Pervasive and Mobile Computing 7 (2011) 60–77

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Measurement of TCP computational and communication energy cost in MANETs

Alaa Ghaleb-Seddik [a,*], Yacine Ghamri-Doudane [a], Sidi Mohammed Senouci [b], Nazim Agoulmine [c]

[a] Networks and Multimedia Systems Research Group (LRSM), Ecole Nationale Superieure d'Informatique pour l'industrie et l'Entreprise (ENSIIE), 1 Place de la Résistance, 91025 Evry CEDEX, France

[b] France Telecom R&D, 2 Av. Pierre Marzin, 22307 Lannion, Cedex, France

[c] Networks and Multimedia Systems Research Group (LRSM), Université d'Evry Val d'Essonnes, Bd. François Mitterrand, 91025 Evry CEDEX, France

ARTICLE INFO

Article history:
Received 8 September 2009
Received in revised form 11 April 2010
Accepted 6 June 2010
Available online 16 June 2010

Keywords:
TCP
Ad-hoc
Multi-hop networks
Energy consumption

ABSTRACT

TCP suffers dramatically when implemented within wireless multi-hop ad-hoc networks due to its specific characteristics (e.g. mobility and being battery dependant). Being battery operated, ad-hoc network nodes have to be energy conservative. Many researches investigated the performance of TCP in terms of throughput and/or energy consumption. Energy consumption studies, however, were only partial. This paper introduces complete, detailed measurements of TCP's energy consumption resulting from both communication and computational energy costs. Our measurements are realized using a hybrid approach, combining simulations and realistic test-bed configuration. Our study is conducted taking into account different TCP variants facing various data loss situations within the network. The obtained results show the impact of TCP's congestion control algorithm on TCP's energy cost.

1. Introduction

TCP is the most commonly-used reliable transport protocol. Today, TCP is supported by almost all Internet applications. However, TCP does not always perform optimally according to the network environment in which it is used. In order to identify TCP's performance limitations and thus be able to address them, it is important to study its behavior, categorize its performance metrics and quantify them in each of the different environments where TCP can be used. In this paper, we analyze an important performance metric within one increasingly-important environment in which TCP is to be used. More precisely, we are interested in studying the energy cost of TCP when used in a wireless multi-hop ad-hoc networks, also known as Mobile Ad hoc Networks (MANETs) environment. The major motivation behind this study resides in the fact that mobile devices are battery-operated and it is crucial to optimize their energy consumption in order to increase their batteries' lifetime. Prior to any improvement, there is a need to better understand how and where energy is consumed in the communication pipeline. The energy consumption of a node can be represented by two major parts: (i) computational energy cost, and (ii) communication energy cost. The computational energy cost of TCP is the energy spent within the node's CPU to perform the various copy operations, compute checksums, respond to timeouts or triple duplicate ACKs, adjust timers, and perform all other book keeping operations. This is the cost linked to the execution of the different TCP congestion control algorithms (Slow-Start, Fast Retransmit/Fast Recovery, and Congestion Avoidance). The communication energy cost is the

* Corresponding author. Tel.: +33 0 1 69 36 73 48; fax: +33 0 1 69 36 73 27.
E-mail addresses: seddik@ensiie.fr, alaa.ghaleb@gmail.com (A. Ghaleb-Seddik), ghamri@ensiie.fr (Y. Ghamri-Doudane), sidimohammed.senouci@orange-ftgroup.com (S.M. Senouci), nazim.agoulmine@iup.univ-evry.fr (N. Agoulmine).

1574-1192/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.pmcj.2010.06.002

energy consumed in the transmission and reception of TCP segments over the wireless links along the route between the source and the destination.

This work is a complete performance study of different TCP variants, within wireless ad hoc networks, in terms of energy consumption. It complements a number of other researches that evaluated the communication energy cost of TCP variants (i.e. energy consumed due to the transmission, retransmission and forwarding of TCP segments) [1–3]. Also, we mention that this work extends our previous study conducted in [4]. Our study in [4] had the objective of calculating the TCP's computational energy cost using a realistic test bed implementation. Additionally, the present work studies the communication energy cost of TCP using NS-2 simulations. The results of both studies are presented in this paper.

In this work, the four major TCP variants, namely TCP New-Reno, TCP SACK, TCP Vegas and TCP Westwood are considered. In order to measure both the computational and communication energy cost while executing their different congestion control algorithms, we implemented different data packet loss models (congestion, interference, link loss, and signal loss). Measurement of the node-level's computational energy consumption is realized using a realistic test-bed configuration. This configuration introduces the effect of a real wireless multi-hop and mobile ad-hoc network environment (i.e. realistic data packet delays and losses). In this paper, we introduce such effects using a MANET delay and packet-loss emulation tool, which we designed and implemented, called SEDLANE (Simple Emulation of Delays and Losses for Ad-hoc Networks Environment). This tool uses Network Simulator version-2 (NS-2) simulation scenarios in order to generate realistic data packet delays and losses in MANETs. The use of such a hybrid approach (simulations and test-bed experiments) makes the evaluation methodology combine the advantages of both methods. Hence, thanks to SEDLANE, the effect of different data packet loss models (congestion, interference, link loss, and signal loss) as well as that of the ad-hoc routing protocol is introduced. Then, through the realistic test-bed configuration, we measure the computational energy cost of TCP's congestion control functions as well as that of the different studied TCP variants at the node's CPU unit, while the communication energy consumption is evaluated through simulations using NS-2.

The ultimate goal of our study is to understand the impact of the different TCP loss recovery mechanisms on TCP performance within wireless multi-hop ad hoc environments. Hence, our conclusions can be used to derive design guidelines for new TCP enhancements suited for wireless multi-hop ad hoc networks.

The reminder of this paper is organized as follows: after presenting the related work in Section 2, we present an overview of the different TCP variants studied in this paper, in Section 3. In Section 4, we describe the methodology we used to evaluate the performance of each of the studied TCP variants (simulations and realistic test-bed) and give an overview of the SEDLANE emulation tool. We show the results obtained through our study and analyze them in Section 5. In Section 6, we discuss the how results obtained through our study can be used as a guidelines and give the specifications of a new TCP variant that can deal with the different data packet loss causes within wireless multi-hop ad hoc networks as well as improving its performance in terms of both throughput and energy consumption. The conclusion of this paper and some ideas for improving TCP performances in wireless multi-hop ad-hoc networks as a future work comes in Section 7.

2. Related work

In the past many researchers have been studying TCP performance in terms of energy consumption and average throughput within wireless mobile networks [1–3]. Due to the specific issues related to wireless ad hoc networks, such as node mobility, bandwidth and energy constraints, it is expected that the performance of TCP will be considerably affected in these environments. Some research projects were specifically interested in studying TCP performance in terms of energy consumption and/or throughput within such environments. In [1], the authors studied the energy consumption and throughput of three TCP variants (TCP Reno, TCP New Reno, and TCP SACK) through test-bed experiments using random data losses. They evaluated TCP total energy consumption and subtracted the idle energy consumption of the TCP nodes. However, in their study, the authors applied random Round Trip Time (RTT) delays and random packet losses in their experiments and did not separate the computational and the communication cost parts. We note here, that RTT delays and data packet loss ratios over wireless ad hoc networks are highly related. For example, as an extreme but realistic case, if the data packet is highly delayed over the network, it could be considered as lost by the TCP sender. Thus, the TCP sender triggers its congestion control algorithm and retransmits the assumed to be lost data packet. Studying TCP using random values does not really reflect the behavior of the wireless ad hoc network environment or nodes' mobility either. The authors in [5] evaluated the energy consumption of the TCP over wireless links with and without the Selective ACKnowledgement (SACK)[1] option The energy consumption was obtained by measuring the time needed to discharge the laptop's battery rather than by direct measurement means. The study did not include other TCP variants and concentrated only on the impact of the SACK option In [6], the authors studied the performance of different TCP variants (TCP Tahoe, TCP Reno, TCP New Reno, and TCP SACK) within wireless static ad hoc networks taking into account different ad hoc routing protocols (DSDV, DSR, and AODV). Using simulations, they evaluated both TCP throughput and communication energy consumption. However, the performance in case of link failure due to the nodes' battery depletion was not studied. Moreover, none of these works [1,5,6] considered the computational energy cost in their studies (i.e. only the communication energy cost was taken into account).

[1] *Selective Acknowledgment* (*SACK*) is an enhancement of TCP in order to improve its behavior in front of out-of-sequence data losses.

The computational energy cost of the TCP implementation was studied in [7]. The study was conducted through test-bed experiments in which they applied random RTT delays and packet losses. Here, again, using random values does not represent the actual behavior of wireless connections, since both RTT delays and data packet losses are correlated. Additionally, the authors in [7] did not evaluate other TCP variants. We also note that the above studies did not investigate the mobility effect in wireless mobile ad hoc networks environments (MANETs).

