UNIVERSITÉ D'EVRY
VAL D'ESSONNE

TCP Performance Study and Enhancements within Wireless Multi-hop Ad Hoc Network Environments

UNIVERSITÉ D'EVRY VAL D'ESSONNE
LABORATOIRE DE RÉSEAUX ET SYSTÈMES MULTIMÉDIA

THESIS

For the award of the PhD degree:

DOCTEUR DE L'UNIVERSITE D'EVRY – VAL D'ESSONNE

Discipline: Networking

By

M^{rs} Alaa GHALEB-SEDDIK

TCP Performance Study and Enhancements within Wireless Multi-hop Ad Hoc Network Environments

30 March 2009

Jury

Pr. Tijani CHAHED Reviewer	Institut National de Télécommunications (INT) ***(France)***
Pr. Véronique VEQUES Reviewer	Université Paris-Sud 11 ***(France)***
Dr. Bachar EL HASSAN Examiner	Faculty of Engineering, Lebanese University ***(Lebanon)***
Pr. André-Luc BEYLOT Examiner	ENSEEIHT, Laboratoire IRIT ***(France)***
Pr. Nazim AGOULMINE Director	Université d'Evry Val d'Essonne ***(France)***
Dr. Yacine GHAMRI-DOUDANE Supervisor	Ecole Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise ***(ENSIIE) (France)***
Dr. Sidi-Mohammed SENOUCI Supervisor	Orange Labs Lannion ***(France)***

DEDICATION

To my father, who had been the source of my motivation and inspiration, whose presence I feel all the time though he's no longer among us.

To my mother, who encouraged and cared for me.

To my husband whose backing and support never faltered along the way.

To my dear daughters who, though I wasn't always there, kept loving me every day.

ACKNOWLEDGEMENT

This work would have never been possible without all those wonderful people who surrounded me and supported me every step in the way.

I would like to express my deep and sincere gratitude to my supervisor, Dr. Yacine GHAMRI-DOUDANE, Professor at ENSIIE. I've been fortunate to benefit from his wide knowledge and deep experience. He's been always driving me to pursue excellence and go beyond what I deemed possible. His understanding, encouragement, and personal guidance were what made it possible to complete this thesis.

I warmly thank Dr. Sidi-Mohammed SENOUCI, Orange Labs, for his valuable advice and continuous support. I'm grateful for his help in presenting and coordinating my work with Orange Labs that lead to the successful delivery of the research project to France Telecom.

I am very grateful to Professor Dr. Nazim AGOULMINE, Head of LRSM, University of Evry, for his detailed and constructive comments, and for his significant support and guidance throughout this work.

I would like to thank Professor Véronique VEQUE, University Paris-Sud 11, and Professor Tijani CHAHED, INT, for accepting to review my work and for their constructive comments.

Also, I would like to thank Dr. Bashar EL HASSAN, Faculty of engineering, University of Libanon, and Professor André-Luc BEYLOT, ENSEEIHT, IRIT Laboratory, for accepting to evaluate my work.

I also want to thank my colleagues with whom I collaborated and for whom I have great regard, and I wish to extend my warmest thanks to all those who have helped me with my work in LRSM, ENSIIE, and the Univeristy of Evry.

My thanks go as well to France Telecom's Orange Labs, who provided scientific and financial sponsorship for this research and provided all the means to have it realized.

I owe my loving thanks to my family, my husband Khaled SEDDIK and my daughters Rana and Noura. Without my husband’s encouragement, understanding, and the love of my daughters, it would have been impossible for me to make this achievement.

Evry, France, March 2009

Alaa GHALEB

RÉSUMÉ

Les réseaux sans fil ad hoc diffèrent des réseaux filaires traditionnels principalement par la multitude de perturbation auxquels ils sont sujets. Alors qu'une rupture de lien est un événement plutôt rare sur des réseaux filaires, et généralement imputable à l'état physique du matériel intermédiaire (câbles, routeurs, etc.), cet événement est courant avec les communications radio. Ceci peut être lié à la qualité du signal reçu de l'autre extrémité ou à la configuration de l'environnement (épaisseur et quantité des obstacles intermédiaires, perturbations électromagnétiques…). De plus, même si les perturbations causées par l'environnement ne mènent pas toujours à une rupture d'un lien, elles peuvent aussi avoir un impact sur la réception sans erreurs des données. Cette volatilité des liens est typique dans les réseaux sans fil alors que pour les réseaux traditionnels filaires ce problème est inexistant.

Le protocole TCP, qui est prévu pour assurer la transmission fiable des données, n'a été conçu qu'en tenant compte des contraintes des réseaux filaires. Ainsi, certains événements dans la transmission de données sans fil peuvent être mal interprétés et engendrer une mauvaise réaction de la part de TCP. Cette mauvaise interprétation affaiblit les performances plus qu'elle ne l'améliorerait.

Pour réduire le fossé de performance dont souffre TCP dans les réseaux sans fil ad hoc, l'objectif de cette thèse est double. Dans un premier temps, une étude complète des performances de TCP dans les réseaux ad hoc est dressée. Celle-ci concerne à la fois les débits atteignables mais aussi la consommation d'énergie induite par l'utilisation de ce protocole de transport dans un réseau sans fil ad hoc. Cette étude permet d'identifier les points d'amélioration du protocole TCP pour qu'il soit utilisable dans les réseaux sans fil ad hoc. Dans un second temps, nous proposons une nouvelle variante de TCP, appelée TCP-WELCOME, dont l'objectif est de traiter de façon adéquate les différents types de perte de paquets sur un réseau ad hoc sans fil et par la même optimiser la performance de TCP.

ABSTRACT

Wireless ad hoc networks are differentiated from traditional wired networks by the multitude of data packet loss situations that they are subjected to. This is due to the intrinsic characteristics of the wireless channel (e.g. signal fading, interference, obstacles, and environment effects) that might obstruct the proper reception of data packet at the other communication end. Moreover, in some case, these vulnerabilities of the wireless channel can result in a complete link failure. Although link failure is of low probability in wired networks since physical cables constitute the data transmission media, it is rather common wireless networks (due to nodes' mobility, battery depletion, obstacles, or some other wireless-channel-related effect). The volatility of the communication channel is a typical problem with wireless links, which is not the case with wired cables.

TCP is a transport protocol that aims at ensuring high reliability and guaranteed reception of data packets. However, TCP was designed primarily for wired networks to address network congestion, which is the main cause for data packet loss in wired networks. Therefore, other types of data packet loss encountered in wireless networks are prone to misinterpretation by TCP, which, in turn, will lead to degradation in the performance of TCP within the network.

In order to overcome the performance limitation of TCP when used within wireless ad hoc networks, the aim of this thesis is twofold. First, a complete performance evaluation study of TCP over wireless ad hoc networks is achieved. This evaluation deals with two performance metrics: the achievable throughput and the energy consumption of TCP within wireless ad hoc networks. This study allows identifying the potential room of improvement to enhance TCP efficiency in wireless ad hoc networks. Second, we propose a new TCP variant we called TCP-WELCOME, that optimizes the performance of TCP in wireless ad hoc networks through its ability to distinguish among, and efficiently deal with, different data packet loss situations, encountered within wireless ad hoc networks.

GLOSSARY

Ad Hoc A wireless, autonomous communication mode that does not require infrastructure configuration.

Acknowledgement, ACK A packet message, used in the Transmission Control Protocol that is sent by the receiver to acknowledge the sender the receipt of a certain packet(s).

Congestion A saturation of communication links due to much data transmitted simultaneously by the network's nodes. This results in both data packet loss and extra transmission delays over the connection.

Congestion Window, CWND The maximum amount of data bytes that can be sent out over the connection without being acknowledged.

Congestion Avoidance A phase where TCP enters after Slow-Start phase. In this phase, the TCP CWND increases slowly in order to avoid network congestion.

Duplicate ACK, dupack TCP receivers generate a duplicate acknowledgment when out-of-sequence segment is received. This one acknowledges only some of packets outstanding at the start of the Fast Recovery.

MANET A type of wireless ad hoc network that is self-configuring network of mobile devices connected through wireless links. Each MANET device is free to move independently.

Segment A TCP segment is the packet of information that TCP uses to exchange data with its peers. TCP receives data from a data stream, segments it into chunks, and adds a TCP header creating a TCP segment.

Slow Start Threshold, SSThresh The slow-start threshold that is used to define the transition from slow-start phase to congestion avoidance phase.

Round Trip Time, RTT RTT is the delay that corresponds to the time needed by the TCP sender after sending a data packet to receive its acknowledgment.

Retransmission Time out, RTO RTO is the maximum time that TCP waits for the data acknowledgment before declaring that the packet is lost.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

1.1 MOTIVATIONS

The last decade has seen a tremendous boom in the mobile communications market. People have become accustomed to the convenience of making calls with mobile phones and browsing the Internet with notebooks via wireless connections. Two prime examples of this development are the great success of the Global System for Mobile Communication (GSM) and the huge number of Wireless Local Area Networks (WLANs) deployments. In the future, more and more mobile devices will become networked, following the trend toward ubiquitous networking. One of the main goals in current research is to design new wireless networks that are flexible, low-cost, and require little administration. In this context, the principle of ad hoc networking received much interest during the past five years. In an ad hoc network, mobile devices communicate with each other in a peer-to-peer fashion; the nodes are independent and self organized. Each node of the network acts as a server, client, and forwards data packets for the other nodes of the network. This feature is not supported in current systems for wired and mobile communications. Hence, the existence of each node in an ad hoc network is of highly importance for the others in order to ensure the network connectivity.

In parallel to that growth, most of the applications are designed to be compatible with TCP. This makes TCP the most famous protocol in the TCP/IP protocol stack with more than 90% of the Internet traffic today. TCP was developed to be implemented initially taking into consideration the characteristics of wired infrastructure networks. Its congestion control algorithm and the data flow mechanism it employs makes it performing well in such networks. Through controlling the number of data packets transmitted over the connection and taking into consideration the advertised data to be received at the receiver side, TCP has the ability to manage the utilization of the available bandwidth. The success of TCP within the wired networks, in addition to the wide variety of applications that are compatible with it, made TCP the first transport protocol to be used within almost all the deployed data networks.

Taking into consideration the differences between wired and wireless (infrastructure and infrastructure-less) data networks, we expect that TCP performance would differ when applied within each of them. Many researches [1] [2] [3] were done recently to help understand the performance of TCP within wireless cellular and wireless ad hoc networks. These researches reveal the problems that can be found within wireless networks and do not exist within wired ones, such as wireless channel errors and link failure problems. These problems represent new challenges to TCP since it was not originally developed to deal with underlying issues. In fact, TCP was developed to solve the problem of data packet loss due to congestion within network, and originally is a congestion-control-oriented protocol. This was logical since the main cause of data packet losses within wired network is the congestion. On the other hand, within wireless (infrastructure and infrastructure-less) networks, there are many other causes of data packet losses. Indeed, wireless ad hoc networks represent a highly challenging environment for TCP as it suffers from new causes of data packet losses that TCP was not designed to deal with. Actually, TCP underperforms drastically within wireless ad hoc networks. The researches done recently [4] show that the performance of TCP is highly impacted when used within wireless ad hoc networks, since, again, it was not designed to cope with wireless networks data packet loss causes. Therefore, it was our conclusion that TCP should be able to:

1- Distinguish and cope with different data packet loss models over the connection,
2- Recover from data packet losses in such a way to enhance its performance over such networks.

These TCP enhancements are among our major contributions within this thesis. Our final objective here is to present a new variant of TCP that is most adapted to the nature of wireless ad hoc networks and their most common data packet loss causes.

As stated above, one of our final aims is to enhance the performances of TCP when this one is running over a wireless ad hoc network. The first question that rises from this is which performance metrics should be considered? In the wired Internet, the main performance measure parameter for TCP is the throughput that may be achieved by TCP while avoiding congestions. In wireless ad hoc networks, this performance measure cannot be the only one used. Indeed, since ad hoc network's nodes are battery-operated, another performance metric is to be considered by any solution targeting such networks: the energy consumption. Contrarily to a common thought, this latter is not only due to the communication aspects (i.e. the energy spent by the network interface for data transmission and reception). The energy consumption is also due to computational cost of any implementation. Indeed, some heavy programs may lead to high energy consumption within the CPU unit while lighter programs leads to less computational energy cost. Hence, the total energy cost of any solution should include its communication energy cost as well as computational energy cost. It is thus important that our new TCP variant maximizes the reached throughput while minimizing the total energy cost as defined above.

1.2 METHODOLOGY AND CONTRIBUTIONS

Measuring all the performance metrics is not always possible using network simulators. Indeed, some nodes' related performance metrics are not implemented into network simulators which are designed to deal with communication effects rather than node-specific effects. Among those effect that are not possible to measure through simulation we can find the computational energy cost of a particular protocol. Today, the evaluation of this parameter is only possible through real implementation experiments. On the other hand, building a real test-bed configuration of wireless ad hoc networks is expensive especially when configuring or deploying networks of a scale beyond a dozen of nodes. Emulation appears, then, as a good compromise between these two approaches: simulation and real test-bed setup. So, in order to be able to evaluate any performance metrics that is not available through simulations, especially the computational energy cost of an application or a protocol, we proposed, implemented and evaluated SEDLANE, which stands for "Simple Emulation of Delays and Losses for Ad hoc Network Environments". SEDLANE, which constitutes the first contribution of this thesis, is a new emulator that allows evaluating the performances of any application or protocol running at the transport layer or over it.

SEDLANE emulates data packets losses and delays over the connection and manipulates the real data packets traffic in order to generate the same effect as within wireless ad hoc networks. SEDLANE has the ability to emulate a wide range of wireless ad hoc network scenarios of any scale with only small number of physical machines. Our proposed emulator, SEDLANE, is used for studying the computational energy cost of the existing TCP variants and their congestion control mechanisms (Slow-Start, Fast Retransmit/Fast Recovery, and Congestion Avoidance).

This constitutes the first step towards a complete study of the existing TCP variants performance. This complete study targets the evaluation of both throughput and energy consumption (communication energy consumption and computational energy consumption). This study is conducted in order to evaluate the performance of TCP in detail when confronted with different data packet loss situations within wireless ad hoc environments. The second contribution of this thesis concerns this complete performance study of different TCP variants in terms of both throughput and energy consumption. This performance study is conducted in two phases. The first phase is a performance study through a network simulator (NS-2). Through this study, we evaluated the connection throughput and the nodes' communication energy cost. The second phase is an experimental evaluation that uses SEDLANE to emulate the effect of wireless ad hoc network environment. This evaluation allowed measuring the computational energy cost of the studied TCP variants and their main TCP congestion control algorithms.

The discussion of the obtained results allowed us to identify the undesired behavior of TCP that leads to underutilizing the network resources. The obtained results, also, helped us to define the best performance requirements of TCP to deal with each data packet loss situation over an ad hoc network. The new TCP variant is developed according to the conclusions gained from this performance study. It gave us some ideas about how to enhance its behavior.

The problem of TCP within wireless ad hoc networks comes from its inability to distinguish between the different data packet loss models within the network. This often leads to an aggressive reaction from TCP when faced with a data packet loss that is not due to congestion. Indeed, dealing with any data packet loss as if it were due to congestion results in resource waste both at the network and nodes' levels. This waste is represented by low bandwidth utilization and higher energy consumption. Therefore, in order to enhance the performance of TCP within wireless ad hoc networks, it is obvious that TCP should be able to identify each data packet loss cause and react accordingly, triggering the most suitable loss recovery action that optimizes both the network and node's resources.

The third contribution of this thesis is TCP-WELCOME, which stands for "TCP variant for Wireless Environment, Link-losses and COngestion packet loss ModEls", a new TCP variant for wireless ad hoc networks. TCP-WELCOME is able to distinguish among and recover from the following data packet loss situations that we identified: (i) link failure, (ii) wireless channel related errors, and (iii) congestion. It includes a Loss Differentiation Algorithm (LDA) that is able to distinguish among the above mentioned data packet loss situations, and a Loss Recovery Algorithm (LRA) that is capable to recover from each of these different data packet loss situations. TCP-WELCOME is developed with the aim to optimize TCP performance from the following points of view. Maximizing the utilization of the available bandwidth (throughput), while minimizing the nodes overall (communication and computational) energy consumption. The performance evaluation of TCP-WELCOME, through simulations and SEDLANE enhanced experiments, allowed us to confirm that it allows optimizing TCP performance.

Finally, a side contribution of this thesis concerns the introduction of the TCP computational energy cost within NS-2. With our implementation of TCP's computational energy cost, NS-2 now has the ability to calculate the overall TCP node's energy consumption (including both communication and computational types).

1.3 PLAN OF THE THESIS

The organization of this thesis represents the evolution of our work as explained above.

The second chapter discusses the evolution of TCP and its congestion control algorithm. The object of this chapter is to show the development of TCP and its characteristics that made it favorable to be implemented and used within data networks. Also, we show the problems that TCP faces when implemented within wireless cellular and ad hoc networks. This leads to a discussion about the improvement made to TCP to enhance its performance within these environments. This chapter states the work done in the domain and shows the problems that still have to be taken into consideration to make TCP a suitable transport protocol for wireless ad hoc networks.

The third chapter discusses the possibility of evaluating the performance of new networks applications and end-to-end protocols over wireless ad hoc network environments using emulation, and presents the new wireless ad hoc emulator SEDLANE. This chapter explains the main idea of SEDLANE, its characteristics, its algorithms, and its functionalities. At the end of the chapter, we test and validate our emulator using a realistic test-bed configuration and prove that SEDLANE is capable of emulating the characteristics of an entire ad hoc network, more precisely the experienced delays and losses over wireless links using a small and an inexpensive network configuration.

In the forth chapter, we conduct a complete performance study of different variants of TCP. In this study, three TCP performance parameters are evaluated: TCP connection throughput, TCP communication energy consumption, and TCP computational energy consumption. The first two parameters were evaluated through NS-2. While the computational energy cost was measured through a realistic test-bed configuration. In this chapter also, we show how SEDLANE can be used in a realistic test-bed configuration in order to measure the TCP computational energy cost of a wireless ad hoc network node. The results obtained and the underlying discussions in this chapter are the guidelines that led to the design of our TCP variant as explained in the fifth chapter. Also, in the same chapter, we discuss the extension of the NS-2 energy model through introducing the computational energy cost of TCP within it.

In the fifth chapter, we discuss the need of a new TCP variant that is most adapted for wireless ad hoc networks. This is realized according to the results obtained from the performance study in the previous chapter. We present the main requirements of this variant in order to enhance the performance of TCP within such networks from two points of view: (i) network's throughput, and (ii) nodes' total energy consumption. We describe the main algorithms of this new variant that we called TCP-WELCOME. We also show, the evaluation results of TCP-WELCOME demonstrating that it outperforms the other studied TCP variants both in terms of throughput and energy consumption.

The last chapter concludes the work done during this thesis, and lists some perspectives that were identified for future work.

CHAPTER 2. EVOLUTION OF THE TRANSPORT CONTROL PROTOCOL

2.1 INTRODUCTION

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 considered as reliable transmission media where congestion is the most common cause to data packet losses. That's why TCP is a congestion-control-oriented algorithm. TCP deploys flow control mechanism 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 through the CWND and SSThresh parameters.

In the following, we will discuss the history of evolution of the existing TCP congestion control algorithm and the different algorithms implemented in order to enhance its performance. The discussion will be based on two main characteristics:

1. The available bandwidth utilization (TCP bandwidth).
2. The complexity of the implemented algorithms and its effect on TCP performance.

We will show, latter through this thesis that this second parameter is of a great interest when dealing with TCP performance enhancement over wireless ad hoc networks.

The rest of this chapter is organized as follows: we start by providing an overview of the evolution of TCP congestion control algorithm and the main TCP variants developed for wired networks. Then, we discuss quickly the performance of TCP within wireless infrastructure networks. After that, we present TCP performance issues within wireless ad hoc networks environment and the proposed enhancements of TCP within such environments. Finally, we summarize our ideas in this chapter and conclude it.

2.2 TCP CONGESTION CONTROL ALGORITHM

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

2.2.1 TCP TAHOE

TCP Tahoe is the first TCP variant to incorporate congestion control mechanisms. Its implementation added a number of new algorithms and refinements to earlier implementations. Mainly, these algorithms and refinements are: Slow-Start, Congestion Avoidance, and Fast Retransmit [5] [6].

