at link loss case, its dependence on RTT values to calculate the transmission rate leads to some misbehavior in the case of wireless channel BER. This misbehavior degrades subsequently the performances of TCP Vegas.

IV Conclusion and Future Work

It was proved that the congestion control algorithm in TCP variants has an important effect on the energy consumption and goodput in an ad hoc network. As a general result, we found that the TCP congestion control algorithms allow for greater energy savings by backing off during burst error cases. Also, we found that the average connection time of a TCP session can give a good indication of the delay introduced in the network, leading to energy consumption. Our research results confirm that TCP as it exits is not suitable for wireless ad hoc networks especially at high BER. On the other hand, we find that TCP variants studied here are more appropriate (having better performances) for dealing with link loss cases that may be considered as a persistent congestion situation in the network. From our comparative study in this paper, we conclude that TCP New-Reno can be considered as a well performing variant within an ad hoc environment among all other TCP variants, because of its ability to handle both random BER and losses due to broken-links efficiently. However, this behavior may be improved as some other TCP variants outperform TCP New-Reno in some situations.

In the presented work, we studied the behavior of TCP variants in static ad hoc networks by varying the type and the importance of losses. In the above scenarios there is no mobility introduced in the network and then no effect due to mobility handling by ad hoc routing protocols. In order to find the effect of both nodes mobility and different routing protocols on TCP performances, we intend (in future work) to extend our study of TCP performance within a mobile ad hoc network environment.

V References

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

[2] S. Agrawal and S. Singh, «An Experimental Study of TCP's Energy Consumption over a Wireless Link», *4th European Personal Mobile Communications Conference*, Feb 20-22, 2001, Vienna, Austria.

[3] H. Singh and S. Singh, «Energy consumption of TCP Reno, New Reno, and SACK in multi-hop wireless networks», in *ACM SIGMETRICS 2002*, June 15-19 2002.

[4] H. Singh, S. Saxena, and S. Singh, «Energy Consumption of TCP in Ad Hoc Networks», *J. Wireless Networks*, Vol. 10(5), Sep. 2004.

[5] M. Allman, V. Paxon, W. Stevens, «RFC 2581: TCP Congestion Control», April 1999, http://www.ietf.org/rfc/rfc2581.txt.

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

[7] Thomas Clausen, «Comparative Study of Routing Protocols for Mobile Ad-Hoc NETworks», INRIA, Mars 2004

[8] V. Tsaoussidis et al., «Energy/Throughput Tradeoffs of TCP Error Control Strategies», *proc. 5th IEEE Symp. Computers and Communications*, France, July 2000.

[9] V. Jacobson., «Congestion avoidance and control», *SIGCOMM'98 symposium on communications architectures and protocols*, pages 314-329, 1988. An updated version is available via ftp://ftp.ee.lbl.gov/papers/congavoid.ps.z.

[10] M. Zorzi and R. Rao., «Energy Efficiency of TCP in a local wireless environment», *Mobile Networks and Applications*, Volume 6, Issue 3, July 2001.

[11] V. Jacobson., «Modified TCP Congestion avoidance Algorithm», *Technical report*, 30 April, 1990. Email to the end2end-interest mailing list, URL ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt.

[12] K. Fall and S. Floyd., «Simulation-based comparison of Tahoe, Reno, and sack TCP», in ACM *computer communications review*, July 1996.

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

[14] D.D. Clark and J. Hoe., «Start-up Dynamics of TCP's Congestion Control and Avoidance Scheme », Technical *report*, June, 1995. Presentation to the Internet end2end Research Group, cited for acknowledgement purposes only.

[15] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, «TCP Westwood: Bandwidth Estimation for enhanced transport over wireless links», *proc. of the 7th annual international conference on mobile computing and networking*, July 2001.

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