None of the above mentioned studies considered evaluating TCP communication energy cost and computational energy cost in the same work. Also, using random values to represent RTT values or data packet losses do not accurately reflect the characteristics of ad hoc networks since losses and delays are correlated. In this paper, we aim to make a clear and complete comparative study for the most commonly-used TCP variants (New Reno, SACK, Vegas, and Westwood) undergoing different data packet loss situations. This study incorporates both the computational and communication energy consumption of TCP. We extended the experimental test-bed used in [7] to measure the computational energy cost by using a realistic wireless multi-hop ad hoc network emulator (SEDLANE[2] [8]) in order to enhance the quality of the obtained results. In fact, using SEDLANE allows for representing more realistic data packet losses and delays over the connection compared to what had been used so far. The decision to use test-bed experiments is based on the fact that most of the node's related characteristics (such as the computational energy cost spent within the node's CPU unit) cannot be obtained using simulators. So far, simulations allowed only for obtaining the nodes' communication energy cost.

In the following section, we present the different TCP variants studied in this work and give an overview of their basic operations.

3. Background

The Transmission Control Protocol (TCP) is a reliable transmission protocol that provides an ordered delivery of a stream of bytes over the communication link between the sender and the receiver. TCP was originally designed to be deployed within wired networks where the communicating nodes are connected through physical cables. Physical cables are reliable transmission media where congestion is the most common cause of data packet losses. That is why TCP is a congestion-control-oriented algorithm. TCP deploys flowcontrol mechanisms through its implemented algorithms (Slow-Start and Congestion Avoidance). These algorithms tend to better utilize the available bandwidth and to avoid congestion episodes over the connection by adjusting the CWND and SSThreshold parameters, to be discussed shortly.

In this section, we present the evolution of TCP since its first version that adopted a congestion control algorithm until now. We will focus on the main variants that had been designed for a wired Internet. For each variant we discuss the main drawbacks that led to the development of other variants.

3.1. TCP New Reno

TCP New Reno is the most widely deployed TCP variant. It was developed as an improvement of two preceding variants: TCP Tahoe and TCP Reno, which are described below.

- TCP Tahoe is the first TCP variant to incorporate congestion control mechanisms. These algorithms are: Slow-Start, Congestion Avoidance, and Fast Retransmit [9,10]. The goal of Slow-Start and Congestion Avoidance is to keep the congestion window (CWND) around an optimal size as much as possible. Slow-Start Fig. 1 increases the congestion window size rapidly to reach the 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 Time Out (RTO), it is regarded as lost and hence retransmitted. While, at the reception of three DUPlicate ACKnowledgments (3 DUPACKs), the first unacknowledged packet is also considered as lost. In this case, the Fast Retransmit algorithm (Fig. 2) (Fig. 3) is in charge of retransmitting the lost packet without waiting for the RTO timer to expire. This speeds up the retransmission of the lost packet.
- TCP Reno [11] is an enhancement of TCP Tahoe, by adding a new algorithm that differentiates between heavy congestion over the connection and light congestion or out-of-order packets situations, and acts accordingly to each of these cases. The congestion control mechanism of TCP Reno retains the enhancements incorporated into TCP Tahoe, but modifies the Fast Retransmit operation to include Fast Recovery [11] mechanism. Fast Retransmit and Fast Recovery are used to recover from data packet losses without the need for RTO timer expiration [12] (Fig. 4).

TCP New Reno [13] is a modification of TCP Reno congestion control algorithm. Its particularity resides on the deployment of partial acknowledgements that help notifying the TCP sender that the following segment in the sequence number is lost. This approach leads to fewer data packet retransmissions over the connection, hence better utilization of the available bandwidth. In TCP Reno, partial acknowledgements take TCP out of Fast Recovery phase by deflating the usable congestion window. In TCP New Reno, partial acknowledgements do not take TCP out of Fast Recovery. Instead, partial acknowledgements received during Fast Recovery are treated as an indication that the packet immediately following the acknowledged one in the

[2] SEDLANE: Simple Emulation of Delays and Losses for Ad hoc Networks Environment.

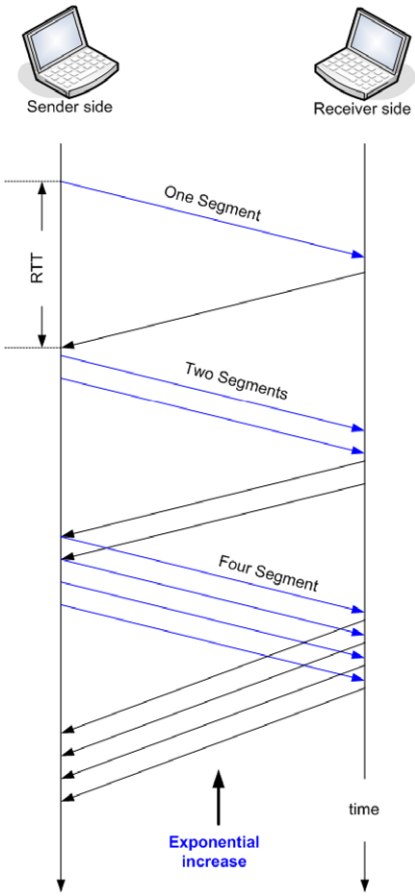

Fig. 1. TCP Slow-start mechanism.

sequence space has been lost, and should be retransmitted. Thus, when multiple packets are lost from a single window of data, TCP 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 [12]. In this way, TCP resends only the lost packets and eliminates unnecessary retransmissions.

TCP New Reno can recover from multiple losses, and is therefore more suited than TCP Reno for a wireless environment, where multiple packet losses are likely to occur during the same transmission window. However, during this recovery, the TCP New Reno sender retransmits only one lost packet per RTT, since it must wait for the partial acknowledgement from the receiver side as it does not know all the lost packets and the loss might be random (not burst). Consequently, when multiple losses occur, TCP New-Reno usually recovers after a considerable delay, which is still a major drawback [14].

3.2. 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 packet. A selective acknowledgement (SACK) option allows receivers to additionally report non-sequential data they have received. The SACK option is used within an acknowledgement packet to indicate which packets were received precisely [12] and thus allows the sender to deduce which packets had been lost. This option aims to speed up the retransmission of lost packets and avoids retransmitting the whole window of data. Adding SACK to TCP does not change the basic underlying congestion control algorithms. TCP SACK implementation preserves the properties of TCP Tahoe and TCP Reno of being robust in the presence of out-of-order packets, and uses retransmit timeouts as the recovery method as a last resort. The main difference between the TCP SACK implementation and the TCP New Reno implementation is the behavior when multiple packets are dropped from one window of data [12]. The TCP SACK sender maintains a list of segments deemed to be missing (based on all the SACKs received) and sends new or retransmitted data when the estimated number of packets in the path is less than the congestion window. When a retransmitted packet is itself dropped, the SACK implementation detects the drop with a retransmission timeout, retransmitting the dropped packet and then slow-starting. TCP SACK exits Fast Recovery under the same conditions as TCP New Reno.

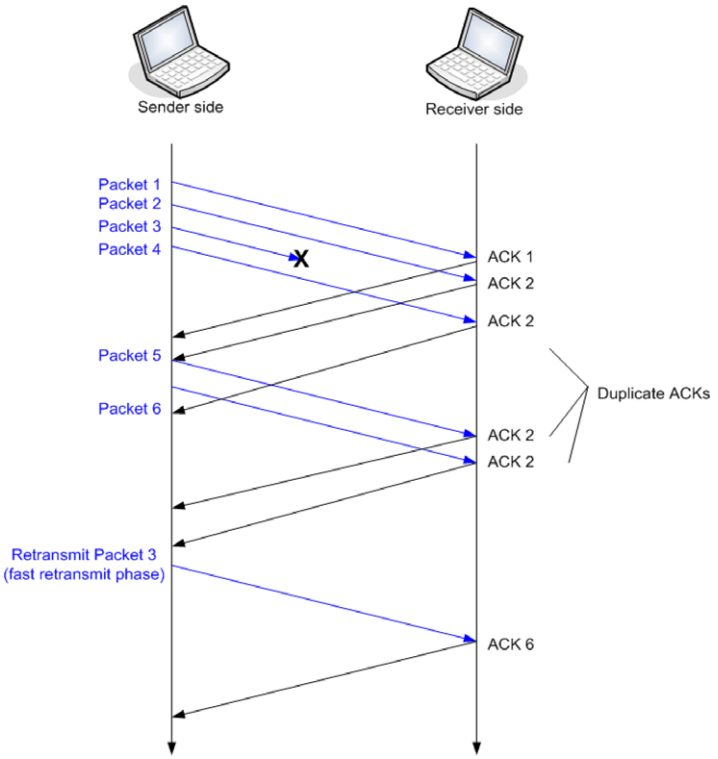

Fig. 2. TCP fast retransmit mechanism.