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 (Figure 2.1) increases the congestion window 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 Time Out (RTO), it is regarded as lost and is retransmitted. On the other hand, 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 (Figure 2.2) 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. Finally, note that in both situations, the packet retransmission is followed by a reduction of both the SSThresh, to CWND/2, and the CWND to its minimum (CWND=1 segment). The Slow-Start phase is then triggered. Figure 2.3 summarizes the CWND variation of TCP Tahoe at both Slow-Start and Congestion Avoidance phases. It also shows how the congestion window and the SSThreshold are adapted according to the network conditions.

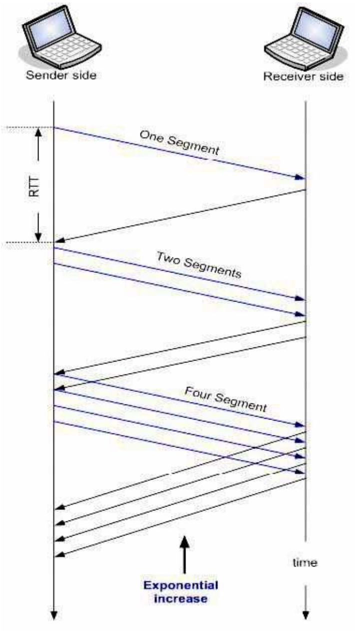


FIGURE 2.1. TCP SLOW-START MECHANISM

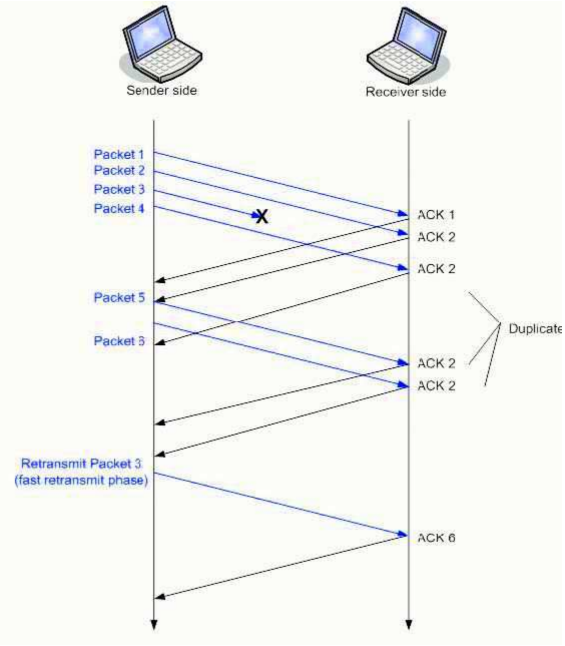


FIGURE 2.2. TCP FAST RETRANSMIT MECHANISM

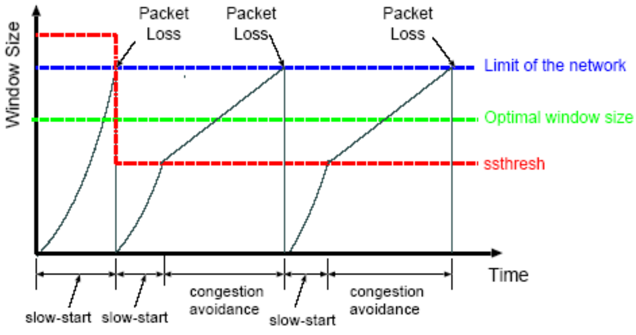


FIGURE 2.3. CONGESTION WINDOW (CWND) EVOLUTION OF TCP TAHOE

In case of data packet losses or out of order packets, TCP Tahoe triggers its congestion control algorithm. TCP retransmits the lost data packets and shrinks its congestion window (CWND) to minimum (CWND = 1 segment). This might be the right action in case of congestion but with out of order packets this is considered as waste of network resources. Congestion losses is of a burst nature (many packets are lost at a time), thus decreasing the data transmitted over the connection would be the right action to recover from the losses. However, out of order packets could be the result of load balancing mechanisms or routing protocols that implements multi-path routing approaches for instance. The packets reach the destination in an out of order manner, but all the packets are received by the receiver. The problem with TCP Tahoe is that, it retransmits the out of order packets and triggers its congestion control algorithm unnecessarily. This leads to unnecessary data transmission rate reduction over the connection. This way, the reliability of TCP in this case results in waste of the available bandwidth utilization and more energy consumption in order to increase its CWND. Regarding the complexity of the implemented congestion control algorithm, we find that it is so simple and do not require much calculations at the CPU unit as it reduces blindly the data transmission rate to one segment after each data loss.

2.2.2 TCP RENO

The problem of TCP Tahoe was addressed and solved in TCP Reno [7] variant, 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 [7] mechanism. The Slow-Start and the Congestion Avoidance algorithms are used by TCP Reno sender to control the amount of data injected into the network while Fast Retransmit and Fast Recovery are used to recover from data packet losses without the need for RTO timer expiration [8]. Fast Retransmit/ Fast Recovery [6] [5] algorithm is triggered in case of the arrival of three duplicate acknowledgements (i.e. that acknowledge the same packet) at the sender side. TCP considers, in this case, that there is a light congestion conditions over the connection and halves its data transmission rate instead of decreasing it to minimum as in TCP Tahoe. As can be noticed from Figure 2.4, TCP Reno recovers more quickly than TCP Tahoe, since it will not shrink the CWND to minimum each time it will encounter a packet loss as the case in TCP Tahoe (Figure 2.3). By doing so, it is expected that TCP Reno will enhance the network performance.

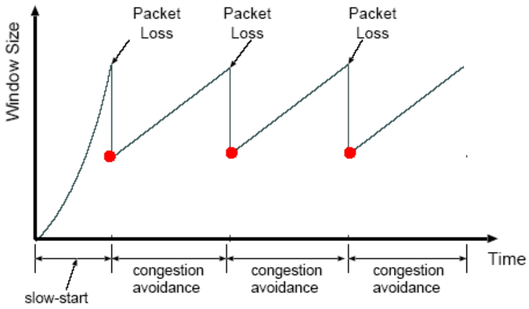


FIGURE 2.4. CONGESTION WINDOW (CWND) VARIATION OF TCP RENO

This mechanism of data loss recovery in TCP Reno is shown to perform better than TCP Tahoe, when a single packet loss occurs in one data segment (random error). Fast

Retransmit/Fast Recovery proved to enhance the performance of TCP in terms of throughput in wired networks, since it tends to better utilize the network resources and avoids unnecessary data packets retransmissions as possible. However, it can suffer from performance problems when multiple data packets are lost in one data segment (burst error). In this case the CWND will be significantly reduced and may return to its original size only after a considerable delay [8].

Additionally, the problem with TCP Reno comes from the fact that each time there is a loss over the connection; it retransmits all the CWND that contains the lost packet. This leads to high number of unnecessary data retransmissions over the connection causing a low throughput. In addition, with frequent random data packet losses, TCP Reno reduces its data transmission rate frequently and returns to perform as in Slow-Start phase leading to low throughput over the connection. Hence, TCP Reno tends to enhance the performance in case of random data packet losses (only one segment lost from the CWND), but it cannot cope with multiple data packets lost from the same CWND over the connection.

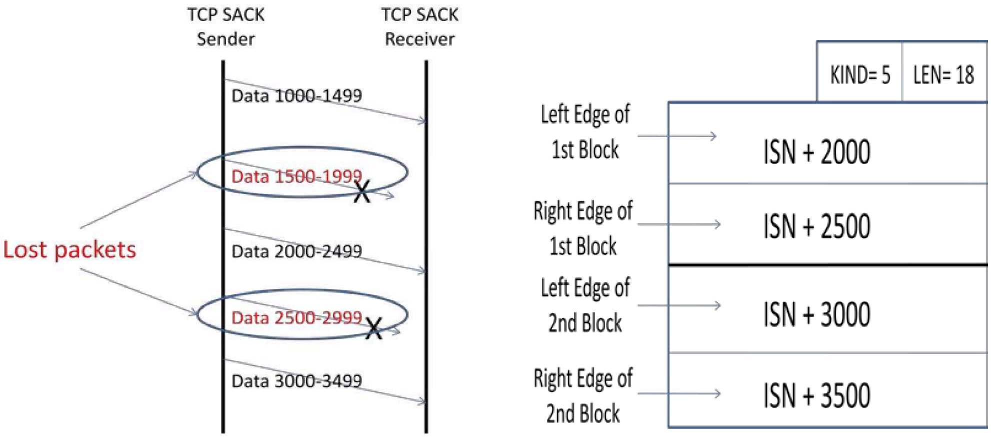
2.2.3 TCP NEW RENO

Another modification of TCP Reno congestion control algorithm is TCP New Reno [9]. TCP New Reno deploys partial acknowledgements that help to notify the TCP sender that the following segment in the sequence number is lost. This approach leads to less data packets retransmissions over the connection, hence better utilization of the available bandwidth. In TCP Reno, partial acknowledgements take TCP out of Fast Recovery by deflating the usable window back to the size of the congestion window. TCP Reno, then, retransmits all the data packets within the same segment and enters Slow-Start phase that leads to unnecessary data packets retransmissions. 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 packet in the 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. TCP New-Reno remains in Fast Recovery until all of the outstanding data, ever since Fast Recovery was initiated, are acknowledged [8]. This way the TCP New Reno sender avoids to retransmit all the CWND that contains the lost packets; it resends only the lost ones and eliminates the unnecessary retransmissions. Also, this approach enables TCP to better deal with the problem of multiple random data losses within the same CWND, as it does not decrease its data transmission rate after each lost packet. Instead, it decreases its data transmission rate only once at the beginning of the Fast Retransmit/ Fast Recovery phase. TCP New Reno stays in this phase until retransmitting all the lost data packets.

TCP New Reno can recover from multiple losses, and is therefore more suited than TCP Reno for the mobile wireless environment, where multiple data packet losses are likely to occur during the same transmission window. However, during this recovery, the TCP New Reno sender retransmits only one 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 [10].

2.2.4 TCP SACK

Traditional implementations of TCP use an acknowledgement number field that contains cumulative acknowledgement, indicating that the TCP receiver has received all of the data up to the indicated byte. 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 [8] 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 of last resort. The main difference between the TCP SACK implementation and the TCP New Reno implementation is the behaviour when multiple packets are dropped from one window of data [8]. 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. Figure 2.5 demonstrates an example of TCP SACK option.



(A) TCP SACK CONNECTION (B) TCP SACK OPTION

FIGURE 2.5. EXAMPLE OF TCP SACK OPTION

The SACK option of TCP will acknowledge the packets that have been correctly received and the sender will deduce the lost packets from the received SACK coming from the receiver side. In the above Figure, as an example, the receiver acknowledges that the packets from 2000 to 2500 and those from 3000 to 3500 have been duly received. Therefore, upon receiving this acknowledgment, the TCP sender realizes that the packets between 1500 and 2000, as well as, those between 2500 and 3000 are lost.

This algorithm improves the transmission of data packets over the connection, but on the other side, it complicates the calculation process at the sender 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.

2.2.5 TCP WESTWOOD

Another way to improve the performance of TCP was to implement a bandwidth estimation algorithm as in TCP Westwood [11] 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 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 explanation here, as well as, our study in this thesis is based on the latter. The improvement is also targeted to be more significant in wireless networks with lossy links. TCP Westwood [11] relies on end-to-end bandwidth estimation to identify the cause of packet 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.

When the pipe size, calculated by multiplying the estimated bandwidth (*BWE*) by the minimum RTT value (*RTT min*) is smaller than CWND, it is more likely that packet losses are due to congestion. This is because the connection is using a CWND value much higher than its share of pipe size, thus congestion is likely to be the cause of packet losses. On the other hand, when "*BWE×RTTmin*" > *CWND*, it indicates that wireless channel errors are more likely to be the cause of the packet losses.

The key idea of TCP Westwood is to exploit TCP acknowledgement packets to derive rather sophisticated measurements as follows. First, the source performs an end-to end estimation of the bandwidth available along a TCP connection by measuring and averaging the rate of returning acknowledgements. Second, after a packet loss episode (i.e. the source receives three duplicate acknowledgements or a timeout) the source uses the measured bandwidth to properly set the congestion window and the slow-start threshold. By backing off to CWND and SSThresh 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 SSThresh; and thus resulting in achieving higher throughput.

2.2.6 TCP VEGAS

Another enhancement of TCP's congestion control algorithm is the network congestion avoidance algorithm implemented within TCP Vegas [12]. TCP Vegas relies on measured RTT values of 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. 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. Notice that the congestion window should only be reduced due to losses that happened at the current sending rate, and not due to losses that happened at an earlier, higher rate. In TCP Reno, it is possible to decrease the congestion window more than once for losses that occurred during one RTT interval. In contrast, TCP Vegas only decreases the congestion window if the retransmitted segment was previously sent after the last decrease. Any losses that happened before the last window decrease do not imply that the network is congested for the current congestion window size, and therefore, do not imply that it should be decreased again. This change is motivated by the fact that TCP Vegas detects losses much sooner than TCP Reno [12].

First, TCP Vegas sets "*BaseRTT*" to the minimum of all measured round trip times (*RTT*); it is commonly the RTT of the first segment sent by the connection, before the router queues increase due to traffic generated by this connection. If we assume that we are not overflowing the connection, then the "*Expected*" throughput is given by:

$$Expected = WindowSize / BaseRTT$$

Where, "*WindowSize*" is the size of the current congestion window, which we assume for the purpose of this discussion, to be equal to the number of bytes in transit.

Second, TCP Vegas calculates the "*Actual*" sending rate. This is done by recording the sending time for a distinguished segment, recording how many bytes are transmitted between the times that segment is sent and its acknowledgement is received, computing the RTT for the distinguished segment when its acknowledgement arrives, and dividing the number of bytes transmitted by the sample RTT. This calculation is done once per round-trip time.

Third, TCP Vegas compares "*Actual*" to "*Expected*", and adjusts the window accordingly. Let $Diff = Expected - Actual$. Note that *Diff* is positive or zero by definition since "*Actual*> *Expected*" implies that we need to change "$BaseRTT$" to the latest sampled RTT. Also, TCP Vegas, defines two thresholds, "$\alpha < \beta$", roughly corresponding to having too little and too much extra data in the network, respectively. When "$Diff < \alpha$", Vegas increases the congestion window linearly during the next RTT, and when "$Diff > \beta$", Vegas decreases the congestion window linearly during the next RTT. TCP Vegas leaves the congestion window unchanged when "$\alpha < Diff < \beta$".

Intuitively, the farther away the actual throughput gets from the expected throughput, the more congestion there is in the network, which implies that the sending rate should be reduced. The β threshold triggers this decrease. On the other hand, when the actual throughput rate gets too close to the expected throughput, the connection is not utilizing the available bandwidth. The α threshold triggers this increase. The overall goal is to always keep between α and β extra bytes in transit.

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 if 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 addition, TCP Vegas performs badly in case of route changes over the network. Actually, if the route changes over the connection and the measured RTT value increases, TCP will not be able to distinguish if that increase is due to congestion or due to route changes. This way, TCP will decrease its data transmission rate interpreting that increase as if is due to congestion. This action is considered as a waste of bandwidth over the connection. Also, an important computational complexity is added for RTT and RTO calculations and adjustments at the reception of each acknowledgement.

2.3 TCP WITHIN WIRELESS MOBILE NETWORKS

As stated earlier, TCP had been initially designed in order to cope with congestion-related packet losses in wired data networks. However, nowadays the networks include more and more wireless links. In the following, we discuss briefly the problems of TCP performance within wireless mobile network environments, and the proposed solutions to overcome these problems.

Wireless mobile networks are infrastructure networks that use wireless radio channels as the last hop connection. Cellular Networks, networks in which a mobile host is connected to the fixed network with the help of Base Stations, is the most common form of wireless networks. Even that this type of networks contains only one wireless link, the performance of TCP is highly affected. A key factor for that unsatisfactory performance is the wireless radio channel quality that can fluctuate greatly in time due to channel fading and user mobility, leading to a high variability of transmission time and delay as well as to data packet losses. Furthermore, the possibility to handoff from one base station to another may lead to sudden and high increase of data packets delay over the connection and a burst data packet loss episode.

In addition to data packet losses due to the wireless channel inefficiencies, high fluctuations of RTT values over the connection may lead the TCP Retransmission Time-Out to expire. This will invoke the TCP sender to trigger its congestion control algorithm. This leads to TCP data packets retransmission; although that the retransmitted packet is not actually lost but simply delayed. As a result, TCP decreases its CWND to minimum and under-utilizes the available bandwidth unnecessarily. In addition, TCP sender consumes more energy to retransmit the considered to be lost data packets without need (redundant retransmission).

Within wireless mobile networks, losses are often due to wireless link channel errors such as interference, variable RTT delays, or nodes' mobility. All the above reasons lead to TCP performance degradation in such environments as it was mainly developed to deal with congestion over wired networks and not other data packet loss types (such as those due to wireless radio channels). In order to improve the performance of TCP within such networks, many proposals were introduced. These solution proposals can be categorized as follows:

- Split-connection approaches,
- Link layer related approaches,
- End-to-end approaches.

The split-TCP [13] approach divides the end-to-end TCP connection between the mobile node and the corresponding node into two separate, independent connections with a proxy serving as a common point between the two connections. This proxy is usually located at the base station. The main idea in this approach is to isolate impacts of wireless link errors and variable RTT delays over it from the impact of the wired connection (mainly congestion in this second). Consequently, TCP congestion control, timeout and retransmission mechanisms in the wired link will not suffer from the fluctuating quality of the wireless radio channel. Even that this approach can optimize the wireless link performance, this approach introduces extra TCP protocol overhead over the connection (extra buffer required for each connection, and higher overall delay), which in turn causes performance degradation. This also, complicates the handoff process over the network. Also, the main idea of the split-TCP violates the end-to-end semantics of TCP.

Many other approaches, such as Snoop, [14] [15] was developed for networks with high wireless channel errors. In this approach, all the modifications are done at the base station side. The main idea of Snoop is very similar to that of split-TCP, snoop tends to hide and isolate the wireless channel errors and looses from the wired network, and avoids the frequent fast retransmit triggering at the other end of the TCP connection (the wired network). This action leads to faster recovery due to the local data packets retransmissions, and hence to better throughput performance than in the split-TCP solution. However, the problems of TCP performance degradation in case of nodes' mobility are not treated or considered in this solution.

The third approach proposed in order to enhance the performance of TCP within wireless mobile networks applies explicit loss notification (ELN) [15] messages to distinguish between congestion-induced and non-congestion-induced losses. However, the transmission of such messages leads to more computational complexity at the TCP node as it introduces more protocol overheads over the TCP connection.

The Wireless Transmission Control Protocol (WTCP) [16] is an example of the end-to-end approaches. WTCP is an end-to-end TCP modification to improve its performance within wireless mobile network. In this variant of TCP, the detection of losses due to congestion is done at the receiver side through the measured inter-packet separation delay. Hence, most of the computations and calculations for the congestion control are done at the receiver side. Even that, WTCP shows the ability to detect and handle both loss types, the fact that most of the calculations are done at the receiver side increases the complexity of the approach and requires protocol modification at both sides (sender and receiver).

All the above discussed solutions may give help to enhance the performance of TCP within wireless mobile networks where only the last communication hop is a wireless channel. However, if all the communication channels are wireless such as within wireless ad hoc networks, we expect that the performance of TCP would be dramatically influenced. In the following section, we discuss the main issues that may influence TCP performance within wireless ad hoc network environments. We also discuss some of the proposed solutions in order to improve its performance within such networks.

2.4 TCP WITHIN WIRELESS AD HOC NETWORKS

Wireless ad hoc networks are infrastructure-less networks that do not require pre-existing network configuration or a centralized administration. Wireless ad hoc devices are totally connected though wireless radio channels, which are considered as scarce resources. In such networks, nodes are independent and self-organized; each device in wireless ad hoc network can take the role of an end system, a server, a router, a gateway, or all of them at the same time. This behavior as well as the implied characteristics may lead to diverse performance issues as explained in the following.

2.4.1 TCP PERFORMANCE ISSUES WITHIN WIRELESS AD HOC NETWORKS

Wireless ad hoc networks inherit some of the wireless networks problems, more precisely, the wireless channel related problems. In addition, ad hoc networks suffer from other problems related to their specific characteristics, such as network partitions, route failures that would result from nodes mobility, node battery depletion, and multi-hop communications. Some other issues may also influence TCP performance. In order to improve TCP performance over wireless ad hoc networks, it must be able to distinguish and to recover from the new data packet loss types those arise within such networks. The new challenges that TCP would confront within such networks are: wireless lossy channels, multi-path routing, network partitions, network topology and the surrounding environment, link failures, and power constraints. We will discuss these issues in the following.

2.4.1.1 WIRELESS LOSSY CHANNEL

Wireless channel errors (bit errors) cause packets to get corrupted and result in TCP data packets (segments or acknowledgements) losses. When acknowledgements do not arrive at the TCP sender within a certain amount of time (the Retransmission Time-Out or RTO), the TCP sender retransmits the segment, exponentially backs off its retransmission timer for the next retransmission, reduces its congestion control window threshold (SSThreshold), and closes its congestion window (CWND) to one segment. Frequent wireless channel errors lead to having continuously small congestion window at the sender side resulting in low connection throughput [17]. The appropriate behavior in such situation is to simply retransmit lost packets without shrinking the congestion window to avoid the delay and the energy consumed and in turn to increase the congestion window size. It is important here to mention that, in the case of CSMA/CA (802.11) based networks, there will be an error correction algorithm; meaning that not all the packets will be lost. In fact, some of them will be corrected and resent again correctly while other will not reach the destination. Furthermore, the CSMA/CA correction action introduces some delay in the network and in the mean time will waste some of the bandwidth resource. Obviously, the delay introduced will increase the RTT value resulting in poor TCP achieved throughput and also a certain energy waste (as ad hoc nodes are battery operated). This situation in ad hoc networks happens on each wireless link which increases the probability that a data packet does not reach the destination.

2.4.1.2 MULTI-PATH ROUTING

Some routing protocols implement multi-path routing approaches and maintain multiple routes between source and destination pairs, in order to minimize the frequency of route re-computation or route discovery process. However, this may result in a significant number of out-of sequence data packets arriving at the TCP receiver side. The TCP receiver generates duplicate acknowledgments that force the sender upon the reception of three duplicate acknowledgments those acknowledge the same packet, to invoke its congestion control algorithm [17]. This action leads to unnecessary retransmissions by TCP sender, resulting in bandwidth resource waste and high energy consumption.

2.4.1.3 NETWORK PARTITIONS

Wireless ad hoc networks may periodically get partitioned for several seconds at a time. If the TCP sender and receiver find themselves in different partitions, all the data packets will be dropped. Hence, TCP sender invokes its congestion control algorithm. If the network remains partitioned for an important amount of time relatively to the Retransmission Time-Out, the situation gets even worse because of phenomena called serial timeouts. A serial timeout is a condition wherein multiple consecutive retransmissions of the same segment are transmitted to the TCP receiver while it is disconnected from the sender side. All these retransmissions are thus lost. Since the retransmission timer at the sender side is doubled with each unsuccessful retransmission attempt (until it reaches 64 sec), several consecutive failures can lead to inactivity that might last for one or two minutes even when the sender and receiver get reconnected [17]. However, the most adequate solution here is to stop the data transmission (to avoid flooding the network with packets that cannot be delivered) until that the TCP sender get reconnected to the receiver.

2.4.1.4 TOPOLOGY AND ENVIRONMENT

Where the nodes are located as well as the nature of their surrounding environment determine which nodes can contact each other and the amount of interference from other nodes [18]. If the nodes are located close to each other, there will be a greater chance that the data will not have to make as many hops as in a network where the nodes are further apart. It is clear that the energy consumption of a mobile node increases with the distance between the two communicating nodes. For that, it is better that the nodes communicate through multi-hop network. Also, networks with a dense concentration of nodes will experience more contention for the available capacity and hence more collisions and interference leading to high TCP data packet losses and thus frequent TCP sender congestion control algorithm triggering. In addition, the environment affects TCP performance in a similar way. Actually, walls and other objects that hinder radio transmissions will lower the effect of high node density.

2.4.1.5 LINK FAILURES

In case of nodes' mobility, each node might move out of the communication range of old neighbors or into the communication range of new ones. This leads to break the established routes (link failures) and also to trigger the establishment of new ones within the network. The implemented ad hoc routing protocol is the one in charge of recovering from the link failure allowing maintaining the communication session between the involved end points. Usually, a broken route results in performance degradation, since no data can be exchanged during the time where the new route is not yet available. To overcome this problem the network layer should find a new route as quickly as possible to resume the dropped communication. In fact, high mobility is not always a bad thing for ad hoc networks. Some authors have observed that mobility can increase performance by distributing traffic more evenly over the network [19]. The problem with TCP in the case of link failure situation is that after resuming the data communication session, TCP sender starts from the Slow-Start phase, with minimum CWND threshold over the links. Indeed, during a link failure multiple data packets can be lost in a bursty fashion. TCP sender shrinks its CWND to minimum considering that the loss is due to congestion. However, in case of link failure, the new discovered route might have higher link capacity compared to the old lost one. Thus, TCP sender will waste the available bandwidth (which is a scarce network resource) over the connection.

2.4.1.6 POWER CONSTRAINS

Within wireless ad hoc networks, the devices are independent and battery operated. Thus, in order to ensure the network connectivity, it is obvious to increase the network nodes lifetime as long as possible. Increasing the nodes' battery lifetime can be done through minimizing the node's energy consumption. In addition, losing a node due to battery depletion leads to broken communication sessions (link failure) even if the node is not the sender or the receiver side of that session. This is because each node within the network forwards the data packets of its neighbors when it participates to multi-hop path. Thus, we get the same effect on TCP performance as in link failure case.

2.4.2 TCP PROPOSED ENHANCEMENTS FOR WIRELESS AD HOC NETWORK ENVIRONMENT

All the above discussed factors affect the performance of TCP within wireless ad hoc networks in different ways, but they are all having a great influence on both the TCP connection throughput and may also have an influence on the node's energy consumption. Let us now discuss the main TCP enhancement approaches to improve its performance within wireless ad hoc networks.

There are many proposals introduced in order to improve the performance of TCP within wireless ad hoc networks. Some proposals dealt directly with enhancing TCP, others dealt with other OSI layers such as network layer and link layer. Also, some other proposed solutions that improved the performance of TCP through cross layer interactions. We discuss in the following the main proposed solutions to improve the performance of TCP within wireless ad hoc networks while dividing them into two categories:

1. TCP enhancements at the transport layer proposals.
2. TCP enhancements through cross layer proposals.

There are also, many other mechanisms that opt to enhance the performance of TCP within such environments and to help distinguish between different data packet loss causes over the connection at other OSI layers, such as network layer proposals. These proposals tend to deal with link failures and link layer proposals to deal with wireless channel related problems over the connection. However, these proposals are out of the scope of this thesis as we tend to enhance the performance of TCP without modifying the lower layers.

2.4.2.1 TRANSPORT LAYER SOLUTIONS

Many proposals that only deal with the transport layer were introduced in order to enhance TCP performance within wireless ad hoc networks. Fixed-RTO and TCP DOOR are examples of those proposals. We discuss here the main idea of both mechanisms and their issues within such networks.

Fixed RTO [20] is a sender side based mechanism that does not imply feedback messages over the connection. This mechanism is able to distinguish between link failure induced and congestion induced data packet losses. When two consecutives retransmission timeout expiration occurs, the TCP sender interprets the loss as a link failure induced loss. Hence, it retransmits the unacknowledged data packet but keeps the RTO value un-changed. Remind having the same situation in the case of standard TCP implementation leads to using "exponential" back-off algorithm. In Fixed-RTO, the RTO values remains fixed until the failed link is recovered and the retransmitted data packet is acknowledged. The assumption that any two consecutive retransmission timeouts are the exclusive results of link failures requires more investigations, as it may be also the case with persistent network congestion. In addition, this solution takes into consideration only two of the wireless ad hoc network data packet loss cases, and ignores the wireless channel related losses which are also common within such environments.

TCP Detection of Out-of-Order and Response (TCP DOOR) [21] is an end-to-end TCP enhancement solution that does not imply any intermediate nodes cooperation. TCP DOOR deals mainly with out-of-order data packets over the connection. The reception of out-of-order data packets is interpreted as an indication of link failure over the connection. In order to detect the out-of-order data packets, another mechanism is needed, either at the TCP sender or receiver side. The TCP sender side based mechanism profits from the non-decreasing property of the acknowledgements' sequence number to detect the out-of-order data packets. In case of duplicate acknowledgements arriving at the sender side, they all have the same sequence number. However, this is not enough for the TCP sender to detect the out-of-order data packets. Hence, the TCP sender uses a one byte option that is added to acknowledgement packets and called Acknowledgement Duplication Sequence Number (ADSN). The ADSN option is incremented and transmitted with each duplicate ACK. On the other side, when implementing a receiver side based mechanism to detect the out-of-order data packets; the TCP receiver requires an additional two-bytes TCP option that is called TCP Packet Sequence Number (TPSN).

The TPSN option is incremented and transmitted with each transmitted TCP data packet (including retransmitted packets). When the TCP receiver detects out-of-order data packets, it uses a specific option bit within the acknowledgement packet header to notify the TCP sender. At the reception of this notification at the TCP sender side, it disables its congestion control algorithm, and enters into congestion avoidance phase. The mechanisms implemented within TCP-DOOR take into consideration only the out-of-order data packets, and ignore other data packet loss causes over the wireless ad hoc network. Also the modifications implemented with the TCP data packet header may cause some incompatibility problems with other TCP existing variants in case the sender or receiver does not support TCP-DOOR.

2.4.2.2 CROSS LAYER SOLUTIONS

Cross layer proposed solutions are mainly based on the explicit exchange of notification messages between the TCP sender and receiver. Here, we introduce some of these proposed solutions.

TCP Feedback (TCP-F) [22] is a feedback based approach that deals with route failures within wireless ad hoc networks. This approach helps the TCP sender to distinguish losses due to route failures from those due to network congestion. When the TCP node's routing agent detects the disruption of a route over the connection, it explicitly sends a Route Failure Notification (RFN) packet to the TCP sender. At the reception of the RFN packet, the TCP sender enters into a snooze state: stops data packet transmission, and freezes all its connection variables (e.g. timers and congestion window size). The TCP sender stays in this snooze state until that the routing agent recovers from the link failure. In this case, the routing agent sends a Route Re-establishment Notification (RRN) packet to the TCP sender side. When the TCP sender receives the RRN packet, it leaves the snooze state and resumes data packet transmission using the previous sender window and timeout values. In order to avoid that the TCP sender stay blocked in the snooze state, it triggers a route failure timer at the reception of a RFN packet. This way, when this timer expires the TCP sender leaves the snooze state and triggers its congestion control algorithm. It is clear that this approach can only be used in the specific case of ad hoc network using a reactive routing approach.

Another TCP feedback based approach is the Explicit Link Failure Notification technique (ELFN) [4]. This one had been designed in order to distinguish the cause of data packet loss and to avoid a wrong interpretation of TCP as if it is due to network congestion. To do so, ELFN uses a real interaction between TCP and the ad hoc routing protocol. This is done through informing the TCP sender of the route failure by messages like "host unreachable" Internet Control Message Protocol (ICMP) message. Then, the TCP sender disables its retransmission timers and enters into a "standby" mode. During the standby period, the TCP sender probes the network to check if the route is recovered or not. When the TCP sender receives the acknowledgment of the probe packet, it leaves the standby mode, resumes its retransmission timers, and continues its communication session. Feedback messages consume nodes' as well as network resources. As for node's resources, sending feedback leads to supplemental energy consumption. Using feedback messages increases the TCP protocol messages overhead over the connection and consumes part of the available bandwidth. In addition, some researches [23] [24] show that ELFN performs badly within wireless ad hoc networks (worse than the standard TCP implementation) especially in case of high network loads (for example in case of 25 TCP connection over the network).

Ad hoc Transport Control Protocol (ATCP) [17] also, employs network layer feedback messages. ATCP tries to deal with both wireless channel related problems and link failure problems. The TCP sender can switch between four different connection states according to the cause of the data packet losses: normal state, persist state, congestion control state or retransmit state. In this proposal, a layer called ATCP is inserted between the TCP and IP layers of the TCP sender side. At the beginning of the connection, ATCP is always in the normal state. ATCP listens to the network state information provided by the ECN messages [25] and ICMP "Destination Unreachable" message; then ATCP chooses the appropriate connection state to be in. At the reception of a "Destination Unreachable" message, TCP sender perceives that there is a link failure over the connection and enters into the persist state. During this state, the TCP sender stops data packets transmission until finding a new route towards the destination (this is done by probing the network). ECN is used in order to notify the TCP sender about network congestion situation over the TCP connection. When the TCP sender receives the ECN packet, it triggers its congestion control algorithm without waiting for the retransmission timeout timer to expire. This is called the congestion control state. Wireless channel induced data packet losses are detected through the reception of three duplicate acknowledgements. ATCP then switches to the persist state and quickly retransmits the lost packet from TCP's buffer (this is known as the retransmit state). After receiving the next ACK, ATCP leaves the persist state to the normal state. Even that ATCP show better performance than standard TCP implementation, the feedback messages used within ATCP have the same drawbacks as in the above mentioned ELFN mechanism, especially when implemented within high load wireless ad hoc networks. This leads to high TCP protocol messages overhead and more energy consumption at the TCP nodes.

2.4.2.3 DISCUSSION

From the above discussed TCP enhancements, we notice that each proposed solution opts to overcome one specific data packet loss problem over TCP connections (a single point of view). Some were using feedback messages and others not. Results show that, the solutions that implement feedback messages over the connection enhance TCP performances better than the others. However, using feedback messages over the connection is a network resources consuming algorithm, especially the TCP nodes' energetic resources (nodes' batteries).

Hence, we argue in this thesis the need of a new TCP variant for wireless ad hoc network environment that adapts the multi-point of view nature of packet losses within such networks. This variant should optimize the performance of TCP within such networks, in both terms of throughput and energy consumption, taking into consideration all the common data packet loss causes within such environment. In addition, in order to avoid resource (bandwidth and energy) waste, we aim at proposing a TCP variant that does not use any additional feedback in the network.

2.5 SUMMARY OF TCP EVOLUTION AND NEXT STEPS

TCP was developed in order to deal with data packet losses within wired networks. We found that, within wired networks, data packet losses are mainly due to congestion. From which comes the name of its main algorithm: Congestion Control Algorithm. For many decades, TCP proved to be highly reliable for data exchange over wired links. Its reliability through the use of acknowledgments insures data integrity and success of delivery at the receiver side. This gives TCP its popularity and helps developing applications that are compatible with it. As a new kind of data networks are introduced, TCP remains the most popular transmission protocol to be used for data exchange over their links.

Many researches [26] [4] [3] had been conducted to show the performance of TCP within wireless and ad hoc networks. All of them had proved that the performances of TCP degrade highly within such networks. This is due to the fact that TCP was not designed to deal with the new causes of data packet losses emerged within these new networks. The main problem here is that TCP always reacts to data packet loss, as if it was due to network congestion. As we mentioned above, within wireless and ad hoc networks, we have more than one possible cause for data packet losses. Dealing with all models of data packet loss as if it was due to congestion, leads to TCP performance degradation. We will see later that each cause of data packet loss requires a particular reaction of TCP in order optimize TCP performance.

Many TCP variants had been developed, in order to enhance its performance within both wired and wireless infrastructure networks. We have seen also, that the challenges that TCP encounter within wireless ad hoc networks are more complicated than those found within wireless infrastructure networks. This is due to the fact that wireless ad hoc networks are totally based on links that are wireless. Thus, the problems that can be found over the last hop of wireless infrastructure networks are multiplied over the many wireless channels of the ad hoc network.

According to the above discussion, we note that in order to make TCP capable to deal with any type of data packet loss over the connection (especially wireless links), it must distinguish between different types of data packet loss causes. Congestion-induced and non-congestion induced or wireless related losses. Many researches dealt with this problem and many proposals where done in order to define this classification concept. But all these researches were conducted for wireless infrastructure networks; where only the last hop is the wireless link.

Within wireless ad hoc networks, we have a new cause for data packet losses that might be due to nodes' mobility or energetic resources depletion. As stated before, in ad hoc networks, each node relays data packets for the other nodes. Thus, when an intermediate node moves out of the communication range of other nodes, the communication session between the communicating end points may be interrupted or closed. Also, the same scenario may be repeated if the intermediate node runs out of battery. We note here the difference between wireless infrastructure networks and wireless infrastructure less networks (ad hoc networks) from the point of view of nodes' availability. For wireless infrastructure networks, the host or the node is one of the communicating end points and does not take part in the communication of other nodes or sessions within the network. This means that if the node disappears, only that communication session will be affected and not the part of the network that is concerned by other communication sessions, which is the case of ad hoc networks. From where, comes the importance of keeping the ad hoc networks' nodes on line as long as possible in order to maintain the network connectivity and keep the communication sessions. Also, that results in a new type of data packet losses that is called link-failure-induced loss. Link failure-induced loss can be either the result of nodes mobility or energetic resources depletion.

Thus, in order to get the best adapted TCP variant for wireless ad hoc network that optimize both network and nodes resources, we have to take into consideration two points:

1. Maximizing the utilization of wireless available bandwidth.
2. Minimizing the overall energy consumption of the networks nodes.

2.6 CONCLUSION

From the evolution history of TCP and the advances that accompanied it, we can say that TCP may be suitable for any kind of networks as soon as it is adapted to deal with its specifications. Concerning wireless ad hoc networks, we also conclude that TCP still needs to be modified in order to deal with different data packet losses that can be found within such networks. The main key in this modification will be the ability to distinguish between the different data packet loss causes over the connection and the ability to take the most appropriate action to recover from that loss taking into consideration the optimization of both network and node resources.

Proposing a TCP variant that allows achieving this goal is the final aim of this thesis. All the contributions that will follow are in order to reach this final aim.

CHAPTER 3. SEDLANE: SIMPLE EMULATION OF DELAYS AND LOSSES FOR AD HOC NETWORK ENVIRONMENT

3.1 INTRODUCTION

Wireless ad hoc network consists of independent nodes that communicate with each other through wireless radio channels with no need for a fixed network infrastructure or a centralized administration. Due to its specific nature, many researches had been performed in order to study wireless ad hoc networks performance. Consequently, new network protocols and applications were introduced. However, it is important to evaluate these new protocols and applications within wireless ad hoc network environment. Generally, the following three approaches can be used to carry out this evaluation:

• Building a real test-bed for wireless ad hoc networks with the desired network conditions, protocols and applications. Although that this approach is very realistic, it is expensive to setup. Moreover, the fact that the experiments are not reproducible does not make it always the best choice. Actually, no researches were done in a scale beyond a dozen of nodes.

• Using a network simulator, such as Network Simulator version 2 (NS-2) [27]. Simulating a network is always easier than constructing a real test-bed. However, the simulator traffic is generated by traffic models that sometimes do not match real applications behavior. Also, some performance parameters cannot be evaluated through simulations (e.g. node related performance parameters such as CPU usage and computational energy consumption).

• Emulation is a compromise between the two approaches above. Emulators are a scalable, cost-effective solution that can offer controlled and reproducible network configurations for test and evaluation.

Emulation is considered both as lower cost and a more realistic approach to test and evaluate network applications and protocols. Indeed, emulating wireless ad hoc networks and taking into consideration their specific characteristics such as nodes' mobility nodes and energy constrains could be of high interest. Therefore, one of our major goals is to build such emulator that allows for an easy evaluation of end-to-end applications and protocols in wireless ad hoc networks.

The rest of this chapter is organized as follows: we start by discussing the motivation behind our work, and then we review some of the existing ad hoc network emulators and discuss their advantages and disadvantages. After, we provide a brief background of Dummynet and Network Simulator (NS-2) tools. Then, we present SEDLANE, our emulation tool, and discuss its main algorithms and operation modes. After discussing all the details of SEDLANE, we validate its functionality using a test-bed and show and analyze the obtained results. Finally, we conclude our work in this chapter and give some ideas for future work.