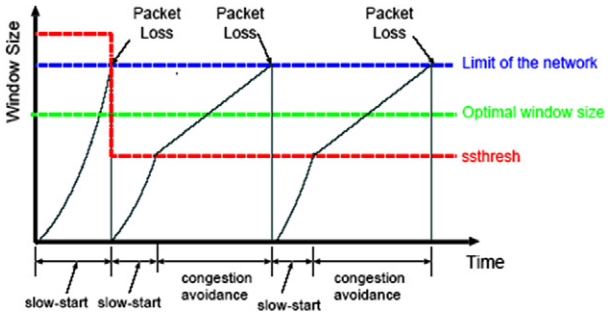

Fig. 3. Congestion Window (CWND) evolution of TCP Tahoe.

This algorithm improves the transmission of data packets over the connection, but on the other hand, it complicates the calculation process at the sender's side as it should retain a complete list of sequence numbers of all the transmitted data packets in order to deduce the numbers of lost ones when needed. This complexity might affect the overall performance of TCP over the connection.

3.3. TCP Westwood

Another way to improve the performance of TCP was to implement a bandwidth estimation algorithm as in TCP Westwood [15] variant. The bandwidth estimation algorithm opts to estimate the available bandwidth over the connection through measuring and averaging the rate of the returning acknowledgements. TCP Westwood then, adapts its data transmission rate according to the available bandwidth over the connection. This enhancement improves the performance of TCP over wireless data networks as it optimizes the usage of the available bandwidth.

TCP Westwood is a sender-side modification of the TCP congestion window algorithm that is intended to bring performance improvements to TCP Reno and TCP New 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 explanation here, as well as, our study in this work is based on the latter. The improvement is also targeted to be more significant in wireless networks with lossy links. TCP Westwood [15] relies on end-to-end bandwidth estimation to identify the cause of packet

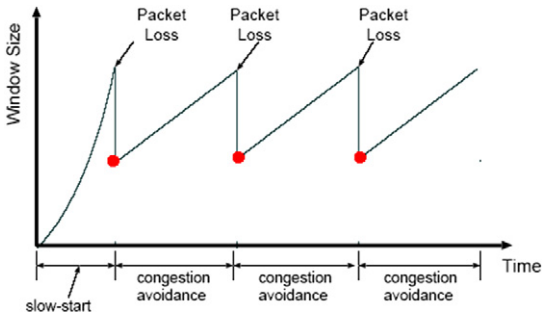

Fig. 4. Congestion Window (CWND) variation of TCP Reno.

loss (congestion or wireless channel effect), which is a major problem in TCP New Reno, and then adapts the CWND size accordingly. This loss cause identification is based on measured RTT values.

The key idea of TCP Westwood is to exploit TCP acknowledgement packets to derive bandwidth estimation and to properly set the congestion window and the slow-start threshold. By backing off to CWND and SSThreshold values that are based on the estimated available bandwidth (rather than simply halving the current values as TCP New Reno does), TCP Westwood avoids overly conservative reductions of CWND and SSThreshold; and thus resulting in achieving a higher throughput.

3.4. TCP Vegas

Another enhancement of TCP's congestion control algorithm is the network's congestion avoidance algorithm implemented within TCP Vegas [16] TCP Vegas relies on measured RTT values of the sent packets to extend Reno's retransmission mechanisms. According to this measurement, the RTO value is updated. When a duplicate acknowledgement is received, Vegas checks to see if the difference between the current time and the timestamp recorded for the first unacknowledged segment (i.e. its RTT) is greater than the timeout value. If so, then it retransmits the segment without having to wait for three duplicate acknowledgements. This change helps TCP Vegas to detect losses much sooner than TCP Reno [16] and other variants. Also, TCP Vegas uses RTT values to calculate the actual transmission rate in the network. Hence, by comparing this value with the expected throughput in the network, TCP Vegas decides how to adapt its transmission rate. TCP Vegas still contains Reno's coarse-grained timeout code as a fallback mechanism.

This enhancement improves the performance of TCP in term of throughput as it discovers the loss of data packets faster than the other variants and in turn recovers from losses faster, in the case of good estimation or measurement of the RTT value over the connection. But, in case of wrong measurement of RTT values, as when the connection starts and there is already congestion over the network links, the calculation of the data transmission rate will be wrong and might cause a persistent congestion over the connection.

In the following section, we will describe the methodology and the test-bed configuration used in order to realize our study.

4. Comparative study of TCP variants

In this section, we study the performance of the most common TCP variants (TCP New Reno, TCP SACK, TCP Vegas and TCP Westwood) mentioned earlier in terms of communication energy consumption and computational energy cost taking into consideration different data packet loss events. The loss situations studied include losses due to wireless channel errors (interferences, and signal losses), losses due to link failures between the communicating nodes, and losses due to network congestion. We mention that, both TCP Tahoe and TCP Reno are not studied in this work, since they are obsolete even within wired networks.

The performance metrics mentioned above cannot all be obtained using the same evaluation tool. The TCP's communication energy cost is obtained through simulations using Network Simulator version 2 (NS-2) [17]. However, since the TCP's computational energy cost cannot be traced using simulations (actual simulators calculate only the communication energy cost) it is measured through a purpose-built realistic test-bed set-up.

The test-bed configuration, as well as the methodology and the evaluation scenarios used to measure the computational energy cost are detailed below. But first, we will give an overview of our wireless multi-hop ad hoc network emulator (SEDLANE), used in the performance evaluation.

4.1. Overview of SEDLANE

As depicted in Fig. 5, the main idea of SEDLANE [8] is to use a hybrid evaluation approach that takes benefit from simulation results in order to enhance real test-bed experiments. It allows for configuring Dummynet [18] pipes (i.e. defining

Fig. 5. The principle of SEDLANE operation.

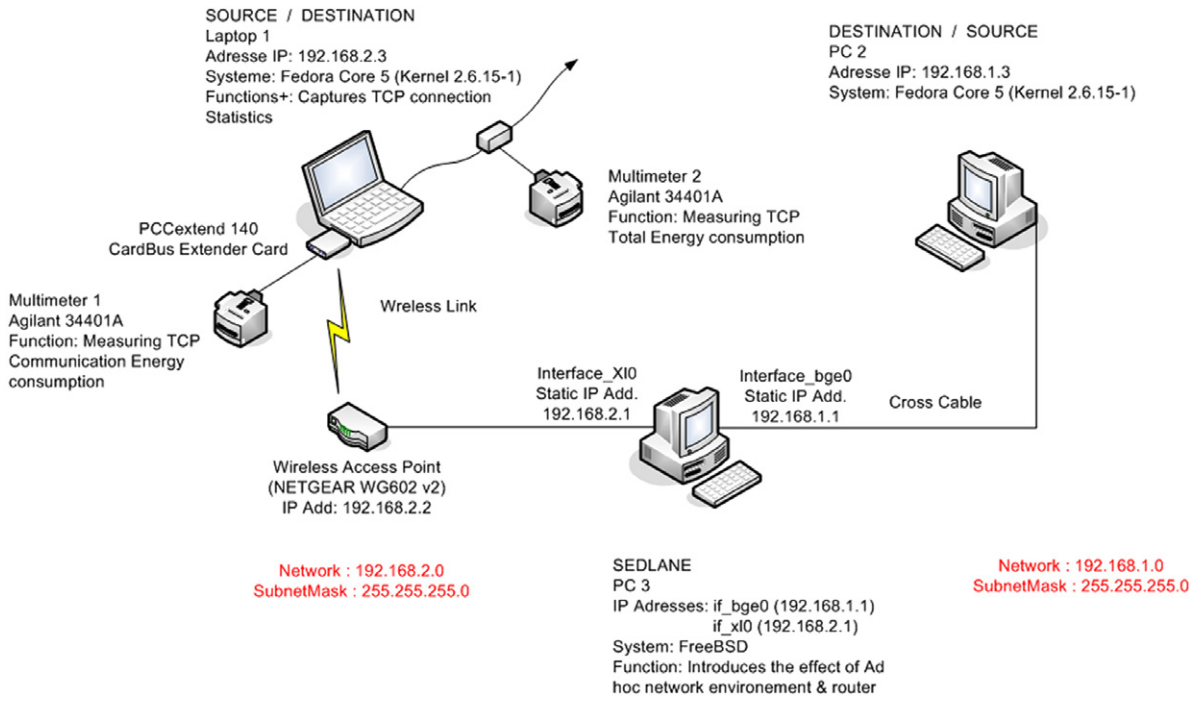

Fig. 6. TCP energy consumption measurements test-bed.

packet loss and delay rules) based on NS-2 (Network Simulator-2) [17] trace files. SEDLANE uses NS-2 TCP trace files to identify classes of packets by grouping the packets that have similar RTT values. Then, SEDLANE dedicates one pipe or communication channel for each group of packets. Hence, delay values (i.e. RTT/2 on each way) and loss rates are distributed among classes. Then, SEDLANE dynamically generates the Dummynet rules to be applied on the transmitted packets. In this way, we control the ad hoc network parameters (in terms of delays and losses) in order to make our experiments more realistic compared to those previously carried.