3.2 MOTIVATION AND PROBLEM STATEMENT

TCP throughput can be calculated using both RTT delay and data packet loss rate over the connection. Floyd et al. [28] and Ott et al. [29] propose a model to estimate the throughput of a TCP connection under known delay and loss conditions as follows (equation 3.1):

$$r_{TCP} = \frac{1.22 * M}{\tau * \sqrt{l}} \qquad (3.1)$$

Where: r_{TCP} is the TCP connection throughput, M is the maximum packet length, τ is the round trip time (RTT) of the connection and *l* is the average loss measured during the lifetime of the connection. From such a model, we can conclude that TCP performance depends mainly on RTT values and loss rates over the connection. Hence, in order to evaluate the performance of TCP variants or applications and protocols running over them, we can use an emulator that manipulates RTT and loss values according to realistic network performance scenarios.

Our goal is to propose a simple emulator that helps in evaluating new wireless ad hoc network applications and protocols running over TCP. From this perspective, we characterize a useful wireless ad hoc network emulator by the following features:

- Simple and easy to implement,

- Does not require any specific or expensive networking hardware,

- The experiments must be controlled and repeatable,

- The ability to use non-complex (easy to extract) characteristics of the connection (such as loss and delay values) to emulate multiple wireless ad hoc network configurations and parameters (such as ad hoc routing protocols, nodes' mobility, and connection throughput),

- Emulating multi-hop wireless ad hoc network of any size using a small number of physical machines.

In this chapter, we propose a new wireless multi-hop ad hoc network emulator that is able to emulate wireless ad hoc network environments of any scale using only small number of physical machines (even only one) and two of the connection's characteristics (RTT values and Losses). We developed a user friendly version of SEDLANE in order to facilitate its usage (Appendix ONE: SEDLANE Available V). If we take into consideration that most network experimentations begin with simulations, it would be useful to extend the simulation results beyond the scope of the simulator's capabilities by using an emulator that gives the same network circumstances (through the network simulator trace files). Since it is easier to control the network environment using a simulation tool, we profit from this advantage by generating the desired scenarios within wireless ad hoc network environment using NS-2 as a network simulation tool. Then we inject the trace files into Dummynet as a traffic shaping tool to apply the same effects on a real traffic scenario, as if we construct a real multi-hop wireless ad hoc network environment. In this way, we obtain a Simple Emulation of Delays and Losses for Ad Hoc Networks Environment (SEDLANE). SEDLANE allows for emulating a multi-hop wireless ad hoc network of any scale in a virtual way and with any mobility scenario without the need of physically moving the ad hoc nodes. SEDLANE does not need any additional knowledge of new emulating tools. It uses the well known and freely available network simulation (NS-2) [27] and traffic shaping (Dummynet) [30] tools. Additionally, it is an inexpensive tool and does not need a special hardware setup. This tool enables us to test the performance of network protocols and applications (transport layer and above) at the end points of any wireless ad hoc network (emulated by SEDLANE). Our aim is to test the end-to-end nodes' performance that cannot be obtained through simulations, such as hardware-related parameters (i.e. computational energy cost at end points). SEDLANE allows for evaluating the tested applications when running in real situations. The entire wireless ad hoc network environment is emulated using only one machine (e.g. a particular machine installed in the middle of the two communicating end points; or even one of the communication end-points that can also play this role). SEDLANE emulates network nodes' mobility, ad hoc routing protocol, and TCP connection throughput through RTT values (delays) and data packet losses within the network.

The objective of this chapter is to demonstrate that such emulation tool is capable of emulating effectively the end-to-end delays and data packet losses of a wireless ad hoc network.

3.3 RELATED WORK

In this section, we review the most known wireless and ad hoc networks emulators, in order to find the one that meets our needs. Portable real-time Emulator (PoEm) [31] is a TCP/IP-based real-time MANET software emulator that is developed for testing and evaluating MANET routing protocols. PoEm works in a client/server structure model and can be run over any hardware platforms providing that they are connected through TCP/IP connections. Each client is represented as a Virtual MANET Node at the server. Several clients can be connected to the server to build an emulated network. PoEm includes multi-radio environment emulation. In order to imitate a real-time MANET environment, PoEm's emulation server receives packets from emulation clients and then relays them to the corresponding clients according to the emulated network topology configuration. PoEm is one of the few emulators that support multi-radio MANETs environment by using the channel-ID indexed neighbor tables. Nevertheless, the structure of the emulator (as server/client architecture) requires many physical machines to be used in order to build the emulated architecture. Thus, the higher the number of nodes to be emulated the more complex and expensive the structure becomes.

User Mode Linux (UML) [32] is a system that can be used for emulating wireless networks. It uses several independent Linux instances each of them running an adapted version of a complete Linux kernel in user mode. It provides a shared network environment through the base Linux system. In [33] the authors state that, UML emulation performance is severely reduced when used to create wireless ad hoc network emulations. This is due to the fact that UML runs in user mode and privileged operating system functions must be emulated by the underlying kernel running on the real hardware. This means that context switching within the virtual UML take up to 100 times longer than on a standard Linux system. In addition, UML offers only a virtual Ethernet interface and not a virtual WLAN wireless interface (i.e. 802.11b interface). UML itself emulates only a single wireless node. To emulate an entire network, a central controlling instance is required to monitor and control all UML instances. Mobility Emulator (MobiEmu) [34] is one of such central control solutions that emulate a basic wireless network based on UML. MobiEmu have a separate physical machine for each emulated node. These limitations made UML and MobiEmu non-scalable.

ModelNet [35] was initially developed for testing large-scale distributed services for wired wide-area network environments. ModelNet architecture is composed of Edge Nodes and Core Nodes. Edge Nodes in ModelNet can run arbitrary architectures and operating systems. They run native IP stacks and function in the way they would in real environments with the exception that they are configured to route IP traffic through ModelNet cores. Whereas, Core Nodes run a modified version of FreeBSD in order to emulate topology-specific hop-by-hop network characteristics. To decrease the number of edge machines required for large-scale evaluations, ModelNet architecture employs Virtual Edge Nodes (VNs). VNs enable the multiplexing of multiple application instances on a single edge machine, with each instance getting its own unique IP address. ModelNet edge machines use internal IP addresses (10.*), thus the number of interfaces that can be multiplexed onto an edge node is not limited by IP address space limitations, but rather by the amount of computational resources (e.g. threads, memory) that a target application uses. ModelNet configures all VNs to route their traffic through a particular ModelNet core.

MobiNet [36] is another emulator that has a similar architecture to that of ModelNet. MobiNet's architecture is composed of edge nodes and core nodes. The edge nodes support a variety of platforms and operating systems. As in ModelNet, MobiNet edge nodes host multiple virtual nodes (VNs) to allow for large-scale emulations. MobiNet core nodes emulate wireless network behavior at multiple layers while eventually routing packets to the edge node hosting the destination VN. MobiNet incorporates mobile wireless ad hoc network characteristics (i.e. nodes' mobility behavior, routing module, and MAC layer collisions). Although the number of physical devices required in ModelNet and MobiNet is reduced, and the platform seems to be well developed for mobile wireless ad hoc environments emulation, its setup remains complex.

Mobile Network Emulator (MNE) [37] uses a static network infrastructure to interconnect devices. Each device has two interfaces: one act as a mobile emulation control channel while the other is used for the emulated wireless network. The latter can be a wireless interface, allowing for some lower layer effects (such as collisions) to be taken into account as well. Information about topology changes is sent through the control channel, causing the nodes to set or remove IPtables-rules accordingly; similar to the way it is done in MobiEmu. The main problem of this approach is that it still needs a separate device for each emulated wireless host and thus does not allow for emulating large ad hoc networks.

Emulation of Mobile WIreless Networks (EMWIN) [38] improves the issue with the number of physical machines by allowing each node to have several network interfaces, each acting as a separate wireless node. EMWIN provides emulation of some MAC layer effects by introducing an additional emulated MAC (eMAC) layer. Even if the number of the required physical machines is reduced, it remains high. So, this approach is still impractical. Although this tool offers some interesting features, its scalability remains limited.

Network Emulator for Mobile Ad-Hoc Networks (NEMAN) [39] can emulate a relatively large scale, up to hundreds of nodes, wireless network using a single physical machine. Using ns-2 scenario descriptions; NEMAN is able to control the physical connectivity between virtual mobile nodes. NEMAN uses Open Link State Routing Protocol (OLSR) [40] in order to establish and maintain the IP layer of the emulated network. NEMAN can be used to develop new ad hoc networks protocols and applications (such as SKiMPy [41]). In addition, the developed code can be easily deployed on wireless devices. Furthermore, NEMAN's architecture components can be run in the user space of the Linux kernel; root-privilege is only needed when configuring the virtual network interfaces. For its performance, NEMAN suffers from a degradation of performance with high number of neighbors that are directly connected and be turned on at the same time. Also, the GUI's performance drastically degrades as the total number of emulated nodes and active links increase.

Wireless Ad-Hoc Network Emulation Using Microkernel-Based Virtual Linux System [42] uses multiple Linux instances running on top of a modified microkernel. A special multiplexer component is implemented in order to interconnect between the Linux instances and the physical networks. Controlling the network connections between different instances helps emulating node movement and error conditions within the network. The presented system for mobile wireless ad hoc network can emulate multiple nodes using a single physical node, and consume significantly less resources than other approaches based on UML. Using the Fiasco[1] [43] microkernel system, which is a descendant of the L4 microkernel development [42], improves the system's performance. Using this system helps emulating mobile wireless ad hoc networks by utilizing a set of Linux instances on top of a L4 microkernel, and allows wireless

[1] Fiasco is a second-generation microkernel that concentrates on only implementing the most basic operating system functionality, leaving advanced aspects to server modules running in a non-privileged system mode.

network connection conditions modeling as well as error injection with varying levels of details. However, the proposed system provides only coarse grain emulations.

Scenario Driven Network Emulation (SDNE) [44] was developed to help developing and testing the Conference Controller Architecture (CCA[2]) without real users connecting through real networks. It emulates both network and users that are engaged in a multimedia conference. SDNE configuration is based on a scenario organization module that is composed of: (i) a map file that describes the physical area under emulation, and (ii) configuration script that describes the emulation scenario. SDNE can emulate different mobility patterns for the wireless nodes, and their effects on the connection behavior. The map files that are used in the system are difficult to produce; trial and error process is needed to find the most suitable map. Also, traffic shaping is performed only on the outgoing packets, which adds to system complexity in order to perform traffic shaping over incoming packets. Other emulation tools also exist in the literature [45] [46]**.**

Table 3.1 summarizes the main characteristics of most common wireless and ad hoc network emulators discussed in the Related Work section.

	PoEm	UML	MobiEmu	MobiNet	MNE	EMWIN	NEMAN	Micro kernel-based	SDNE
Low costs		x	x	x	x		x		
Scalability	x					x	x		
Portability	x	x	x	x	x	x	x		
Routing	x		x	x	x		x		x
Simplicity						x	x		
Mobility	x	x	x		x	x		x	x
User - GUI	x		x	x	x		x		

TABLE 3.1. DIFFERENT AD HOC NETWORKS EMULATORS CHARACTERISTICS

From the table above, we can conclude that none of the presented emulator tools answers our needs. Furthermore, none of the emulation tools presented above emulates an ad hoc network taking the TCP connection characteristics into consideration.

3.4 BACKGROUND

In this section, we describe briefly, the main tools used to design and implement our emulator SEDLANE, both Dummynet [30] and NS-2 [27].

[2] CCA is a framework that is designed to support interactive multimedia conference users connecting from a diverse range of network connection types.

3.4.1 DUMMYNET

Dummynet is a traffic shaping tool that has been originally designed for testing networking protocols [30]. Through Dummynet, we can enforce delays, packet losses, queue size, and bandwidth limitations.

Figure 3.1 illustrates the main function of Dummynet. Dummynet is entirely controlled by the system's *ipfw* (*IP Fire-Wall*) commands and a set of *sysctl* (*System Control*) variables. The *ipfw* commands help the user to define the rules to be applied, by Dummynet, on the packets crossing a particular network interface on its input or output. Unlike many other traffic shaping packages, Dummynet has a very little overhead. All processing is done within the kernel and no data copying involved in moving packets through pipes. Dummynet implements the concept of pipes, which are defined as a communication channels between the source and the destination. It is capable of handling thousands of pipes. Each packet will be manipulated according to the rules associated with each pipe (communication channel), and a packet can undergo several rules. We can use the same command to configure different network parameters, such as bandwidth, delay, queue size, and packet loss. For more details about Dummynet configurations and control, please refer to [30].

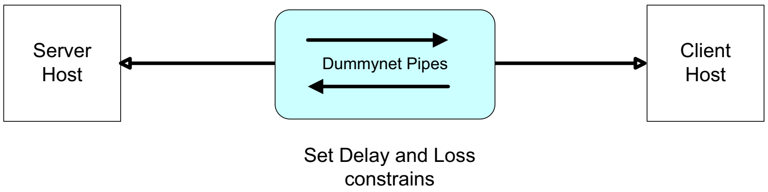


FIGURE 3.1. THE PRINCIPLE OF DUMMYNET OPERATION

Our decision to use Dummynet was based on the fact that, Dummynet helps implementing data packet losses and delay constraints efficiently within the system's kernel, and with minimum processing overhead.

3.4.2 NETWORK SIMULATOR – VERSION 2

Network Simulator version 2 (NS-2) [27] is an open source simulation tool that is used to simulate a wide range of data networks, wired and wireless data networks. NS-2 provides substantial support of simulating different TCP variants, many routing protocol, and multicast protocols over wired and wireless networks.

NS-2 is built in C++ and provides a simulation interface through OTcl, an Object-oriented Tool Command Line. The user describes a network topology by writing OTcl scripts, and then the main ns program simulates that topology with the specified parameters.

3.5 SIMPLE EMULATION OF DELAYS AND LOSSES FOR AD HOC NETWORKS ENVIRONMENT (SEDLANE)

The main idea of SEDLANE is to configure the Dummynet pipes (defining rules) using the NS-2 trace files (cf. Figure 3.2). SEDLANE uses the NS-2 TCP trace files to identify different data packet classes. This is done by gathering the packets that have similar RTT values. Then, SEDLANE dedicates one pipe, or communication channel, for each group of packets. Packet loss percentage can be either calculated; from the NS-2 TCP trace file or NS-2 standard trace file; and then applied to Dummynet pipes. Hence, according to the identified classes of packets, RTT values, and loss rates that are distributed among classes, SEDLANE dynamically generates the Dummynet rules to be applied on data packets. The different algorithms used by SEDLANE are described below.

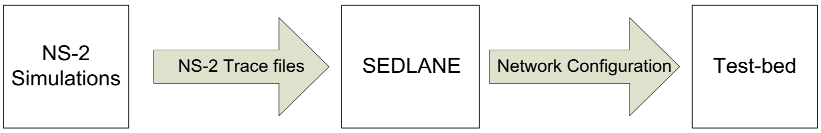


FIGURE 3.2. THE PRINCIPLE OF SEDLANE OPERATION

3.5.1 SEDLANE OPERATION MODES

We have two operation modes in which we can run SEDLANE: Simultaneous and Sequential. The choice of using either depends on the user's experimentation methodology. We will show in the evaluation study the advantages of each of these modes.

3.5.1.1 SIMULTANEOUS OPERATION MODE

Dummynet, by default, configures the communication channels (*ipfw* pipes) simultaneously, meaning that it configures all the communication channels at once (in parallel). In simultaneous operation mode, SEDLANE follows the default operation mode of Dummynet, and configures all *ipfw* rules (Dummynet communication channels) at once, assigning each pipe a different probability (calculating probability values will be explained later). SEDLANE operates in simultaneous mode by default. This mode reflects the normal operation mode of the system's *ipfw*. Figure 3.3 describes this operation mode.

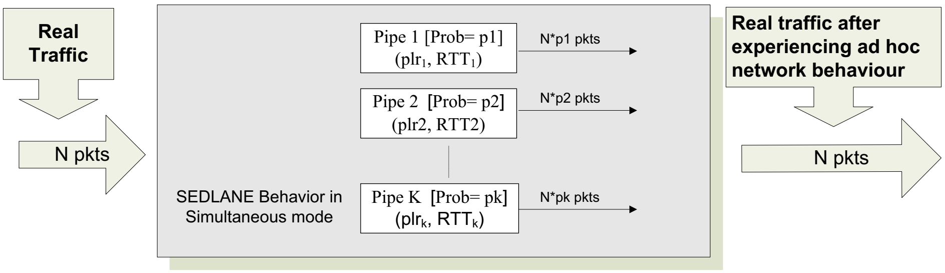

Real Traffic
N pkts
Pipe 1 [Prob= p1] (plr₁, RTT₁)
N*p1 pkts
Pipe 2 [Prob= p2] (plr2, RTT2)
N*p2 pkts
SEDLANE Behavior in Simultaneous mode
Pipe K [Prob= pk] (plrₖ, RTTₖ)
N*pk pkts
Real traffic after experiencing ad hoc network behaviour
N pkts

FIGURE 3.3. SEDLANE SIMULTANEOUS OPERATION MODE

As can be shown in the Figure, using simultaneous operation mode applies all the *ipfw* rules without time constrains. Thus, if the user needs to emulate an ad hoc network for a time greater than that of the simulation scenario, SEDLANE's simultaneous mode will be the right choice. This constitutes the main advantage of this operation mode.

3.5.1.2 SEQUENTIAL OPERATION MODE

In the sequential operation mode, as shown in Figure 3.4, SEDLANE configures only one *ipfw* rule at a time. Each rule will be flushed after a certain "lifetime" (the time during which the simulated connection remained at a particular RTT values range) before a new rule (with a new average RTT and loss values) is configured. The "lifetime" of each rule, known also as Sequential Delay, is extracted from the NS-2 TCP trace file. Each rule will have a different lifetime corresponding to that in the NS-2 TCP trace file (i.e. the time during which RTT values stays close to an average value). The algorithm used to calculate the sequential delay values from the TCP trace file will be explained later. In this operation mode, SEDLANE tries to emulate the performance values exactly as extracted from the NS-2 TCP Trace file, and respecting the time of the simulation scenario.

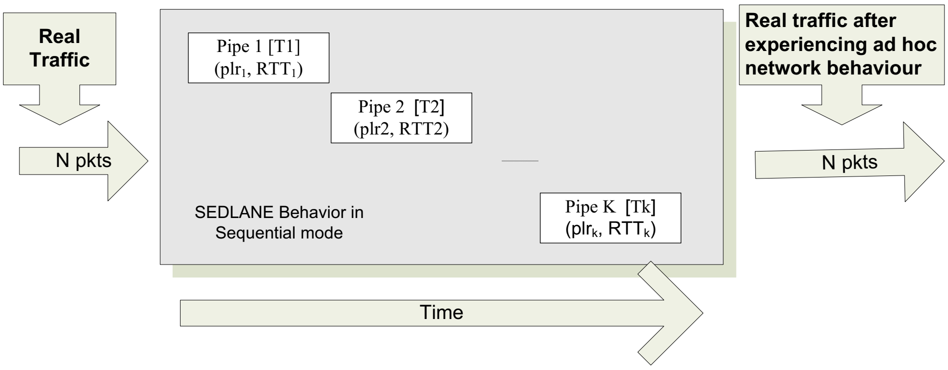

Real Traffic
N pkts
Pipe 1 [T1] (plr$_1$, RTT$_1$)
Pipe 2 [T2] (plr2, RTT2)
Pipe K [Tk] (plr$_k$, RTT$_k$)
SEDLANE Behavior in Sequential mode
Real traffic after experiencing ad hoc network behaviour
N pkts
Time

FIGURE 3.4. SEDLANE SEQUENTIAL OPERATION MODE

3.5.2 SEDLANE FUNCTIONAL ARCHITECTURE AND ALGORITHMS

SEDLANE starts by reading NS-2 trace files, provided by the user. According to the traces contained in these files and the users' provided arguments, SEDLANE decides which of its calculation algorithms will be triggered. Figure 3.5 shows the SEDLANE functional architecture and its main calculation algorithms. These algorithms will be discussed in details below.

3.5.2.1 INPUT DATA FROM TRACE FILES

SEDLANE extracts the following data from the NS-2 trace files specified by the user:

- Number and values of different RTT samples found in the file,
- Timestamp of each RTT transition,
- Total data bytes transmitted and during each RTT,
- Total data bytes retransmitted and during each RTT,
- Total connection time,
- Lifetime for each RTT,
- Total data bytes dropped and during each RTT.

All of the data extracted above represent the targeted connection parameters to be emulated. These values are used to calculate the communication channels configuration. The calculation algorithms used are discussed in details below.

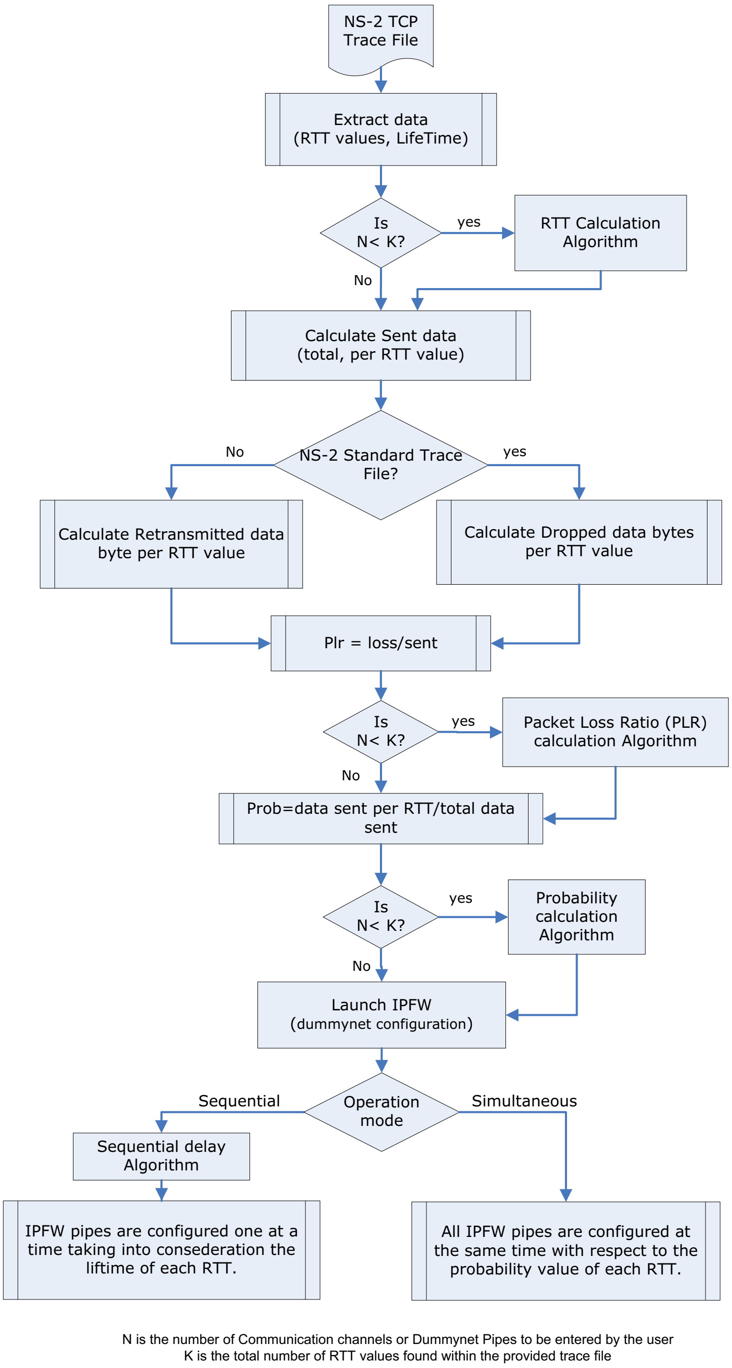

NS-2 TCP Trace File
Extract data (RTT values, LifeTime)
Is N< K?
yes
RTT Calculation Algorithm
No
Calculate Sent data (total, per RTT value)
NS-2 Standard Trace File?
No
yes
Calculate Retransmitted data byte per RTT value
Calculate Dropped data bytes per RTT value
Plr = loss/sent
Is N< K?
yes
Packet Loss Ratio (PLR) calculation Algorithm
No
Prob=data sent per RTT/total data sent
Is N< K?
yes
Probability calculation Algorithm
No
Launch IPFW (dummynet configuration)
Sequential
Operation mode
Simultaneous
Sequential delay Algorithm
IPFW pipes are configured one at a time taking into consederation the liftime of each RTT.
All IPFW pipes are configured at the same time with respect to the probability value of each RTT.
N is the number of Communication channels or Dummynet Pipes to be entered by the user
K is the total number of RTT values found within the provided trace file

FIGURE 3.5. SEDLANE OPERATION FUNCTIONS

3.5.2.2 NUMBER OF COMMUNICATION CHANNELS (N)

This argument defines the number of rules to be configured according to the desired accuracy. Note that configuring a large number of rules requires more CPU resources and might affect the performance of the test-bed. If the number of different RTT samples found in the TCP trace file (K) is equal to or less than the number of communication channels provided by the user (N), SEDLANE dedicates a Dummynet pipe for each RTT value. Otherwise, SEDLANE calculates new RTT values that correspond to the specified number of pipes (N) as given by the user. Calculating the new RTT values is done through RTT calculation algorithm, which is explained below.

3.5.2.3 RTT CALCULATION ALGORITHM

If the number of the different RTT samples found in the TCP trace file (K) is greater than maximum number of pipes defined by the user (N), SEDLANE triggers its RTT calculation algorithm (as shown in Figure 3.5). This algorithm is used to calculate new RTT samples from the original ones found in the TCP trace file. These new values are calculated as shown in equation 3.2.

$$\begin{aligned}
RTT_{new_1} &= \frac{RTT_i + RTT_{i+1}}{2} \\
RTT_{new_2} &= \frac{RTT_{i+2} + RTT_{i+3}}{2} \\
&\ldots\ldots \qquad (3.2) \\
RTT_{new_{(n-1)}} &= \frac{RTT_{(n-1)} + RTT_{(n)}}{2}
\end{aligned}$$

Where: $RTT_{new1,2,\ldots,n-1}$ are the new generated RTT samples; RTT_i and RTT_{i+1} are two original consecutive RTT samples.

SEDLANE repeats the calculations until the new number of RTT samples matches the defined number of pipes (N). It is possible that the number of resulting *ipfw* rules becomes less than the targeted number of pipes. This comes from the fact that SEDLANE does not allow configuring two consecutive pipes with equal RTT values. Therefore, in simultaneous mode, only unique RTT values are allowed. Thus, there will be one rule per RTT value. The resulting RTT values are average values.

3.5.2.4 PROBABILITY CALCULATION ALGORITHM

Probability is used by Dummynet to represent the probability of getting a match on this rule meaning that this data packet will be manipulated by this *ipfw* rule. The probability assigned to each *ipfw* rule depends on the SEDLANE's operation mode: either sequential or simultaneous. In sequential mode, each rule will have a deterministic probability; since there is only one rule configured at a time. In simultaneous mode, the probability assigned to each rule is the ratio of the transmitted data bytes during a certain RTT lifetime to the total number of data bytes transmitted over the connection, as extracted from the NS-2 TCP trace file. Equation 3.3 demonstrates how probability values are calculated for each RTT sample.

$$prob_{RTT_i} = \frac{txbytes_{RTT_i}}{txbytes_{total}} \qquad (3.3)$$

Where: $prob_{RTT_i}$ is the assigned probability for RTT_i, $txbytes_{RTT_i}$ is the data bytes transmitted during the RTT_i lifetime, and $txbytes_{total}$ is the total data bytes transmitted over the emulated connection.

In case, the number of RTT transitions in the trace file (K) is higher than the targeted number of pipes (N), the probability of each new RTT value is calculated according to equation 3.4.

$$prob_{new_1} = \left(\frac{prob_{RTT\,i} + prob_{RTT\,i+1}}{2}\right)\left(\frac{M}{M-1}\right) \text{ and so on.} \tag{3.4}$$

Where: $prob_{new_1}$ is the new generated probability; $prob_{RTTi}$ and $prob_{RTT\,i+1}$ are two original consecutive probability values; *M* is the length of original probability list (number of pipes corresponding to the number of original RTT values); *M-1* is the length of the new generated probability list. This operation will be repeated in iteration till *M* becomes equal to the targeted number of pipes *N.*

3.5.2.5 PACKET LOSS RATIO (PLR) CALCULATION ALGORITHM

In order to calculate the packet loss ratio for each *ipfw* rule, SEDLANE performs the following operations:

- Calculate the amount of data bytes transmitted during each RTT lifetime,
- Calculate the amount of data bytes dropped or retransmitted during each RTT lifetime,
- Calculate the PLR for each RTT by dividing the corresponding amount of dropped or retransmitted data bytes by the amount of sent data bytes.

These operations are detailed below.

3.5.2.5.1 CALCULATING THE AMOUNT OF TRANSMITTED DATA BYTES

SEDLANE calculates the amount of transmitted data bytes for each RTT, $txbytes_{RTT}$, as well as the total amount of data bytes transmitted, $txbytes_{total}$. If the number of RTT values retrieved from the NS-2 TCP trace file (K) is greater than the desired number of pipes (N) to be configured, the following iteration takes place and a new list of $txbytes_{RTT}$ values is generated using equation 3.5.

$$txbytes_{RTT(i)new} = \frac{txbytes_{RTT(i)} + txbytes_{RTT(i+1)}}{2} \tag{3.5}$$

The algorithm then calculates the new sum of transmitted data bytes, as in equation 3.6.

$$txbytes_{newTotal} = \sum txbytes_{RTT(i)new} \tag{3.6}$$

The algorithm maintains always the original total amount of transmitted data bytes. This is done by using equation 3.7.

$$txbytes_{RTT(i)adapted} = txbytes_{RTT(i)new} \times \frac{txbytes_{total}}{txbytes_{newTotal}} \tag{3.7}$$

Thus, after each iteration round, the total amount of transmitted data bytes will always be equal to $txbytes_{total}$. The iteration above will continue till we have a number of $txbytes_{RTT}$ values that is equal to the desired number of pipes to be configured.

3.5.2.5.2 CALCULATING THE AMOUNT OF LOST DATA BYTES

SEDLANE adopts two methods to calculate lost data bytes. By default, SEDLANE uses NS-2 TCP trace file to calculate the data packet loss ratio. It records the number of retransmitted data bytes during each RTT lifetime, and uses this value as the amount of lost data bytes. The retransmitted data bytes include data bytes retransmissions due to retransmission time out (RTO) and fast retransmit (FR). Alternatively, SEDLANE is able to calculate the data packet loss ratio from the standard NS- 2 trace file. In this case, SEDLANE counts the amount of dropped data bytes during the lifetime of each RTT as in equation 3.8. In case the number of RTT values retrieved from the NS-2 TCP trace file (K) is greater than the desired number of pipes (N) to be configured, SEDLANE executes the algorithm explained in the previous section and maintains the total amount of lost data similarly.

$$Lostbytes_{RTT(i)} = \sum_{lifetime} Dropbytes_{RTT(i)} \tag{3.8}$$

3.5.2.5.3 CALCULATING THE PACKET LOSS RATIO

After determining the amount of both transmitted and lost data bytes for each pipe, PLR is calculated simply using equation 3.9.

$$plr_{RTT(i)} = \frac{lostbytes_{RTT(i)}}{txbytes_{RTT(i)}} \tag{3.9}$$

Where: $plr_{RTT(i)}$ is the packet loss ratio of RTT_i; $lostbytes_{RTT(i)}$ and $txbytes_{RTT(i)}$ are the lost data bytes and transmitted data bytes of RTT_i respectively.

3.5.2.6 SEQUENTIAL DELAY CALCULATION ALGORITHM

Sequential delay is the lifetime of each RTT value. This argument could be calculated from NS-2 TCP trace file or provided by the user as a fixed number. If entered by the user, this value will be used for all configured *ipfw* pipes. Meaning that, all RTT values will have the same lifetime duration. If calculated from NS-2 TCP trace file, it will be calculated using equation (3.10).

$$RTTi_{lifeTime} = TS_{RTT(i+1)} - TS_{RTTi} \tag{3.10}$$

Where: $RTTi_{lifeTime}$ is the lifetime of $RTTi$; TS_{RTTi} and $TS_{RTT(i+1)}$ are the timestamps of $RTTi$ and $RTT_{(i+1)}$ respectively.

3.5.3 SEDLANE VALIDATION

In order to validate SEDLANE, we tested it using different wireless ad hoc network scenarios. The validation test-bed is shown in Figure 3.6.

We use *TTCP (Test TCP)* [47] as a TCP traffic generation tool between the source and destination. The laptop in the middle runs SEDLANE and is configured to be the ip routing gateway to the other two. Thus, we guarantee that the traffic exchanged between the end point laptops must pass through the SEDLANE station, where we deploy the emulation rules. We send 16Mbytes of TCP data using *TTCP*. We analyze the traffic using *Tcpdump* [48] and *Tcptrace* [49]. *Tcpdump* is a powerful open source tool that allows us to sniff network packets and make some statistical analysis out of those dumps. *Tcptrace* is an analysis tool. It can take the files produced by several popular packet-capture programs, such as *Tcpdump,* as input. *Tcptrace* can produce several types of output containing information on each connection, such as elapsed time, data bytes and segments sent, received and retransmitted, round trip times, window advertisements, and connection throughput. It can also produce a number of graphs for further analysis.

We configured our simulation scenarios using NS-2 (in order to get TCP trace files) as follows: each simulation consists of a 20 nodes network confined in a (670m x 670m) area. 14 TCP connections were established (ftp traffic used with a packet size of 1000 bytes). The simulation time is set to 400 seconds. We used the Optimized Link State ad hoc Routing protocol OLSR [40] as an ad hoc routing protocol in our simulations, since it was developed for mobile ad hoc networks and can be considered as stable in such environments. We test both simultaneous and sequential operation modes of SEDLANE. In our simulations, we vary the nodes' mobility rate (5 and 30 m/s) to get different loss and delay variations.

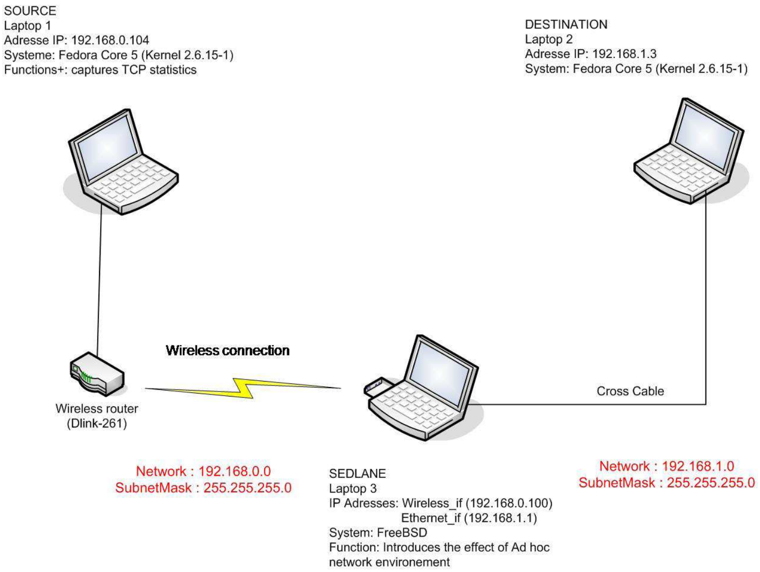

SOURCE
Laptop 1
Adresse IP: 192.168.0.104
Systeme: Fedora Core 5 (Kernel 2.6.15-1)
Functions+: captures TCP statistics
DESTINATION
Laptop 2
Adresse IP: 192.168.1.3
System: Fedora Core 5 (Kernel 2.6.15-1)
Wireless connection
Cross Cable
Wireless router
(Dlink-261)
Network : 192.168.0.0
SubnetMask : 255.255.255.0
SEDLANE
Laptop 3
IP Adresses: Wireless_if (192.168.0.100)
Ethernet_if (192.168.1.1)
System: FreeBSD
Function: Introduces the effect of Ad hoc
network environement
Network : 192.168.1.0
SubnetMask : 255.255.255.0

FIGURE 3.6. SEDLANE VALIDATION TEST-BED

In order to evaluate SEDLANE performance, we compare the data extracted from the NS-2 trace files (simulation results) with SEDLANE results (emulation results). The output results are captured by *Tcpdump* at the sender side, and then analyzed by *Tcptrace*. In the sequential operation mode, we examine two parameters: evolution of RTT values and average connection throughput between the communicating end points. While in the simultaneous operation mode, we evaluate the probability distribution of RTT values with respect to the total amount of data transmitted.

3.5.4 SEDLANE RESULTS

In this section, we analyze SEDLANE validation results obtained using the real test-bed configuration shown in Figure 3.6 for the two operation modes.

3.5.4.1 SEDLANE SEQUENTIAL OPERATION MODE

In this operation mode, *ipfw* pipes are configured and applied according to NS-2 TCP trace file. Thus, the RTT evolution applied by SEDLANE should follow the RTT transitions sequence in the input file. In addition, the time during which an RTT value will be applied on data packets should correspond to the RTT lifetime in the NS-2 TCP trace file. Figure 3.7, Figure 3.8, Figure 3.9, and Figure 3.10, compare the performance of the simulations done by NS-2 and the experiments done using SEDLANE. We compare the evolution of the RTT values over the connection (Figure 3.8, and Figure 3.10) and the TCP throughput (Figure 3.7, and Figure 3.9) in each case. Both Figures 3.8 and 3.10 confirm that SEDLANE highly respects the delay of data packets (RTT values) over the connection and the duration of each transition as found in the NS-2 trace file. When emulating losses and delays within an ad hoc network, it is expected to view the effect of these parameters on the end-to-end average throughput of the emulated connection. So, this behavior should be the same in SEDLANE as in the simulation results. We calculate the average throughput from the NS-2 TCP trace file (simulation result) and compare it with the average throughput of the communicating nodes in our test-bed (SEDLANE). The obtained results are drawn in Figure 3.7 and Figure 3.9. It can be shown from these figures that, the effect of losses and delays imposed by SEDLANE lead to the same degradation behavior as in the simulations regardless of the data transmission rate. It is worth to note that, the total number of data bytes transmitted by NS-2 simulation scenarios is not necessarily the same as in the test-bed scenarios. However, we get always the same average throughput behavior which is our target.