For more details on the SEDLANE processes, the reader can refer to [8].

4.2. Test-bed configuration and measurements methodology

For the measurements of the TCP computational energy consumption, we extended the methodology used in [7] in which we make use of SEDLANE emulator.

4.2.1. Test-bed configuration

Our test-bed configuration is shown in Fig. 6. It is composed of a DELL LATITUDE D410 laptop playing the role of the sender end side while the receiver end side is a DELL OPTIPLEX GX 520 Personal Computer (PC). Between the communicating nodes we implement SEDLANE on a second DELL OPTIPLEX GX 520 PC, to get the effect of a wireless ad-hoc network environment between the sender and receiver sides. The laptop communicates with the PC over a wireless link channel.

In order to calculate TCP energy consumption within the CPU unit, we measure both (i) the total energy consumption within the laptop, and (ii) the energy consumed within the wireless card for transmission and reception, in addition to the idle energy when no TCP or communication processes are triggered. The difference between the two measured values will be the computational energy consumption. Obviously, the measurements are taken at the TCP sender side. Synchronization is ensured between the communicating end points and the PC where the measurements were taken.

What we call the idle energy in the paper is the energy consumed at the CPU level when no TCP or communication processes are triggered. This idle energy being not due to the TCP connection, it is not taken into account by our analysis (it is subtracted from the total consumed energy). However, the additional energy (other than the idle one) that is consumed at the CPU level once a TCP connection is running is taken into account. This includes the energy consumed by all the

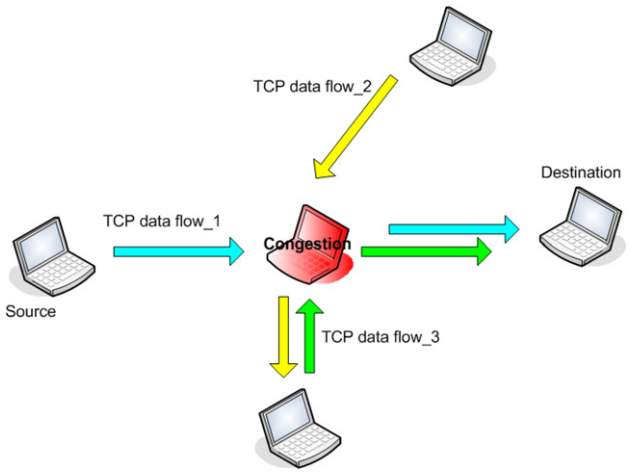

Fig. 7. Network congestion scenario.

communication mechanisms at the different layers of the protocol stack such as the energy consumed due to the use of contention resolution schemes and in-between transmission idle time (timers).

In order to match this computational energy consumption to the TCP operations, we use a minimal Linux distribution in which we turn off the display, the power manager and the x-server to minimize the effect of any other running applications on the measured current. The reason for turning off the power management as described in [8] is the fact that it helps to determine better the current draw that corresponds to the TCP code execution. Last but not least, all the processes/daemons that are not necessary to TCP operations are simply removed from the Linux distribution making it strictly minimal. By taking all these precautions, we ensured that the remaining energy consumption is due to the TCP congestion control algorithm's execution and timer adjustments. Energy consumption is determined by measuring the input voltage and current draw using two Agilent 34401A digital multimeters that have a resolution of one millisecond. We do not use the laptop's battery because avoiding the use of a battery allows for a more steady voltage to be supplied to the device [19]. In order to directly measure the current and voltage draw of the wireless 802.11b PCMCIA card, the card was attached to a Sycard PCCextend 140A CardBus Extender [20] that in turn attaches to the PCMCIA slot in the laptop. In this way, we can separately but simultaneously measure the current draw of the laptop and the current draw of the wireless 802.11b PCMCIA card.[3] Fig. 7 describes this test-bed.

Actually, using PCMCIA card as the NIC for the laptop was necessary to measure the communication energy consumption through the SYCARD. We note that the PCMCIA interface introduces additional energy consumption not to be experienced by embedded cards. However, at the level of the wireless chipsets that are used in both, these are the same and thus have a similar behavior. As it is not straightforward to measure the energy of a component that is embedded in the motherboard, the only way to do it is to relay on PCMCIA cards as NIC for the laptops. Then, we use the SYCARD, a neutral PC-extenders card in terms of energy consumption to measure the NIC energy consumption. We strongly believe that the interpretation of the results in both cases will not vary too much. Especially for the results concerning the computational energy cost of TCP.

4.2.2. TCP computational energy cost calculation

In order to calculate the TCP computational energy cost, we calculate first the total energy consumed at the laptop, and then we subtract the wireless communication energy due to data transmission/reception over the wireless network card as well as the idle energy (i.e. the energy consumed when TCP is not running). The following equations illustrate how the TCP computational energy is calculated.

First, we calculate the radio communication energy consumption (E_R):

$$E_R = I_R * T * V_R \tag{1}$$

where I_R is the measured radio current (over the wireless network card), T is the time in seconds during which the measurement are taken, and V_R is the wireless card voltage (5 V).

Then, we calculate the system's total energy consumption (E_T):

$$E_T = I_T * T * V_T \tag{2}$$

where I_T is the measured total current going into the system (the power supply current), T is the time in seconds during which the measurement are taken, and V_T is the power supply voltage (20 V).

[3] Sycard PCCextend 140 CardBus extender card is a debug tool for development and test of PC cards and hosts.

Then, we calculate the system's idle energy consumption when TCP is not running (E_{idle}):

$$E_{idle} = I_{idle} * T * V_T \tag{3}$$

where I_{idle} is the calculated idle current (deduced from the measurement data), T is the time in seconds during which the measurement are taken, and V_T is the power supply voltage (20 V).

Finally, we calculate the TCP's computational energy cost (E_{comp}):

$$E_{comp} = E_T - E_{idle} - E_R. \tag{4}$$

4.3. Evaluation scenarios and topologies

In order to have a wide range of results that help to better understand the behavior of TCP in front of different data loss situations, we define different data loss scenarios that represent the most common data packet losses over wireless ad hoc network environments. Our predefined data loss scenarios are: (i) network congestion, (ii) interference, (iii) link losses and (iv) signal losses. Before describing the implemented scenarios, let us describe the reason behind the choice of AODV (Ad hoc On-Demand Distance Victor [21]) as a routing protocol in our evaluations.

4.3.1. The implemented ad hoc routing protocol

The choice of the ad hoc routing protocol algorithm is important from two points of view: (i) its robustness and promptness to recover from a link failure, (ii) the overhead and frequency of its routing information update messages which might result in a congestion or traffic interference over the network links. For example, if the time needed by the implemented ad hoc routing protocol to recover from link failures is longer than the TCP's RTO, TCP triggers its congestion control algorithm, and backs off for a certain period of time, then enters Slow-Start phase. Also, it might happen that the routing protocol recovers from the link failure but TCP stays in the idle state, since TCP does not know about the link recovery. On the other hand, if the time taken by the ad hoc routing protocol is lower than TCP's RTO, TCP may recover from data packet loss without entering Slow-Start phase and decreasing its CWND to minimum. Moreover, the overhead of ad hoc routing update messages could aggravate a congestion situation over the TCP connection. This leads to more congestion control actions triggered to recover from the packet losses.

In order to compare the start-up and route recovery time of different ad hoc routing protocols, we have studied their performance through simulations, and the results are shown in Table 1. Table 1 discusses the start-up time, i.e. the time needed by the ad hoc routing protocol to build up its routing information table (in case of proactive protocol) or finding a new route (in case of reactive protocol) in order to start communicating, as well as, the route recovery time needed after a link failure. The comparison is shown for four main ad hoc routing protocols: Ad hoc On-Demand Distance Victor (AODV) routing protocol [21], the Dynamic Source Routing (DSR) protocol [22], the Destination-Sequenced Distance Vector (DSDV) routing protocol [23], and the Optimized Link State Routing (OLSR) protocol [24]. The values depicted in Table 1 allow us to recall that in reactive ad hoc routing protocols (AODV and DSR), the routing protocol triggers its route discovery process only when the sender has to send data towards the destination or when a route used is broken. Contrarily, proactive protocols (DSDV and OLSR) need a longer time to build their routing table and also to recover from a route failure. This is due to the fact that they build their routing tables for the whole network before any communication request can be triggered.

Our main concern is to evaluate TCP performance over MANETs. Therefore, we decided to run the best MANET configuration. So, according to the table above, we choose to run our simulations using the Ad hoc On-Demand Distance Victor (AODV) ad hoc routing protocol. Although the figures in the above table show that DSR has a shorter route recovery time than AODV due to its caching feature, we found, through simulations, that it has a higher routing messages overhead than AODV, since any intermediate node has the right to answer to the route discovery request. Also, the caching feature on the intermediate nodes sometimes causes problems when responding by stale routes that are in their cache. Hence, our decision to implement AODV since it has the lowest routing messages overhead [21]. Extending our performance evaluation to other routing protocols is outside the scope of this paper, though, this would constitute an interesting evaluation study.

We discuss in the following sections the simulations scenarios used by SEDLANE in order to emulate the behavior of a wireless multi-hop ad hoc network environment in our experiments.

4.3.2. Simulation scenarios

The TCP simulation scenarios, using NS-2, are implemented as follows:

(1) *Creating network congestion:* In this packet-loss model, we create a congested node at the middle of a five node topology. This is done by generating three TCP data traffic flows that must pass by this intermediate node to reach the other communicating end. Fig. 7 illustrates this simulation scenario. One should also note that, different levels of data congestion can be generated by controlling the number of TCP data flows crossing this particular network node at a certain time.

(2) *Interference between neighboring nodes:* Fig. 8 illustrates this scenario in which two TCP connections are on-going in parallel. The main TCP connection (TCP data flow 1 in Fig. 8) is disturbed by the interferences generated by the second

Table 1
Comparative study of ad hoc routing protocols.

Ad hoc routing protocol	Start-up time (s)	Route recovery (s)
AODV	∼0.03	∼1
DSDV	∼90	∼31
DSR	∼0.07	∼0.2
OLSR	∼6	∼7

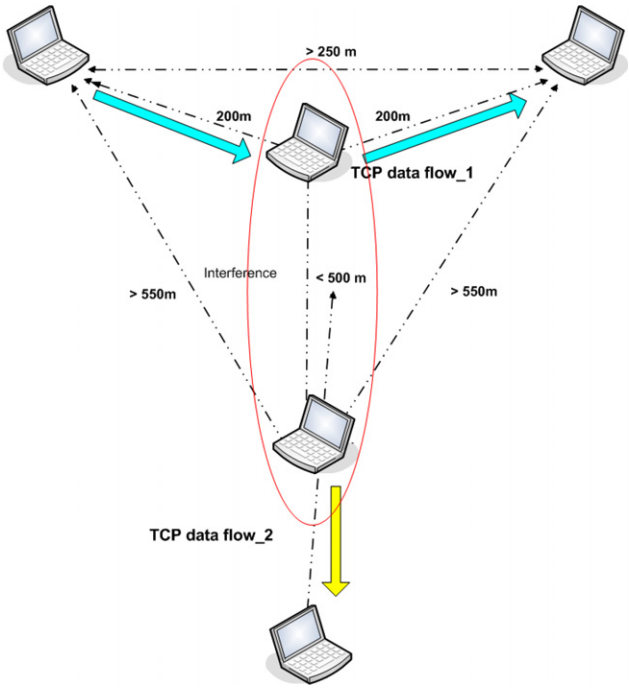

Fig. 8. Interference scenario.

TCP connection (TCP data flow 2 in Fig. 8). Indeed, the node acting as a forwarder for the main TCP connection is placed within the interference range of the second TCP connection sender. This situation creates interference and thus data packet drop.

(3) *Link failure and communication route changes:* In this model we force TCP to change its communication path by shutting down the intermediate node between the communicating end points. In addition, we imply routes with different numbers of hops (Fig. 9). Thus, once TCP changes the communication route, the characteristics of the path between the communicating nodes change. It is obvious that the choice and the establishment delay of the new route will be dependent on the implemented ad-hoc routing protocol. Packet losses and delay changes will also be implied by the link loss and the characteristics of the new recovered route.

(4) *Wireless signal loss:* This scenario illustrates the situation where the wireless signal is not stable. The communicating nodes lose the connection due to signal loss then they resume the communication when the signal comes back. As shown in Fig. 10, signal losses are generated by moving one of the intermediate nodes out of the radio range of its connection neighbor for a while and then moving it back.

5. TCP energy consumption results

Contrarily to previous works that concentrated on the operating system, hardware and device-level energy consumption due to TCP [7], the objective of our analysis is to analyze the energy cost of each TCP function and variant in order to facilitate improving their behavior in MANETs. So, in the following we first analyze the computational energy cost of the main TCP functions: Slow-Start, Fast Retransmit/Fast Recovery and Congestion Avoidance. Then, we make a comparison of the different TCP variants in terms of both computational and communication energy cost. This is realized according to the different data packet loss scenarios applied: network congestion, interference, link loss, or signal loss. We note that the computational energy cost is measured at the sender's side, since most of TCP calculations (CWND and RTO calculations) are done at the TCP sender node. Hence, the computational energy cost in this section is measured and calculated according to the total number of transmitted data bytes.

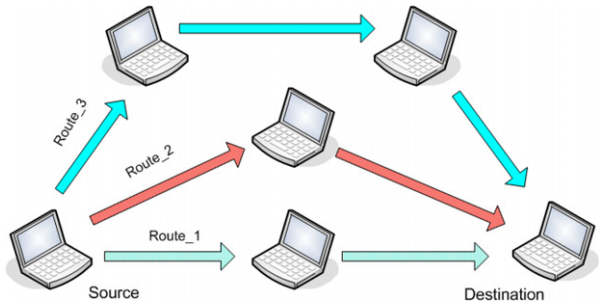

Fig. 9. Link failure and communication route changes.

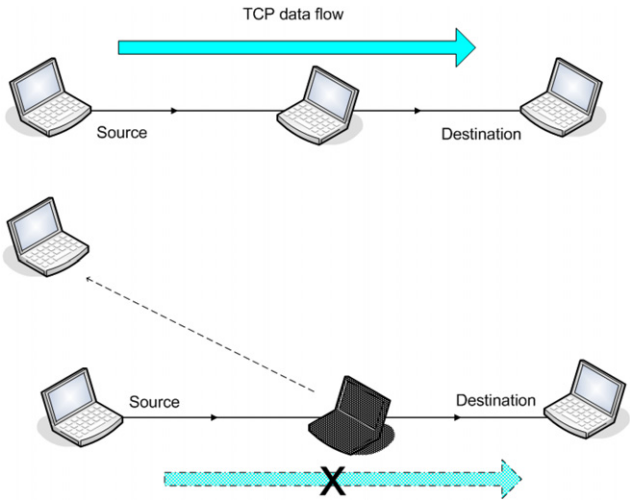

Fig. 10. Wireless signal loss scenario.

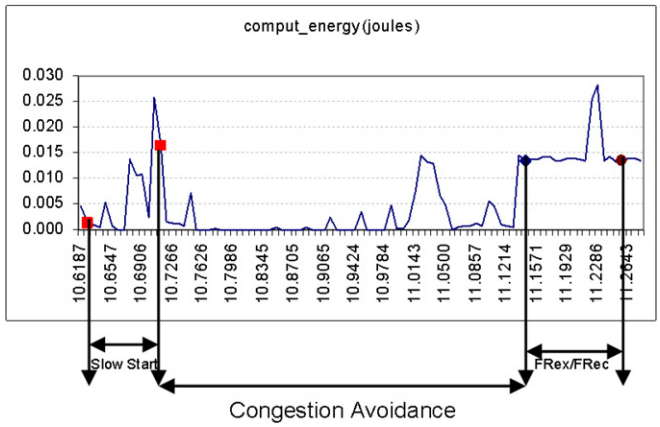

Fig. 11. TCP New-Reno computational energy cost example.

5.1. TCP functions computational energy cost

Fig. 11, shows an example of TCP New Reno computational energy cost when faced with packet losses due to network congestions. The Figure shows that, the computational energy cost, in joules, of the system is higher during Fast Retransmit/Fast Recovery phase compared to that of both Slow Start and Congestion Avoidance phases for the reasons mentioned above.

In order to obtain the energy consumption of the main TCP functions (Slow-start, Fast Retransmit/Fast Recovery, and Congestion Avoidance), we use log files at the sender side to log the start and end times of each TCP function. Then, we use this information to match the energy consumption by each function using the energy consumption measurement record.

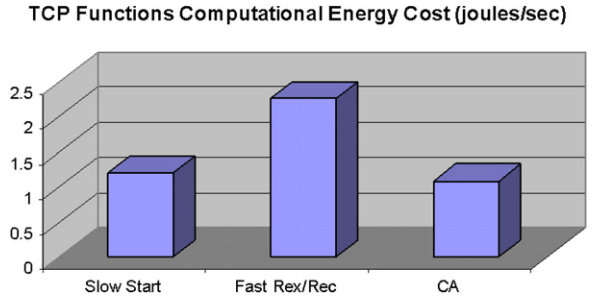

Fig. 12. TCP computational energy cost (J/s).