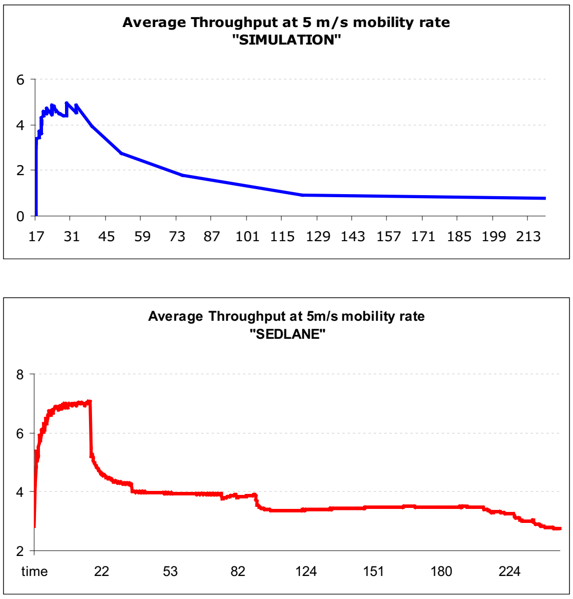


FIGURE 3.7. AVERAGE THROUGHPUT AT 5M/S MOBILITY RATE

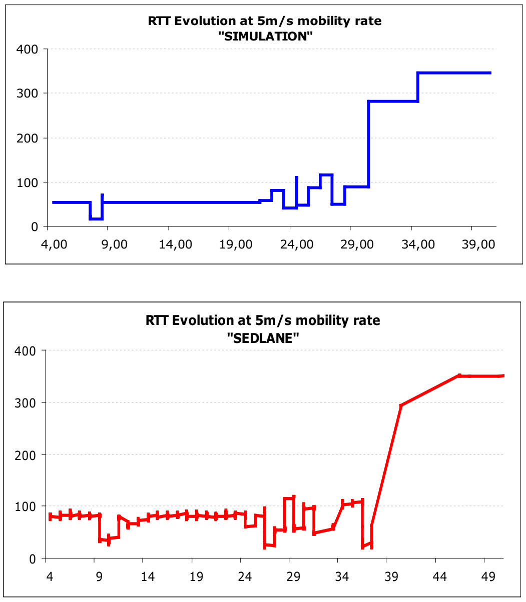
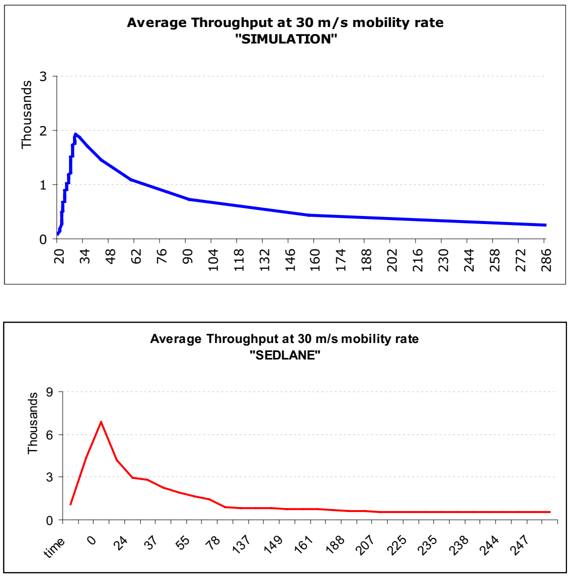


FIGURE 3.8. RTT EVOLUTION AT 5M/S MOBILITY RATE



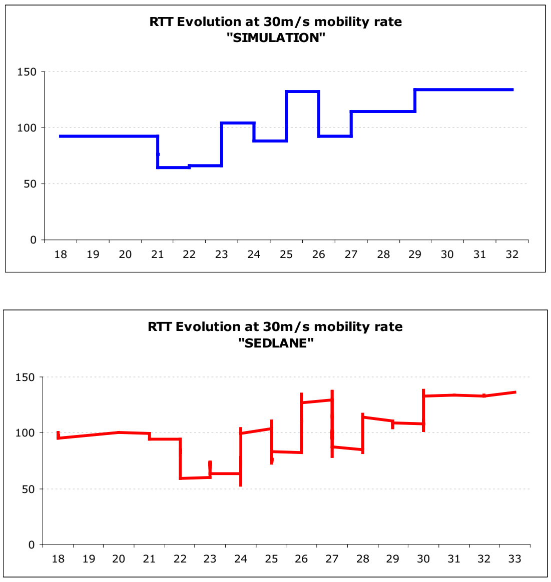


FIGURE 3.9. AVERAGE THROUGHPUT AT 30M/S MOBILITY RATE

FIGURE 3.10. RTT EVOLUTION AT 30M/S MOBILITY RATE

The lifetime of an RTT value is the time during which the RTT value stays unchanged. Each RTT value can have a different lifetime according to the network environment characteristics. In SEDLANE sequential mode, the RTT values are applied in a sequential manner (one after another) according to the lifetime of each RTT value traced within the simulation trace file. Thus, respecting the exact values of RTT lifetime is an important aspect to get exactly the same performance as in the performed simulation.

Figure 3.11 shows that SEDLANE is able to accurately calculate and apply the RTT values lifetimes that are found within the simulation trace file. The RTT lifetimes traced from the simulation trace file are identical to those applied by SEDLANE to manipulate the data packets over the connection (as can be verified from the figure).

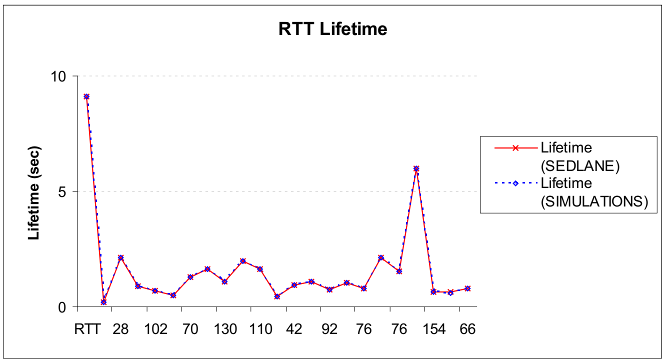
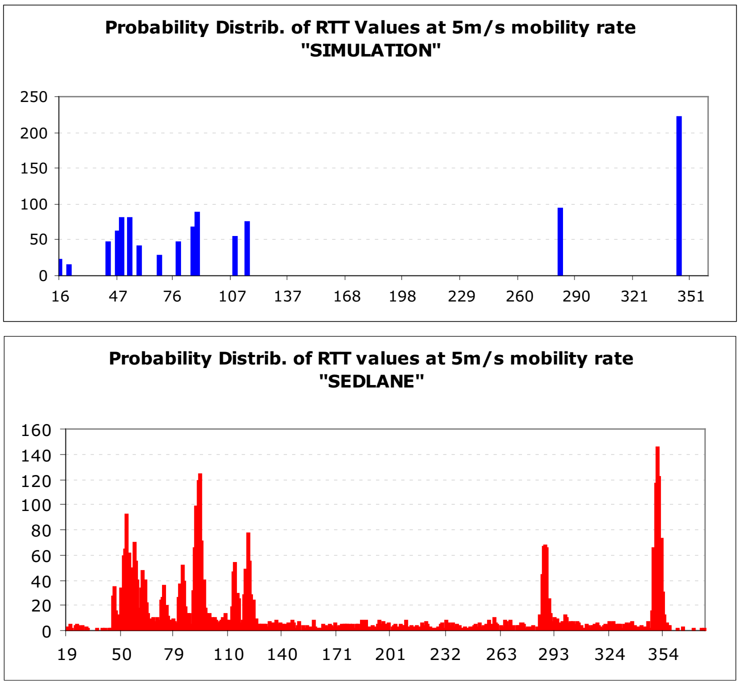


FIGURE 3.11. RTT LIFETIME IN SECONDS AT 5 M/S MOBILITY RATE

3.5.4.2 SEDLANE SIMULTANEOUS OPERATION MODE

In SEDLANE simultaneous operation mode, the *ipfw* pipes are configured according to NS-2 TCP trace file but applied according to its calculated probability. This probability reflects the amount of data bytes transmitted for each RTT transition with respect to the total amount of data bytes transmitted over the emulated connection. In this operation mode, SEDLANE follows the default *ipfw* operation guidelines. Figure 3.12 shows that SEDLANE respects the distribution of RTT probabilities within the used NS-2 TCP trace file. Here again, the results are confirmed regardless of the amount of data bytes transmitted within the simulation scenarios. Note that simultaneous operation mode does not provide a fine grain emulation of the scenario represented by the TCP trace file.; as it uses a probabilistic approach to apply the *ipfw* rules.



Probability Distrib. of RTT Values at 5m/s mobility rate "SIMULATION"

Probability Distrib. of RTT values at 5m/s mobility rate "SEDLANE"

FIGURE 3.12. PROBABILITY DISTRIBUTION OF RTT VALUES AT 5M/S MOBILITY RATE

In Figure 3.13, we show the accuracy of SEDLANE when applying the exact data loss ratios at each RTT value within the simulation trace file. We can see clearly from the figure that SEDLANE has the ability to well calculate and impose the right values of loss ratios over the connection and at the exact time (the corresponding RTT value).

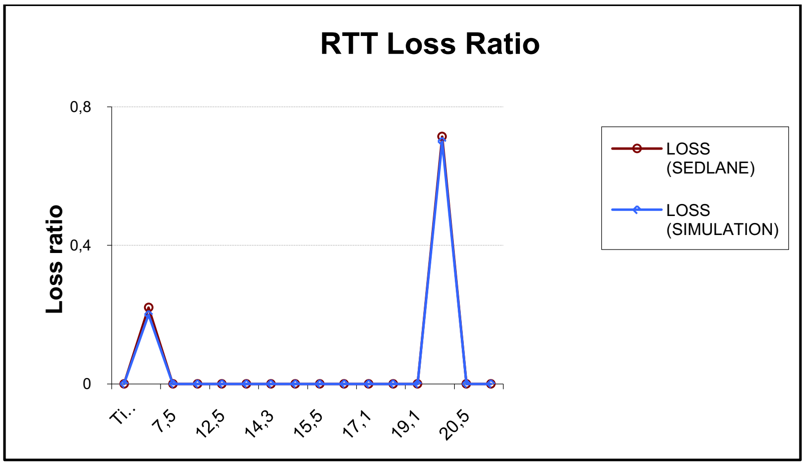


FIGURE 3.13. RTT LOSS RATIO VALUES A T 5M/S MOBILITY RATE

3.5.5 SUMMARY

In this chapter, we proposed a new and simple emulation tool, SEDLANE. SEDLANE takes advantage of network simulation tools to control the desired network conditions, since it uses the simulations trace files to introduce the same effects on real data traffic in a simple and inexpensive test-bed configuration. SEDLANE uses both data losses and RTT delay values as network performance parameters in order to emulate the targeted connection. SEDLANE implements many calculation algorithms that extract the necessary data information from the simulation trace file, and calculate the connection parameters used to configure the *ipfw* rules or the communication channels applied to manipulate traffic data packets over the emulation test-bed. SEDLANE can be used in two different operation modes: simultaneous and sequential operation modes. The validation results confirm that SEDLANE is capable of emulating the different network parameters and having an accurate emulation of network performance. The results also show that SEDLANE can emulate wireless ad hoc network scenarios and gives the same performance in terms of delay and data packet losses as in the network simulator. Thus, SEDLANE helps in testing and evaluating many wireless ad hoc network features (such as mobility rates, ad hoc routing protocols, and transmission control protocol).

3.6 CONCLUSION

Emulation is an evaluation and testing approach that can be used in order to build realistic and inexpensive test-bed configurations and conduct the experiments that cannot be done through simulations. SEDLANE is a new multi-hop wireless ad hoc network emulator that is used to test and evaluate wireless ad hoc networks' protocols and applications. SEDLANE helps emulating wireless ad hoc networks of any size using only one physical node. It can emulate nodes' mobility without needing to physically moving the communicating machines. SEDLANE can be used to test the applications and protocols that run over the transport layer. SEDLANE emulates the network connection through the data packets delays and loss rates over the connection. There are different versions of SEDLANE, which were developed in order to facilitate its usage. The results show that SEDLANE has the ability to efficiently emulate the connection characteristics and respect the emulated network conditions.

Building a complete test-bed configuration that meets all the conditions needed and to vary them to get the desired results would be costly. Thus, using SEDLANE within a simple and realistic test-bed configuration of only three physical nodes and having the effect of an entire wireless ad hoc network environment is the equivalent low-cost and non-complex solution of building the entire test-bed.

As a future work, we aim to extend SEDLANE in order to facilitate the evaluation of Peer-to-peer (P2P) applications that involve multiple parties. This extension should help emulating multiple connections between peers within the wireless ad hoc network and to use more complex simulation configurations.

In the next chapter, we will show how SEDLANE could be used to evaluate and measure one of the wireless ad hoc networks' nodes related performance parameters: the computational energy cost of TCP within wireless ad hoc network nodes. Indeed, this parameter cannot be measured using simulations.

CHAPTER 4. COMPARATIVE STUDY OF TCP VARIANTS WITHIN WIRELESS AD HOC NETWORKS

4.1 INTRODUCTION

TCP (Transmission Control Protocol) is considered the most popular reliable transport protocol today. Several different TCP variants have been developed for both wired and wireless networks. Improving the performance of TCP within wireless ad hoc network environments is one of the major aspects of this thesis. It is necessary; however, that we first deeply evaluate the performance of existing TCP variants in the context of wireless ad hoc networks.

We present in this chapter a complete performance study and analysis of the most known TCP variants. We aim to study some TCP variants that employ different congestion control algorithm approaches in order to study the effect of such algorithms on TCP performance within wireless ad hoc network environments. Some of these variants were developed to be implemented within wired networks, while others were developed for wireless environments. The studied variants in this chapter are the most common ones: TCP New Reno, TCP SACK, TCP Vegas, and TCP Westwood. These variants and the performance of each will be discussed in details.

In this chapter, we try to answer a couple of questions: Could the existing TCP Variants be used for wireless mobile ad hoc networks? What will be the performance of each in wireless ad hoc networks? The answers to these questions conclude the purpose of our work in this chapter. As a conclusion of this discussion, we show the necessity of having a new TCP variant for the modern wireless ad hoc networks and the possibility to capitalize on the existing variants to develop this new variant.

The chapter is organized as follows: The first section discuses the related works. Then, we present an overview of the different TCP variants evaluated in our performance study, their algorithms, and functionalities. In the third section, we describe the methodology we use to evaluate the performance of each of these TCP variants (simulations and realistic test-bed). Finally, we show the results obtained through our study and analyze them which lead us to the conclusion of this chapter.

4.2 RELATED WORK AND MOTIVATIONS

During the last few years, many researchers have been studying TCP performance in terms of energy consumption and average throughput within wireless mobile networks [1] [2] [3] [50]. Due to the specific issues related to wireless ad hoc networks, such as nodes' mobility, bandwidth and energy constrains, 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 (energy consumption and/or throughput) within such environments. In [3], the authors studied the energy consumption and throughput of three TCP variants (TCP Reno, TCP New Reno, and TCP SACK) through test-bed experiments. They evaluated TCP total energy consumption and subtracted the idle energy consumption of the TCP nodes. However, the authors applied random RTT delays and packet losses in their experiments. We note here, that RTT delays and data packet loss ratios over wireless ad hoc networks are highly correlated. Therefore, studying TCP using random values does not really reflect the behavior of wireless ad hoc network environment. The authors in [2] evaluated the energy consumption of the standard TCP over wireless links with and without the SACK option. Their study was conducted considering different wireless-related loss situations (e.g. interference). In this work, the authors do not study TCP throughput over the connection or compare their results with other TCP variant performance.

In [4], the authors studied the effect of mobility on TCP performance in terms of throughput within mobile ad hoc networks, through simulations. They conclude that TCP throughput drops significantly in the case of link-failure-induced losses due to nodes' mobility. However, in this work, the authors do not evaluate TCP energy consumption. Additionally, they consider only the standard TCP variant in their study. In [18], the author investigates the behavior of the standard TCP implementation in terms of throughput and compares it with that of ATCP [17], considering different data packet loss situations. ATCP is intended to enhance the performance of TCP within wireless mobile ad hoc networks. This study is conducted through simulations. However, the author in [18] , as well as, in [23] does not evaluate the TCP energy consumption.

In [26], the authors study the performance of different TCP variants (TCP Tahoe, TCP Reno, TCP New Reno, and TCP SACK) within wireless static ad hoc networks taking into consideration different ad hoc routing protocols (DSDV, DSR, and AODV). Using simulations, they evaluate both TCP throughput and communication energy consumption. However, the performance in case of link failure due to nodes' battery depletion was not taken into consideration. The computational energy cost of the standard TCP implementation was studied in [50]. 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 connection, since both RTT delays and data packet losses are correlated. Additionally, the authors in [50] do not evaluate the TCP throughput and do not study other TCP variants.

None of the above mentioned studies considers evaluating TCP throughput, communication energy cost and computational energy cost in the same work. Also, using random values to represent RTT values or data packet losses cannot reflect accurate ad hoc network characteristics since losses and delays are correlated. In this chapter, we aim to make a clear and complete comparison study between the most common TCP variants (New Reno, SACK, Vegas, and Westwood) under different data packet loss situations. This study incorporates communication and computational energy consumption as well as throughput. Also, we extend the experimental test-bed used in [50] [3]to measure the computational energy cost by using a realistic ad hoc network emulator (SEDLANE[3] [51]) 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. Here, however, we incorporate the computational energy cost of the TCP congestion control algorithms within the energy model of NS-2 to be capable of evaluating the total energy consumption of TCP variants (both communication and computational) using simulations.

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

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

4.3 COMPARATIVE STUDY OF TCP VARIANTS

In this section, we study the performance of TCP variants mentioned earlier in terms of communication energy consumption, computational energy cost, and average throughput taking into consideration different data packet loss events. The studied loss situations 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. For both the throughput and communication energy consumption, we obtain them through simulations using Network Simulator version 2 (NS-2) [27]. On the other hand, the computational energy cost cannot be traced using simulations (network simulators only include the communication energy cost). The node's level characteristics are not yet applied at network simulators. Hence, in order to measure the computational energy cost of TCP variants, we build a realistic test-bed with a special configuration set-ups. The simulations and the test-bed configuration, as well as the methodology used to measure the computational energy cost are detailed below.

4.3.1 SIMULATION 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 these 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. Each scenario is described in this section. First, let's describe the reason behind the choice of the ad hoc routing protocol used in our evaluations.

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 the congestion situation over the TCP connection. This leads to more congestion control actions triggered to recover from the packet losses.

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

TABLE 4.1. COMPARATIVE STUDY OF AD HOC ROUTING PROTOCOLS

Table 4.1 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. The comparison is shown for four main ad hoc routing protocols: Ad hoc On-Demand Distance Victor (AODV) routing protocol [52], the Dynamic Source Routing (DSR) protocol [53], the Destination-Sequenced Distance Vector (DSDV) routing protocol [54], and the Optimized Link State Routing (OLSR) protocol [55]. The values depicted in Table 4.1 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 used route is broken. Contrarily, proactive protocols (DSDV and OLSR) needs 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 MANET. Therefore, we decided to run the best MANET configuration. So, according to the table above, we choose to run our simulations using 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 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 [52]. Extending our performance evaluation to other routing protocols is outside the scope of our work, though, this would constitute an interesting evaluation study.

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

1) Congestion scenario: In this scenario, we create a congested node at the middle of a five-node topology by generating three TCP data traffic flows that must pass by this intermediate node to reach the other communicating end point (Figure 4.1). 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 (Figure 4.1).

2) Interference between neighboring nodes: In this case, two TCP connections are established in parallel. The main TCP connection (TCP data flow 1 in Figure 4.2) is disturbed by the interferences generated by the second TCP connection (TCP data flow 2 in Figure 4.2). Indeed, the node acting as forwarder for the main TCP connection is placed within the interference range of the second TCP connection sender. So, this situation creates interference and thus data packet losses.

3) Link loss 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 employ routes with different number of hops (Figure 4.3). Thus, each time 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 generated by the link loss and the new chosen route.

4) Signal loss scenario: This scenario illustrates the situation where the wireless signal is not stable. The communicating nodes loose the connection due to signal loss then they resume the communication when the signal comes back. Signal losses are generated by moving one of the intermediate nodes out of the radio range of its connection neighbors (Figure 4.4). Signal loss event is repeated twice within the described scenario, the first one is for duration of 25 seconds and then for 15 seconds. Indeed, the simulations were repeated considering signal loss durations ranging from 2 seconds to 25 seconds and the results were similar.