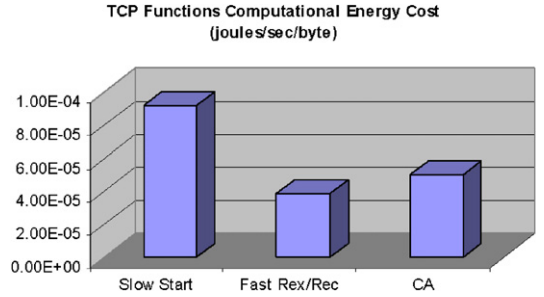

Fig. 13. TCP computational energy cost per sent byte (J/s).

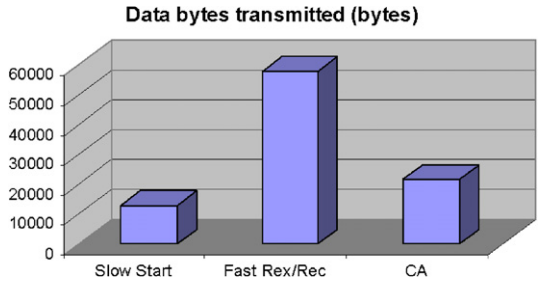

Fig. 14. Data bytes transmitted per TCP function.

The results show that the computational energy cost of the Fast Retransmit/Fast Recovery phase is extremely high compared to that of both the Slow-Start and Congestion Avoidance phases (Fig. 12). Indeed, Fig. 12 shows that the energy consumption is almost doubled. However, this is no longer the case when the energy consumption is calculated per the amount of data sent by TCP (Fig. 13). This is due to the fact that the TCP Fast Retransmit/Fast Recovery process consumes an important amount of energy when triggered but it does so for a short period of time during which it may send several TCP segments in one burst. During the Congestion Avoidance, TCP increases its transmission rate by one segment each RTT. Here, TCP has a regular throughput and computational overhead that are lower than the one of Fast Retransmit/Fast Recovery phase (Figs. 12 and 14). However, it has a higher energy consumption per each sent byte compared to the Fast Retransmit/Fast Recovery phase (Fig. 13).

5.2. Computational and communication energy cost of TCP variants

This section aims at comparing the performance of the four TCP variants studied (New Reno, SACK, Vegas, and Westwood) in terms of computational and communication energy cost. This comparison is conducted taking into consideration different packet loss situations encountered in wireless multi-hop ad hoc networks as explained in Section 4.3.2. We use AODV as an ad hoc routing protocol within our simulations and test-bed experiments as explained earlier.

5.2.1. Network congestion results

In order to isolate the effect of network congestion from the other packet loss reasons, we use a static ad hoc network without route changes but in which network-access contentions are created. The results depicted in Fig. 15 demonstrate that TCP Vegas has the highest computational energy cost per each sent byte among all the other variants. This is due to the fact that TCP Vegas is a variant that tries to avoid congestion. In order to achieve this, TCP Vegas continuously performs

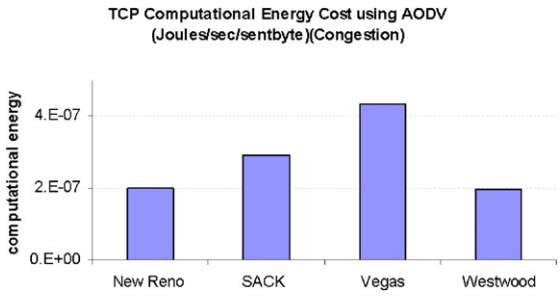

Fig. 15. TCP computational energy cost (Congestion case study).

complex calculations in order to adapt its TCP transmission parameters with each received acknowledgement (ACK). This certainly leads to some degree of reliability in some cases. However, this behavior costs a lot in terms of processing that translates into a higher computational energy cost compared to all the other variants studied (Fig. 15).

On the other hand, we notice that TCP Westwood performs better in terms of computational energy cost, because it modifies its transmission parameters only when there is a data packet loss over the connection and not continuously as in TCP Vegas. This involves less computational overhead, despite the probable increase of the number of retransmissions compared to TCP Vegas. We also note, in Fig. 15, that TCP Westwood and New Reno have almost the same performance in terms of energy consumption per each sent byte despite that the loss ratio is higher with TCP New Reno in the case of congestion. From that we can conclude that the light computational cost (i.e. the one due to Fast Recovery/Fast retransmit process) of resending packets by TCP New Reno is neutralized by the computational overhead introduced by TCP Westwood (i.e. loss analysis to identify the packet loss cause).

From Fig. 16, we can see that TCP Westwood has the best performance among the variants in terms of communication energy cost. The ability of TCP Westwood to adapt its data transmission rate according to the estimated bandwidth leads to savings in terms of energy consumption. Whereas, TCP New Reno and TCP SACK, both resume the communication, after a congestion event, staring from the minimum data transmission rate (1 segment). 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 the router's queue is building up and that congestion is about to happen, so it triggers its congestion avoidance mechanisms. This will lead to faster recovery from packet losses. However, the high communication energy cost of TCP Vegas comes from the fact that TCP Vegas detects the would-be losses much sooner than TCP New Reno and the other variants, then retransmits the would-be lost packet after receiving the first duplicate acknowledgement. In some cases, such as during congestion periods, the delay over the connection increases due to buffering, leading to a mistaken action, as the packet is only delayed and not lost. In this case, retransmitting the assumed to be lost packet contribute to an unnecessary increase in the communication energy consumption. Indeed, the simulations results show that, TCP Vegas has no lost bytes over the connection compared to other variants. However, the precipitance of TCP Vegas to recover from losses results in a high number of unnecessary retransmissions.

Finally, we note that though TCP SACK has the ability to resend the lost data packets faster than TCP New Reno due to the Selective ACK option; Fig. 15 demonstrates that its cost is higher. Indeed, SACK implies an important overhead in terms of processing and storage in order to extract the numbers of lost data packets at the sender side. This leads to high energy consumption.

5.2.2. Traffic interference results

In order to isolate the effect of network traffic interference from the other packet loss reasons, we use a static ad hoc network without route changes. Comparing Figs. 15 and 17, we can note that, in general, the computational energy cost is increased in the case of interferences compared to the case of congestion. This is obvious since TCP had been designed to deal with congestion situations. The misbehavior of TCP in front of data packet losses due to other effects such as interference leads to a higher computational energy cost.

Referring to Fig. 17 we can see that TCP Vegas has the worst performance in terms of computational energy cost compared to the other studied TCP variants. This is due to the same reasons as explained in the previous sections. As for TCP Westwood, though it has the lowest loss ratio compared to other TCP variants; its computational energy consumption is higher than that of both TCP New Reno and TCP SACK. This is due to the algorithms used by TCP Westwood to adapt its performance according to network conditions and their continuous triggering by packet losses. TCP Westwood recalculates and modifies its data transmission rate after each data packet loss; this is made often in case of interferences. Note that in case of congestion, TCP Westwood has the same performance as TCP New Reno, while in case of losses due to wireless errors its behavior becomes more complex without significant improvement to the throughput.

Interestingly, we found that TCP New Reno and TCP SACK have almost the same performance in terms of computational energy cost. Although the number of retransmitted data with TCP SACK is less than that in TCP New Reno, the processing overhead of TCP SACK neutralizes the advantages gained while using selective acknowledgements.

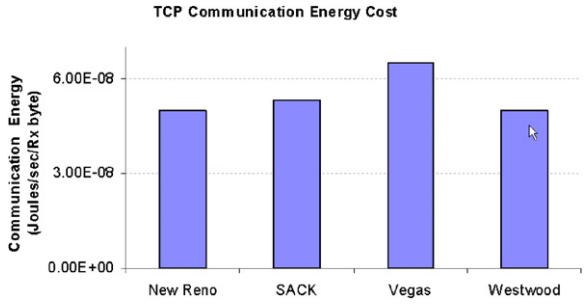

Fig. 16. TCP communication energy cost (Congestion case study).

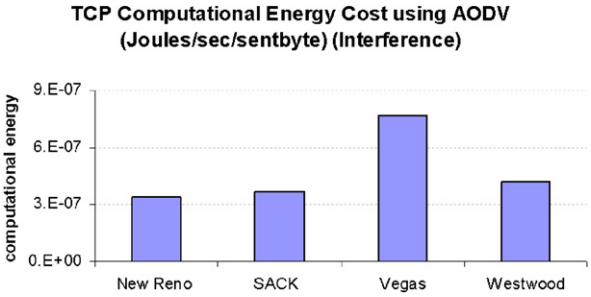

Fig. 17. TCP computational energy cost (Interference case study).

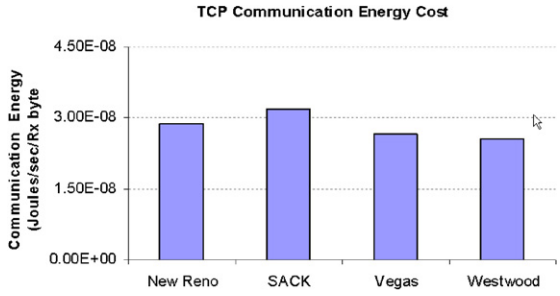

Fig. 18. TCP communication energy cost (Interference case study).

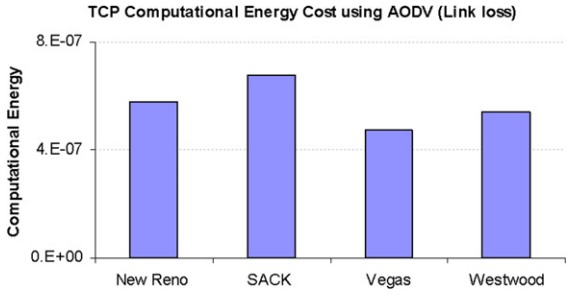

Fig. 19. TCP computational energy cost (Link loss case study).

For the communication energy cost results, both TCP Vegas and TCP Westwood outperform TCP New Reno in terms of communication energy cost, since both variants have the ability to adapt their CWND after data packet loss according to the network conditions as can be verified from Fig. 18. This makes them losing fewer data packets in the network.

5.2.3. Link failure and communication route changes results

Fig. 19, shows that the computational energy cost of most TCP variants increases compared to the two studied scenarios above. This is an expected observation because TCP as it is nowadays was not designed to cope with network link failures. In network link failure situations, we must expect burst packet losses to which TCP has an aggressive reaction.

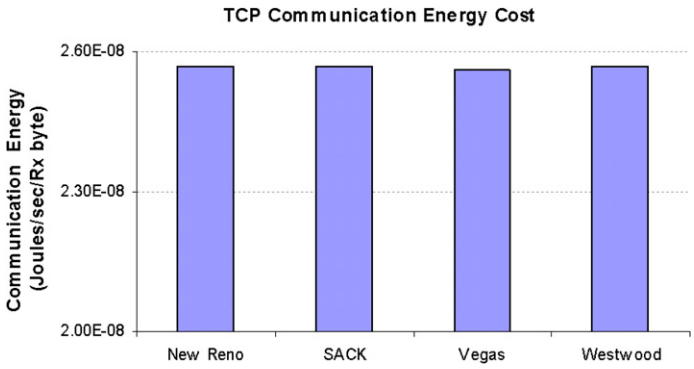

Fig. 20. TCP communication energy cost (Link loss case study).

In case of link failure, we notice that all TCP variants in almost all cases identify the data packet loss through TCP Retransmission Time-Out (RTO). Since they are not designed to cope with such situations (link losses), they all react similarly: i.e. consider the packet loss as if it were due to strong network congestion and trigger the Slow-Start process. As mentioned earlier, the Slow- Start process is the least efficient one in terms of energy cost. Let us note here that, theoretically, triggering the Slow-Start phase is not necessary as the packet loss cause is not strong congestion.

If we look now into each variant separately, we conclude that TCP Vegas and TCP Westwood can be considered as the first and second most performing variants respectively. This is because both variants have the ability to rapidly readjust the data transmission rate over the connection according to the characteristics of the new recovered route. However, TCP New Reno or TCP SACK prove to be less rapid in that aspect and thus they consume more energy by staying in the Slow-Start phase and slowly increasing their throughput.

In addition, simulation results also show that all TCP variants, in case of link failure events, detect data packet losses through RTO. Thus, they all perform similarly by backing off and entering Slow-Start phase (even if they stay in this state for different durations). This can be seen in Fig. 20. The four TCP variants are shown to have comparable performance levels in terms of communication energy cost. The similar behavior of all the studied TCP variants confirms that none of them is able to deal with link failure loss situations over the connection.

5.2.4. Wireless signal loss results

Intermittent loss of the radio signal is another cause for end-to-end communication disruption. Signal loss can be a result of encountering geographical obstacles or unfavorable weather conditions. This results in data packet losses over the TCP connections. Wireless signal loss can be considered as a special case of link failure. In fact, we consider here the special case when the signal is lost between two communicating end points; there is no way to resume the communication session unless the signal is restored. Thus, signal loss might be viewed as a network partitioning case where the communicating end points are totally disconnected from each other. The main difference between link failure and signal loss models is the ability to resume the communication session after the signal loss using the same route (that also has to be re-established by the routing protocol). In the link loss case, both nodes, the sender and the receiver, would search for another route to complete the session. While in the signal loss case, this is topologically not possible. After signal loss recovery, the TCP sender will start the communication session from the beginning, (i.e. from Slow-Start phase). This will be the case, each time the communicating nodes get disconnected in the absence of wireless signal. Recall that in this loss scenario, signal losses are frequent. That implies that almost all TCP variants stay most of the connection's lifetime in the Slow-Start phase. In addition, TCP data packet losses would be identified through RTO expiration.

Fig. 21 demonstrates that TCP SACK is the worst performing variant in terms of computational energy cost, since it requires calculations at the sender's side to identify lost packets. In addition, the sender's side must keep a copy of all the sent data packets in case of a need to resend them. For TCP Westwood, after each data packet loss episode over the connection, it calculates and adjusts its CWND and SSThreshold parameters. This, therefore, leads to more calculations at the sender's side and consequently more computational energy consumption than both TCP New Reno and even TCP Vegas in this case. TCP Vegas identifies the losses as due to heavy congestion over the connection, and triggers its congestion control algorithm. This could be considered the right action in this case. Abstaining from resending data packets that would never reach the destination is an acceptable action. Thus, the trade-off between complexity and amount of data sent makes TCP Vegas the most energy-efficient. TCP New Reno can be considered as the second most performing variant in terms of computational energy cost due to its simplicity, since it stops data transmission and enters Slow-Start after signal recovery. In TCP New Reno, no complex calculations are triggered.

Regarding the communication energy cost, Fig. 22, show that all TCP variants, studied in this work, have almost the same performance in terms of communication energy consumption. They all back off for a certain period of time then restart the communication (after the wireless signal comes back) from the Slow-Start phase.

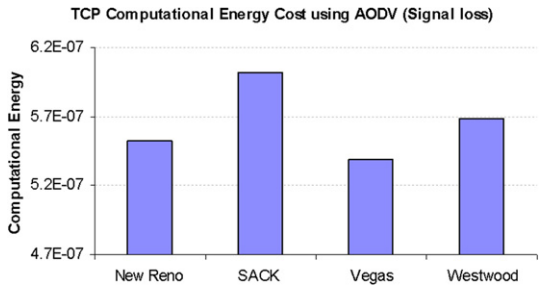

Fig. 21. TCP computational energy cost (Signal loss case study).

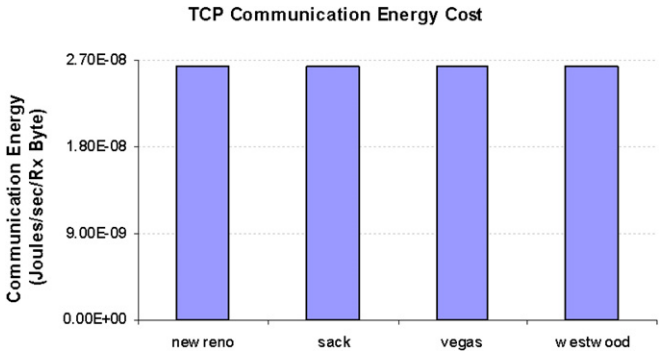

Fig. 22. TCP communication energy cost (Signal loss case study).

Finally, it is worth noting that we have studied the signal loss event using different levels of signal loss durations, ranging from few seconds to few dozens of seconds, and in all cases, the four TCP variants had the same behavior as the one depicted in Figs. 21 and 22.

We notice also, from the above results, that the computational energy consumption is higher than the communication energy cost. Actually, it is not the first time that this result is proved. In [7], the authors (B. Wang and S. Singh) had also concluded that. In fact, the communication energy cost becomes higher than the computational energy cost only when the number of hops is high (i.e. the sum of communication energy consumed at each hop becomes higher that the TCP-computational energy cost that is consumed only at the end points of the TCP connection).

6. Discussion

It was shown that the congestion control algorithm in TCP variants affects the energy consumption within wireless multi-hop ad hoc networks. In order to enhance the TCP's performance within wireless multi-hop ad hoc networks, we proposed to first identify the different types of data packet loss situations within such networks, and then to study the performance of existing TCP variants when dealing with such data packet loss models. Our ultimate goal was to investigate the most appropriate TCP reaction in order to recover from each different data packet loss cause, taking into consideration optimizing the usage of limited and scarce network resources such as energy resources.