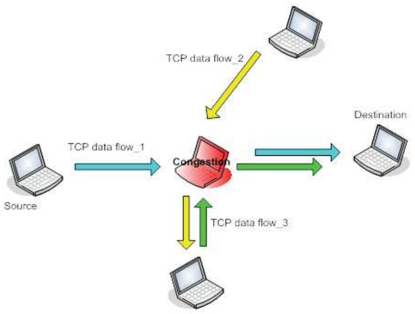


FIGURE 4.1. NETWORK CONGESTION SCENARIO

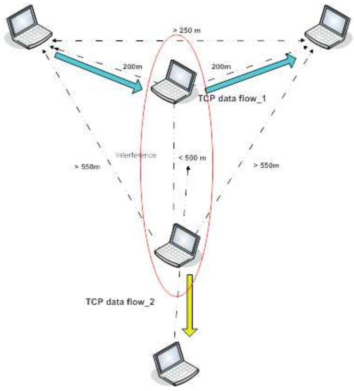


FIGURE 4.2. INTERFERENCE SCENARIO

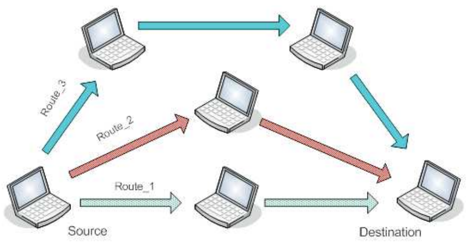


FIGURE 4.3. LINK FAILURE AND COMMUNICATION ROUTE CHANGES

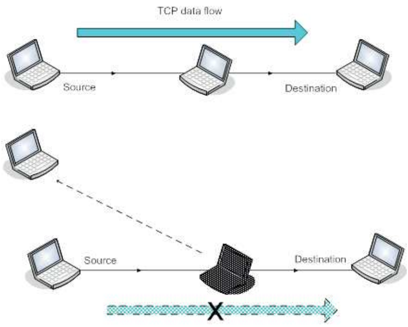


FIGURE 4.4. SIGNAL LOSS SCENARIO

4.3.2 TCP'S THROUGHPUT AND COMMUNICATION ENERGY CONSUMPTION EVALUATION

According to the scenarios described above, we, first, study two TCP performance parameters. The first is the energy consumed in transmission, reception, forwarding and retransmission of data packets. This energy is calculated in proportion to the amount of received data, i.e. energy consumed per received bit. The second parameter studied is the TCP's connection throughput. Each simulation was run ten times in order to obtain the results discusses below.

More precisely, we discuss in details the results obtained through NS-2 simulation tool. We evaluate and compare both throughput and communication energy consumption of the studied TCP variants taking into consideration different data packet loss scenarios (network congestion, interference, link failure and signal loss).

4.3.2.1 NETWORK CONGESTION

From Figure 4.5 and Figure 4.6, we can see that TCP SACK can be considered as the best performing variant in terms of connection throughput. This is due to the Selective ACK feature that allows TCP SACK to terminate the retransmission of lost packets more quickly than TCP New Reno (which has to wait for all partial ACKs to identify the missing segments at the receiver). In addition, TCP SACK has the ability to resend only the lost data packets without retransmitting the entire data window that contains the lost packets. TCP Westwood has the best performance among others 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 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 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 and leads to a waste of bandwidth. 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.

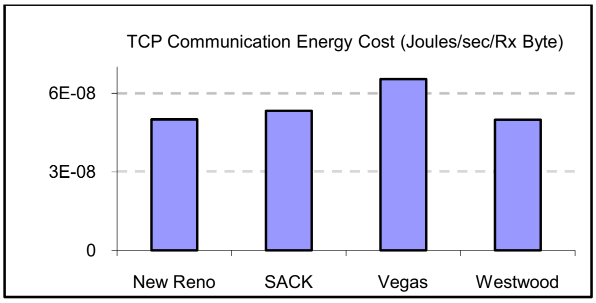


FIGURE 4.5. COMPARISON OF TCP COMMUNICATION ENERGY CONSUMPTION

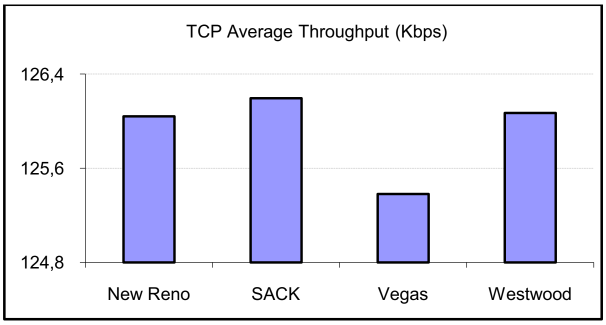


FIGURE 4.6. COMPARISON OF TCP THROUGHPUT

4.3.2.2 INTERFERENCE

Figure 4.7 and Figure 4.8 show, that TCP SACK has the highest connection throughput compared to the other studied variants. The ability of TCP SACK to resend only the lost data packets one after another without the need to wait for RTT before sending them (as in TCP New Reno), leads to better throughput over the connection.

Both TCP Vegas and TCP Westwood outperform TCP New Reno in terms of both communication energy cost and connection throughput, since both variants have the ability to adapt their CWND after data packet loss according to the network conditions. Although that, both TCP Vegas and TCP Westwood have almost a comparable performance in terms of communication energy consumption, TCP Vegas outperforms TCP Westwood in terms of connection throughput since it is capable to recover from data losses at the reception of the first duplicate acknowledgment with no need to wait for the third duplicate acknowledgment as in TCP West wood.

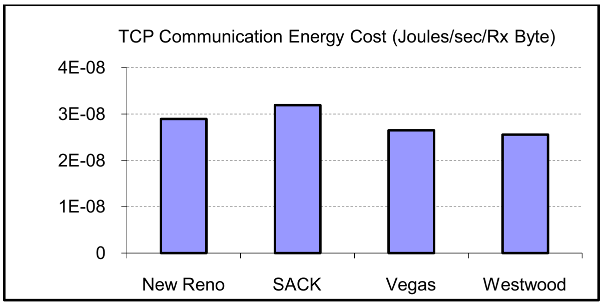


FIGURE 4.7. COMPARISON OF TCP COMMUNICATION ENERGY CONSUMPTION

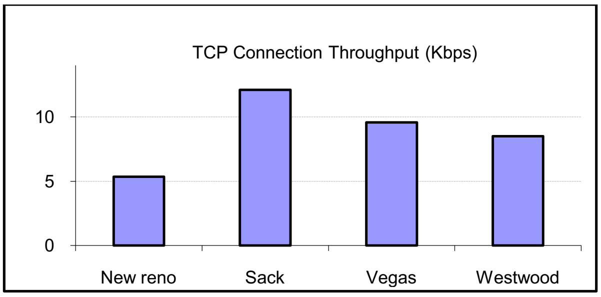


FIGURE 4.8. COMPARISON OF TCP THROUGHPUT

4.3.2.3 LINK FAILURE

In mobile wireless ad hoc network, it is obvious that the nodes are prone to broken communication paths between the communicating end points (due to mobility or depletion of nodes' batteries). This type of data loss situations results in burst data packet losses over the connection. Although, burst data losses might be the result of network congestion, treating them consistently as such might result in suboptimal performance.

The simulations results show that all TCP variants, in case of link loss events, detect data packet losses through RTO. Thus, they all perform similarly by backing off and entering Slow-Start phase. This can be seen in the Figures (Figure 4.9 and Figure 4.10) below. The four TCP variants are shown to have comparable performance levels in terms of both communication energy consumption and throughput. 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.

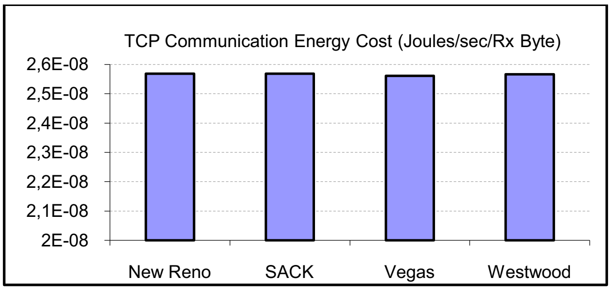


FIGURE 4.9. COMPARISON OF TCP COMMUNICATION ENERGY CONSUMPTION

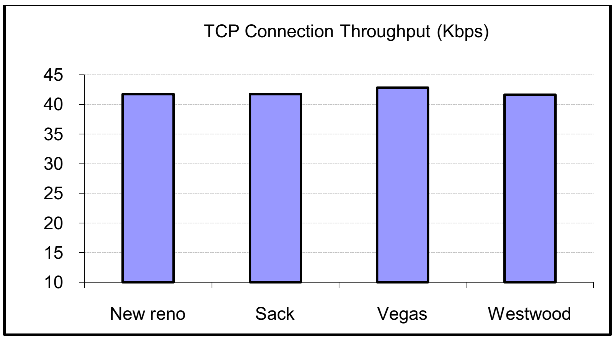


FIGURE 4.10. COMPARISON OF TCP THROUGHPUT

4.3.2.4 SIGNAL LOSS

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.

Signal loss can be considered as a special case of link failure case. In signal loss situation the network becomes partitioned due to missing wireless channel signal. This means that there is no possible connection between the two partitions unless the wireless signal comes back. In this case, the results (Figure 4.11 and Figure 4.12) show that all TCP variants, studied in this work, have almost the same performance in terms of both communication energy consumption and throughput. 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.

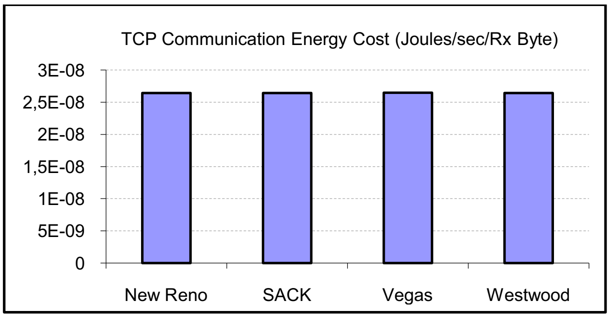


FIGURE 4.11. COMPARISON OF TCP COMMUNICATION ENERGY CONSUMPTION

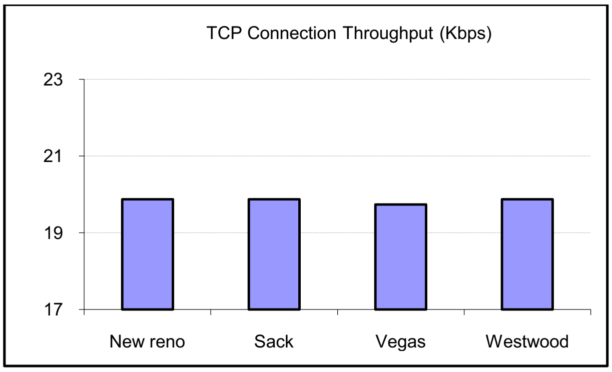


FIGURE 4.12. COMPARISON OF TCP THROUGHPUT

4.3.3 TCP COMPUTATIONAL ENERGY CONSUMPTION MEASUREMENTS

In order to complete the results obtained by simulations and to get the overall performance results of each of the studied TCP variants. We study now the third performance parameter of TCP which is the node's computational energy cost. The computational energy cost of TCP is the energy consumed by the node's CPU unit in order to perform the various copy operations, checksums computation, responding to timeouts and triple duplicate ACKs, adjusting timers, and other book keeping operations. This cost is thus linked to the execution of the different TCP congestion control algorithms (Slow-Start, Fast Retransmit/Fast Recovery, and Congestion Avoidance). It is important to note that this is complementary to the evaluation realized in the previous section that analyzed the radio-related energy cost of TCP variants (i.e. the energy consumption due to the transmission, retransmission and forwarding of TCP segments by ad hoc nodes).

Here, we also evaluate the same four major TCP variants: TCP New-Reno, TCP Vegas, TCP SACK and TCP Westwood. In order to measure the computational energy cost while executing TCP's different congestion control algorithms, we implement different data packet loss models (the same data loss scenarios used in the simulations above). Measuring the node-level energy consumption is realized using a realistic test-bed configuration. This configuration should introduce the effect of a real wireless mobile ad hoc network environment (i.e. realistic data packet delays and losses). We introduce such effects using SEDLANE, our MANET delay and packet-loss emulation tool presented in CHAPTER 3. Recall that SEDLANE uses NS-2 simulation results in order to generate realistic data packet delays and losses in MANETs. The use of such a hybrid approach enables our evaluation to take advantage of both simulation and test-bed experiments approaches. Hence, thanks to SEDLANE, the effect of different data packet loss models (congestion, interference, link failure, and signal loss) and ad hoc routing protocol are introduced.

Our study, in this section, has multiple benefits. The main foreseen benefits that motivated our work are: (1) to enable the understanding of the energy consumption model of TCP at the CPU level and thus to facilitate the future development of new TCP congestion control algorithms for MANETs that are energy-efficient; (2) to give to other researchers working on analytical modeling of TCP a set of results to develop energy models for the computational cost of TCP congestion control algorithms; and (3) to allow the incorporation of our node-level energy models into any network simulators (NS-2 for example) so as to obtain the overall energy cost, computational and radio costs of TCP connections. Currently, network simulators only include the radio energy cost.

4.3.3.1 MEASUREMENT METHODOLOGY

The methodology used in our computational energy consumption measurements extends the one previously used in [50] as we add the use of SEDLANE. Our test-bed configuration (as shown in Figure 4.13), is composed of a laptop as a sender end while the receiver end side is a Personal Computer (PC). Between the communicating nodes we implement SEDLANE (on a second 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 packet transmission and reception. The difference between the two measured values will be the computational energy consumption. Obviously, the measurements are taken at the TCP sender, since the TCP algorithms are executed at the sender side.

In order to correlate this computational energy consumption to the TCP operations, we use a minimal Linux distribution; in which we turn off the display, the power management and the x-server. This allows for minimizing the effect of any other running applications on the measured current. The reason for turning off power management as described in [50] is the fact that it helps better to determine the current draw which corresponds to TCP code execution. We do not use the battery in the laptop because, as noted in [56], avoiding the use of battery allows for a more steady voltage to be supplied to the device which allows a more accurate measurement. Thus, all the processes/daemons/features that are not necessary to TCP operations are simply removed from the Linux distribution making it strictly minimal. By taking all these precautions, we ensure that the remaining energy consumption is due to TCP congestion control algorithms execution and timer adjustments.

Energy consumption is determined by measuring the input voltage and current draw using two Agilent 34401A digital multi-meters that have a resolution of one millisecond. 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 [57] 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. Sycard PCCextend 140 CardBus extender card is a debug tool for development and testing of PC cards and hosts. Figure 4.14 shows the real measurements test-bed implementation.

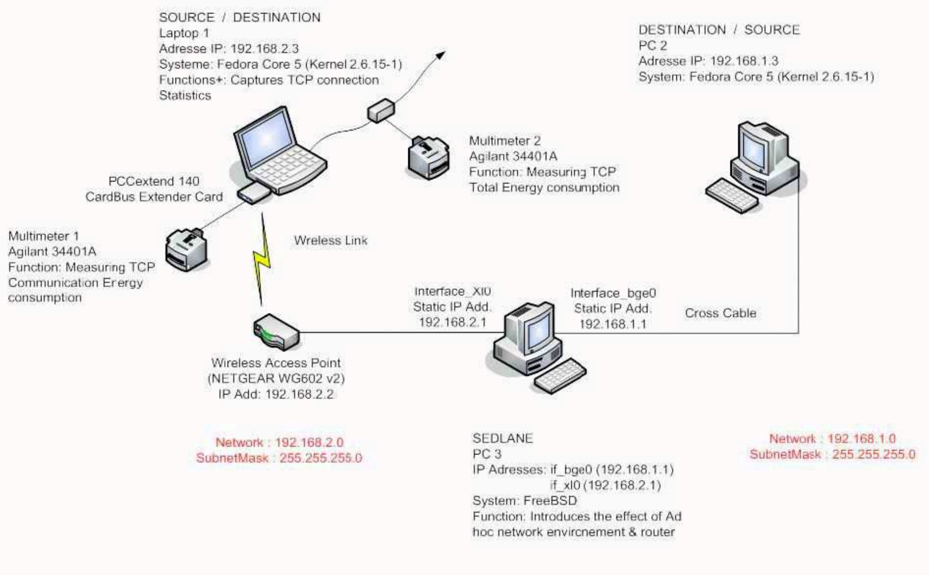


FIGURE 4.13. TCP ENERGY CONSUMPTION MEASUREMENTS TEST-BED

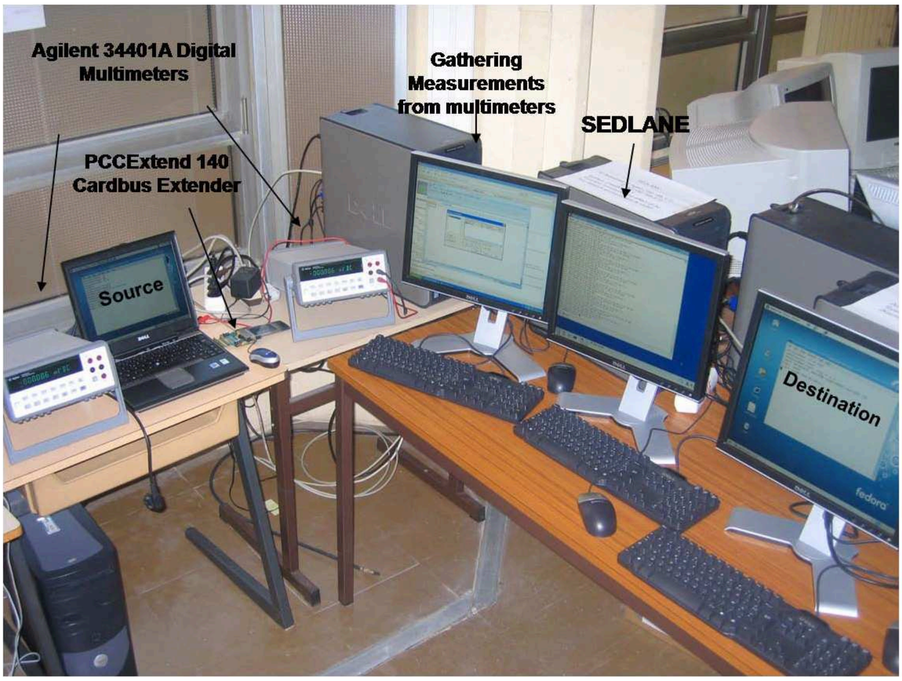

Agilent 34401A Digital Multimeters
Gathering Measurements from multimeters
SEDLANE
PCCExtend 140 Cardbus Extender
Source
Destination

FIGURE 4.14. THE IMPLEMENTED MEASUREMENTS TEST-BED

In this study, we use the same scenarios used in NS-2 simulation study: network congestion, interference, link failure, and signal loss. By using the same scenarios, we intend to complement the results obtained by simulations with test-bed measurements to get the overall results for each TCP variant. Using simple network scenarios that define precise and deterministic data loss situations is done explicitly in order to study the exact reaction of TCP faced with each data loss situation separately.

4.3.3.2 TCP COMPUTATIONAL ENERGY COST CALCULATION

In order to calculate the TCP computational energy cost, we calculate first the total energy consumed at the laptop (the system), 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):

$$E_R = I_R * T * V_R \tag{4.1}$$

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 (5v).

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

$$E_T = I_T * T * V_T \tag{4.2}$$

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 (20v).

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

$$E_{idle} = I_{idle} * T * V_T \tag{4.3}$$

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 (20v).

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

$$E_{comp} = E_T - E_{idle} - E_R \tag{4.4}$$

4.3.3.3 TCP COMPUTATIONAL ENERGY COST RESULTS

Contrarily to previous studies that concentrated on the operating system, hardware and device-level energy consumption due to TCP [58], 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 computational energy cost. This is realized according to the different data packet loss models: network congestion, interference, link loss, or signal loss. For each TCP variant, the computational energy cost is calculated and compared to the end to end performance characteristics. Finally, we identify and briefly discuss a set of design features that a new TCP variant should have to be resource-efficient (energy and bandwidth) when used in MANETs. We note that the computational energy cost is measured at the sender 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.

4.3.3.3.1 TCP FUNCTIONS COMPUTATIONAL ENERGY COST

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.