We found that the ability of TCP to differentiate the data packet loss cause over the connection (as in TCP Westwood) helps to react better and recover from data packet loss according to the identified loss cause. Unfortunately, TCP Westwood is only able to recognize congestion-related and wireless-related (e.g. interference) packet loss causes. In addition, the ability to adjust TCP's data transmission rate after a loss episode (as in TCP Vegas) can enhance its performance within wireless multi-hop ad hoc networks.

The problem of TCP and its most existing variants within wireless multi-hop ad hoc networks resides in its inability to distinguish between different data packet loss causes. Thus, TCP reaction is not always optimum which would lead to network performance degradation and resource waste. From this perspective, we argue that designing a new Loss Differentiation Algorithm (LDA) and a new Loss Recovery Algorithm (LRA) that are able to cope with most common data loss causes within wireless multi-hop ad hoc network is necessary. These new algorithms should have the capability to identify as well as to deal with these common packet loss models within such networks environment, and having as objective enhancing the performance of TCP in terms of energy consumption (both communication and computational).

Table 2 summarizes the results shown and described above in this paper.

Table 2
Summary of the comparative study.

Test scenario	TCP New Reno	TCP SACK	TCP Vegas	TCP Westwood	Comments/why
Network congestion	Best	Worst	Worst	Average	The light computational overhead of TCP New-Reno and its adequate reactions led to this result.
Interference	Best	Average	Worst	Average	The light computational overhead of TCP New-Reno and its adequate reactions led to this result.
Link failure	Average	Worst	Best	Average	The ability of TCP Vegas to readjust its performance parameters over the connection according to the characteristics of the new recovered route.
Signal loss	Average	Worst	Best	Average	TCP Vegas identifies the losses as due to heavy congestion over the connection, and triggers its congestion control algorithm. This could be considered the right action in this case.

7. Conclusion

In this paper, we conducted a complete performance study of four TCP variants (TCP New Reno, TCP SACK, TCP Vegas and TCP Westwood), within wireless ad hoc multi-hop network environments. We started by identifying the different data packet loss situations that TCP may confront within wireless multi-hop ad hoc networks: (i) network congestion, (ii) interference, (iii) link failure, and (iv) signal loss. Our study concerns both TCP computational energy and communication energy cost. The TCP's communication energy cost results were obtained through NS-2, while the computational energy cost results were obtained through a hybrid approach (i.e. using simulation results to configure a realistic test-bed and perform accurate experiments).

The obtained results show that TCP cannot cope with all data packet loss situations found within wireless ad hoc network. TCP's behavior needs to be enhanced in order to handle the different loss causes other than congestion (link failure, signal loss, and interference). When looking at the reaction of TCP when faced with non-congestion loss types, we can notice high degradation in performance leading to waste of scarce network resources; such as nodes' available energy (batteries). The existing TCP variants that were originally developed for wired networks do not always behave optimally when confronted with the different data packet causes that characterize wireless ad hoc networks. Some show good performance in certain cases and bad performance in others. It is worth noting that, when these variants were developed caring about the nodes' energy consumption was not a big concern. Also, the variants that were developed with loss differentiation capabilities, as TCP Westwood, to enhance TCP performance within wireless networks do not take into consideration other data packet loss situations that would be faced within wireless ad hoc networks, such as link failures.

We found, also, that the complexity of the implemented congestion control algorithms of TCP variant and their failure to cope adequately with certain loss causes may lead to unnecessary energy wastage at the node's level. The experiments' results show that the phase Fast Retransmit/fast recovery is the most efficient in terms of computational energy cost with respect to the transmitted data when compared to both Slow-Start and Congestion Avoidance phases. In addition, the simulation results show that unnecessary data retransmissions due to misinterpretation of the loss cause leads to increasing the communication energy consumption.

Finally, we also identified some tracks to follow in order to create a novel TCP variant that is energy-efficient in MANETs. Knowing where the most TCP energy consumption is spent is the main key to improving TCP functions and performance within MANETs. For example, helping TCP to avoid unnecessary retransmissions or reducing TCP CPU calculations would help to minimize TCP's energy consumption. In future work, we will develop this new TCP variant for MANET. This one should have an optimized behavior in regards of the different TCP performance parameters in such environments: throughput, radio-energy cost and the computational energy cost.

References

[1] H. Singh, S. Singh, Energy consumption of TCP Reno, New Reno, and SACK in multi-hop wireless networks, in: Proceedings of ACM SIGMETRICS'02, June, 2002.
[2] A. Seddik-Ghaleb, Y.M Ghamri Doudane, S.-M. Senouci, A performance study of TCP variants in terms of energy consumption and average goodput within a static Ad Hoc environment, in: Proceedings of ACM International Wireless Communications and Mobile Computing Conference, IWCMC'06, July, 2006.
[3] A. Seddik-Ghaleb, Y.M Ghamri Doudane, S.-M. Senouci, Effect of Ad Hoc routing protocols on TCP performance within MANETs, in: IEEE International Workshop on Wireless Ad-hoc and Sensor Networks, IWWAN'06, June 2006.
[4] A. Seddik-Ghaleb, Y.M. Ghamri-Doudane, S.M. Senouci, TCP computational energy cost within wireless Mobile Ad Hoc Network, in: 7th ACS/IEEE International Conference on Computer Systems and Applications, AICCSA-2009, Rabat, Morocco, May, 2009.
[5] S. Agrawal, S. Singh, An experimental study of TCP's energy consumption over a wireless link, in: 4th European Personal Mobile Communications Conference, Febreuary, 2001.
[6] V. Ramarathinam, M.A. Labrador, Performance analysis of TCP over static wireless ad hoc networks, in: Proceedings of ISCA 15th International Conference on Parallel and Distributed Computing Systems, PDCS'02, September, 2002.
[7] B. Wang, S. Singh, Computational energy cost of TCP, in: Proceedings of IEEE INFOCOM'04, March, 2004.

[8] A. Seddik-Ghaleb, Y.M Ghamri Doudane, S.-M. Senouci, Emulating end-to-end losses and delays for ad hoc networks, in: Proceedings of IEEE International Conference on Communications, ICC'07, June, 2007.
[9] M. Allman, V. Paxon, W. Stevens, TCP Congestion Control, RFC 2581, IETF, April, 1999.
[10] V. Jacobson, Congestion avoidance and control, ACM SIGCOMM'88 symposium on communications architectures and protocols, vol. 18, no. 4, August, 1988.
[11] V. Jacobson, Modified TCP congestion avoidance algorithm, end2end-interest mailing list, April, 1990. ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail.
[12] K. Fall, S. Floyd, Simulation-based comparison of Tahoe, Reno, and sack TCP, ACM Computer Communications Review 5 (3) (1996).
[13] S. Floyd, T. Henderson, A. Gurtov, The NewReno modification to TCP's fast recovery algorithm, RFC 3782, IETF, April, 2004.
[14] M. Michel, L.S. Nelson, F José, On the performance of TCP loss recovery mechanisms, in: Proceedings of IEEE International Conference on Communications, vol. 3, May, 2003, pp. 1812–1816.
[15] S. Mascolo, C. Casetti, M. Gerla, M.Y. Sanadidi, R. Wang, TCP Westwood: Bandwidth estimation for enhanced transport over wireless links, in: 7th Annual International Conference on Mobile Computing and Networking, ICMCN'01, July, 2001.
[16] L.S. Brakmo, S.W. O'Malley, Larry L. Peterson, TCP Vegas: New techniques for congestion detection and avoidance, ACM SIGCOMM'94, August, 1994, pp. 24–35.
[17] NS-2. http://www.isi.edu/nsnam/ns/.
[18] Dummynet. http://info.iet.unipi.it/luigi/ip_dummynet/.
[19] P. Gauthier, D. Harada, M. Stemm, Reducing power consumption for the next generation of pdas: It's in the network interface, in: Proceedings of MoMuC'96, Septembre, 1996.
[20] Sycard technologies, Sycard technologies, pccextend 140 cardbus extender, July, 1996. [Online], HYPERLINK http://www.sycard.com.
[21] C.E. Perkins, E.M. Royer, Ad-hoc on-demand distance vector routing, in: proceedings of 2nd IEEE Wksp. Mobile Comp. Sys. And Apps, WMCSA'99, February, 1999.
[22] D.B. Johnson, D.A. Maltz, Dynamic source routing in Ad-Hoc wireless networks, in: T. Imielinski, H. Korth (Eds.), Mobile Computing, Kluwer Academic Publishers, 1996, pp. 153–181.
[23] C.E. Perkins, P. Bhagwat, Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers, Comp. Comm. Rev. (Oct.) (1994) 234–244.
[24] Thomas Clausen, Comparative study of routing protocols for Mobile Ad-Hoc NETworks, INRIA Research report, RR-5135, March, 2004.