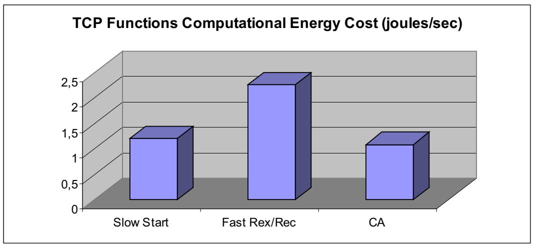


FIGURE 4.15. TCP COMPUTATIONAL ENERGY COST (JOULE/SEC)

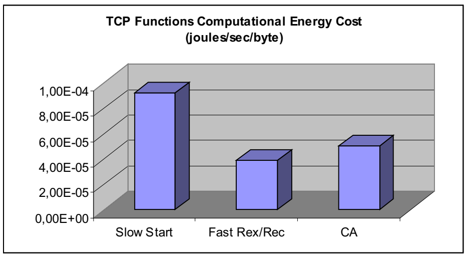


FIGURE 4.16. TCP ENERGY COST (JOULE/SEC/SENT BYTE)

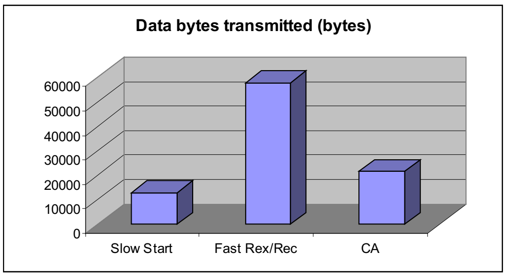


FIGURE 4.17. DATA BYTES TRANSMITTED/TCP FUNCTION

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 (Figure 4.15). Indeed, Figure 4.15 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 (Figure 4.16). 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 on one burst. Hence, the ratio energy-cost/data-sent remains low in the Slow-Start phase. 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 (Figure 4.15 and Figure 4.17). However, it has higher energy consumption per each sent byte compared to Fast Retransmit/Fast Recovery phase (Figure 4.16).

Figure 4.18 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 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.

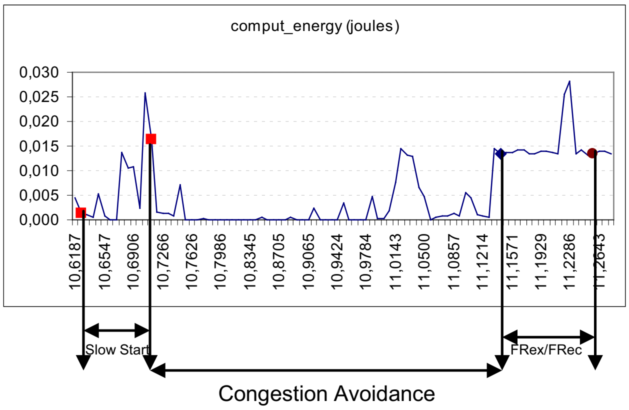


FIGURE 4.18. TCP NEW-RENO COMPUTATIONAL ENERGY COST EXAMPLE

4.3.3.3.2 COMPUTATIONAL ENERGY COST OF TCP VARIANTS

This section aims at comparing the four TCP variants (New Reno, SACK, Vegas, and Westwood) faced with different packet loss situations encountered in wireless ad hoc networks: network congestion, interferences, link loss, and signal loss. We choose to use the Ad-hoc On-demand Distance Vector (AODV) [52] as ad hoc routing protocol. AODV triggers a route discovery only when the sender needs to send data to the destination. This results in low routing protocol messages overhead over the network links.

4.3.3.3.2.1 EFFECT OF NETWORK CONGESTION

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. The results depicted in Figure 4.19 demonstrate that TCP Vegas has the higher 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 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 higher computational energy cost compared to all the other studied variants (Figure 4.19).

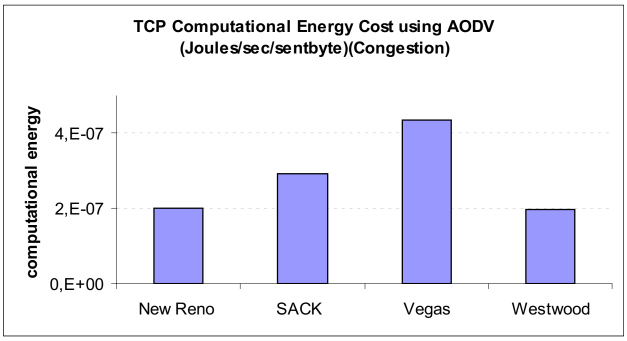


FIGURE 4.19. TCP COMPUTATIONAL ENERGY COST (JOULE/SEC/SENT BYTE)

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 that TCP Westwood and New Reno have almost the same performance in terms of energy consumption per each sent byte (Figure 4.19) 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).

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; Figure 4.19 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.

4.3.3.3.2.2 EFFECT OF TRAFFIC INTERFERENCE

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 Figure 4.19 and Figure 4.20, we can note that the computational energy cost is increased in the case of interferences compared to the case of congestion. The fact that TCP uses congestion control algorithms, means that TCP has the ability to better deal with network congestion conditions than traffic interference ones. Therefore, the misbehaviour of TCP in front of data packet losses due to interference leads to higher computational energy cost.

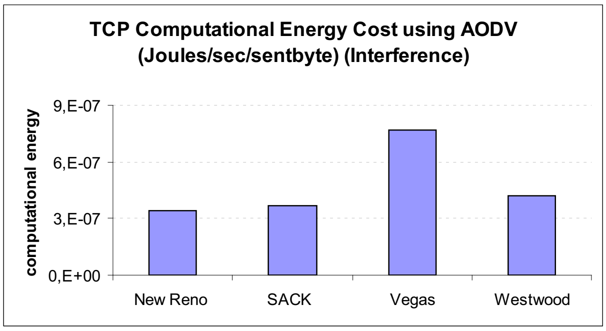


FIGURE 4.20. TCP COMPUTATIONAL ENERGY COST (JOULE/SEC/SENT BYTE)

Referring to Figure 4.20 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 complexity of the algorithms used by TCP Westwood and their continuous triggering by packet losses (it recalculates and modifies its data transmission rate after each data packet loss). 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 behaviour becomes more complex without significant improvement to the throughput.

Interestingly, we find that TCP New Reno and TCP SACK have almost the same performance in terms of computational energy cost. Although that 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 of the use of Selective acknowledgements.

4.3.3.3.2.3 EFFECT OF LINK LOSS

Figure 4.21 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.

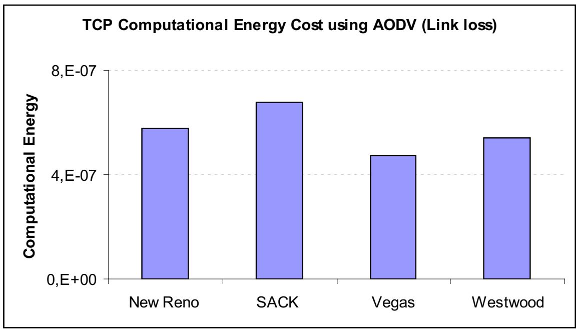


FIGURE 4.21. TCP COMPUTATIONAL ENERGY COST (JOULE/SEC/SENT BYTE)

In case of Link Loss, 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. classify 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. TCP New Reno or TCP SACK are proved to be less rapid in that aspect and thus they consume more energy by staying in the Slow-Start phase and increasing their throughput slowly.

4.3.3.3.2.4 EFFECT OF SIGNAL LOSS

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's why 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.

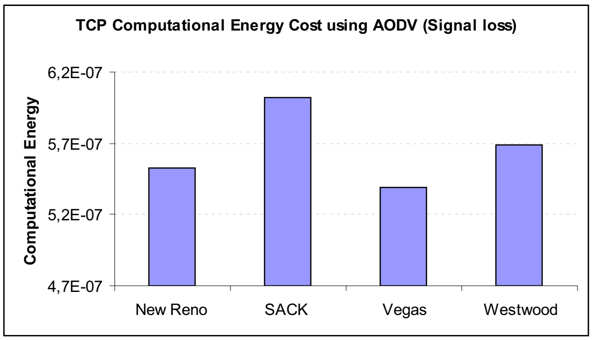


FIGURE 4.22. TCP COMPUTATIONAL ENERGY COST (JOULE/SEC/SENT BYTE)

Figure 4.22 demonstrates that TCP SACK is the worst performing variant in terms of computational energy cost, since it requires calculating lost packets at the sender side. In addition, the sender 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 side and consequently more computational energy consumption than both TCP New Reno and TCP Vegas. TCP Vegas, however, will identify 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 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.

Finally, we should note that, we have studied the signal loss event using different levels of signal loss duration time ranging from few seconds to few dozens of seconds, and in all cases, the four TCP variants had the same behaviour.

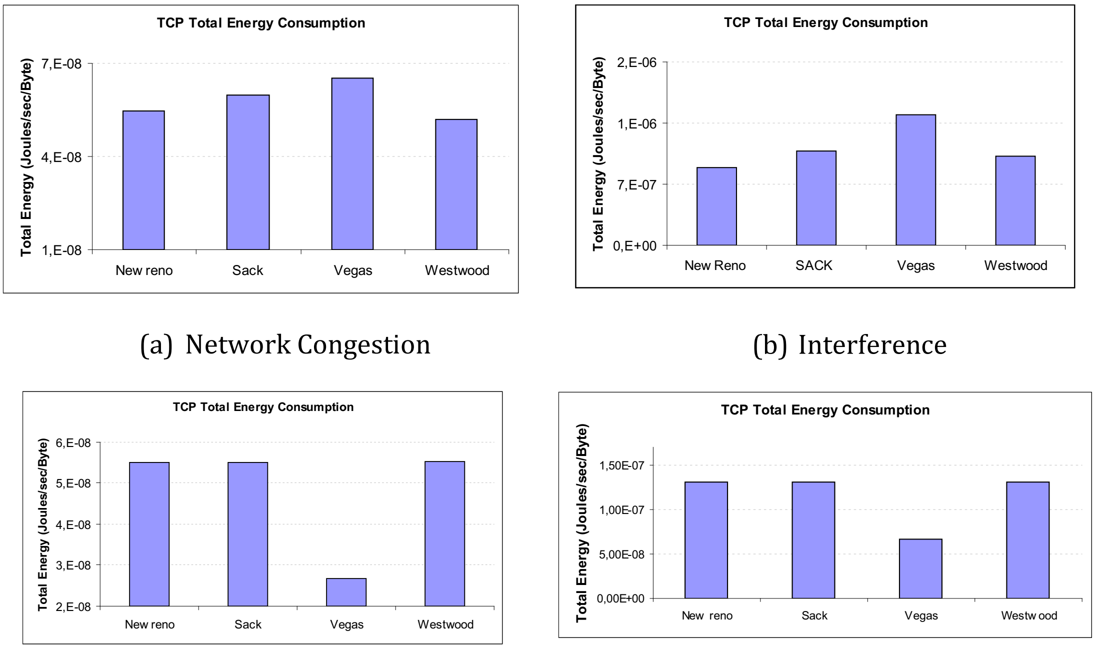
4.4 INCORPORATING TCP COMPUTATIONAL ENERGY COST INTO NS-2

NS-2 energy model does not include the node's TCP computational energy cost. It applies only the communication energy cost. That is why we thought of incorporating the results we had obtained through the earlier described test-bed into NS-2. By including the computational energy cost of TCP's algorithms (Slow-Start, Fast Retransmit/ Fast Recovery, and Congestion Avoidance), we can get the total energy consumption at network nodes using NS-2 simulations. We have integrated the obtained values into NS-2's energy consumption module and modified the code in such a way that each time TCP enters into a specified TCP function or algorithm it measures the time passed in this function and then calculates the computational energy consumption to subtract it from total available energy for each node. This is done with all TCP algorithms. After implementing our contribution into NS-2, we have repeated the above described scenarios in order to get the Total Energy Consumption (communication energy and computational energy cost) using the same TCP variants studied above.

We used the same loss scenarios (network congestion, interference, link failure, and signal loss) described in the evaluations above. The results are described below.

4.4.1 TCP TOTAL ENERGY CONSUMPTION

In this section, we discuss the TCP total energy consumption results obtained through simulations after incorporating our test-bed results in its energy module.



(a) Network Congestion

(b) Interference

(c) Link Failure

(d) Signal Loss

FIGURE 4.23. TCP TOTAL ENERGY CONSUMPTION

4.4.1.1 NETWORK CONGESTION

From Figure 4.23 (A), we can conclude that TCP Vegas has the worst performance among all the studied variants, as it has the highest total energy consumption and the lowest connection throughput (c.f. Figure 4.6). We have seen in the previous sections that TCP Vegas has the highest communication and computational energy consumption compared to the other studied variants. The fact that TCP Vegas calculates and updates its performance parameters (CWND and RTO) at the reception of each acknowledgment leads to this high computational energy cost. Also, as TCP Vegas tries to retransmit the lost packets at the reception of the first duplicate acknowledgement, it increases the communication energy consumption due to unnecessary retransmissions in some cases as seen earlier.

TCP Westwood has the least total energy consumption compared to other variants. TCP Westwood outperforms TCP New Reno due to its ability to adjust its data transmission rate according to the available bandwidth over the network. The difference between TCP Westwood and TCP Vegas is that TCP Westwood recalculates and adjusts its performance parameters only after loss detection and not at the reception of each acknowledgement (each RTT) as in TCP Vegas. Less calculations means less processing overhead at the CPU leading to less computational energy cost. TCP SACK consumes more energy than TCP New Reno due to the fact that TCP SACK at the sender side must identify the lost packets to be sent through the SACK option upon each received acknowledgments from the receiver. It is also clear that TCP SACK has an important transmission, reception, and forwarding overhead linked to the selective acknowledgment that is not employed in the other TCP variants.

We can conclude also, that TCP New Reno does not consume much energy because of its light processing overhead at the CPU level. TCP New Reno merely halves its data transmission rate (CWND) after losses recognized by 3 duplicate acknowledgements or minimizes it in case of RTO. In addition, TCP New Reno avoids any unnecessary retransmissions as it triggers its congestion control algorithm directly after data packet losses identified through RTO. Thus, light processing overhead and lower retransmission rate leads to lower total energy consumption at the sender side.

4.4.1.2 INTERFERENCE

Regarding the total energy consumption of TCP Variants in the case of data loss due to interference (Figure 4.23 (B)), we find that TCP Vegas has the highest total energy consumption. Packet interference induced losses can be classified as wireless channel errors induced losses. For TCP Vegas, the computational energy cost due to CWND and RTO calculations at the reception of each acknowledgement from the receiver leads to high total energy consumption.

TCP New Reno has the lowest total energy consumption per received byte since it is able to recover from multiple data loss during the same window. It has low computational energy cost sine it employs very simple operations: it halves its data transmission rate after data losses recognized by three duplicate acknowledgements. Contrarily to TCP New Reno, TCP SACK retransmits the lost packets without need to wait for the receiver acknowledgements, which leads to better usage of the available bandwidth and slightly higher energy consumption per received byte compared to TCP New Reno. This slight difference, again, is due to the computational, storage and communication overhead of the selective acknowledgments.

TCP Westwood has the ability to distinguish between congestion-induced data losses and wireless signal related data losses. This feature is of high interest in the case of interference. It allows for achieving a good throughput (Figure 4.8), but has a cost in terms of energy that makes its performance equivalent to TCP New Reno for this performance metric.

4.4.1.3 LINK FAILURE AND SIGNAL LOSS

In these cases (Figure 4.23 (C) and (D)), TCP Vegas has the best performance in terms of total energy consumption per each sent byte. The ability of TCP Vegas to adapt its data transmission rate and RTO, after link loss or signal loss recovery, helps to decrease the total energy consumption.

For TCP New Reno, and TCP SACK, they react similarly by backing off then entering slow-start phase, and decreasing data transmission rate to minimum. Though that, TCP Westwood has lower communication energy consumption than both TCP New Reno and TCP SACK (Figure 4.9), due to its ability to adapt its data transmission rate after data loss even, according to the network conditions. The processing overhead of TCP Westwood in order to identify the data packet loss neutralizes this advantage.

4.5 DISCUSSION

It was proved that the congestion control algorithm in TCP variants affects the throughput and energy consumption within wireless ad hoc networks. In order to enhance TCP performance within wireless 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 is 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 (bandwidth and energy resources).

Through simulations, we found that the ability of TCP to differentiate the data packet loss cause over the connection (as in TCP Westwood) helps to better react 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. Our second conclusion here concerns the ability to adjust TCP's data transmission rate after a loss episode (as in TCP Vegas). In some cases, such ability leads to better performance.

Also, we studied the TCP computational energy cost using a hybrid approach (i.e. using simulation results to configure a real test-bed configuration and perform accurate repeatable experiments). We also identified some tracks to follow in order to create a novel TCP variant that is energy-efficient in wireless ad hoc networks. For example, classifying data packet loss causes and efficiently adapting the TCP connection parameters (e.g. CWND and RTO) can help avoiding unnecessary data packet retransmissions. Consequently, this leads to better bandwidth utilization and lower communication energy consumption.

Our study shows that most of the TCP-related energy consumption occurs within the CPU of the sender node. Additionally, we found that, a significant part of the TCP communication energy consumption is due to unnecessary data packets retransmission because of: (i) inability of TCP to identify the data packet loss cause over the connection, or (ii) suboptimal TCP performance parameters (CWND, and RTO) adaptation. Knowing where the most TCP energy consumption occurs is the key improvement we can bring to TCP functions and their performance within wireless ad hoc networks. For example, helping TCP to avoid unnecessary retransmissions and minimize CPU calculations will help in optimizing TCP energy consumption and will lead to better usage of the network's available bandwidth.

4.6 CONCLUSION

In this chapter, we conducted a complete performance study of four TCP variants (TCP New Reno, TCP SACK, TCP Vegas and TCP Westwood), within wireless ad hoc network environment. We started by identifying the different data packet loss situations that TCP may confront within wireless ad hoc networks: (i) network congestion, (ii) interference, (iii) link failure, and (iv) signal loss. Our study concerns both TCP connection throughput and energy consumption (computational end communication energy consumption). TCP throughput and communication energy cost results were obtained through NS-2, and the computational energy cost results were obtained through a realistic test-bed implementation.

The above results show that TCP cannot cope with all data packet loss situations found within wireless ad hoc network (static or mobile). TCP's behavior needs to be enhanced in order to handle the different loss causes other than congestion. 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) and the available wireless bandwidth. The existing TCP variants that were originally developed for wired networks do not always behave optimally when confronted with the different data packet loss models within wireless ad hoc networks. Some show good performance in certain cases and bad performance in others. It is worth to note 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.

The problem of TCP and its most existing variants within wireless 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 need to design a new Loss Differentiation Algorithm (LDA) and a new Loss Recovery Algorithm. These new algorithms should have the capability to identify and deal with the most common packet loss models within wireless ad hoc networks environment. This will constitute the objective of the next chapter.

CHAPTER 5. TCP-WELCOME: TCP VARIANT FOR WIRELESS ENVIRONMENT, LINK LOSSES, AND CONGESTION PACKET LOSS MODELS

5.1 INTRODUCTION

According to the obtained results in the previous chapter, we found that none of the existing TCP variants is well adapted to deal with all data packet loss situations that can be encountered within wireless ad hoc networks. The results show that, the performance of TCP degrades significantly within wireless ad hoc networks. Moreover, some of the studied TCP variants perform well in certain cases while they perform badly in other cases. In a wired network environment, the nodes always have an energy supply source and do not necessarily need to be energy conservative, as it is the case with wireless ad hoc network nodes that are battery-dependent devices (the worst case is a wireless sensor network, WSN).

The results of the previous chapter show that the ability of TCP to distinguish among congestion-induced and wireless-related data losses (as in TCP Westwood) leads to an improved performance in some cases. However, TCP variants that incorporate a loss differentiation algorithm do not consider all types of data packet loss that can be encountered within wireless ad hoc network environments. In fact, they consider congestion-induced and wireless-channel-related losses only. While in wireless ad hoc networks, we have other data packet loss causes such as link failure that might be due to nodes' mobility (in case of wireless mobile ad hoc networks) or nodes' battery depletion. We also find that the TCP variant that is able to adjust its performance parameters (CWND and RTO) after data losses (as in TCP Vegas), in certain cases, can improve the performance within the network. However, adjusting TCP performance parameters regularly could result in performance degradation instead of improvement (especially in term of energy consumption). We also have to mention that the obtained results (in the last chapter) conclude that each data packet loss situation necessitates different data packet loss recovery reaction that optimizes TCP performance within the network.

The work done in this domain addresses the problems of TCP within wireless infrastructure networks where only the last hop is the wireless channel. Most of this work takes into consideration how to distinguish among wireless channel related and congestion data losses. Within wireless infrastructure networks, link failure affects only the end user (the concerned node), while within wireless infrastructure-less networks (such as wireless ad hoc networks); link failure affects the whole network as each node might forward data packets to other network's nodes. From this perspective comes the importance of dealing with link-failure-induced losses, since it is a very common situation within wireless ad hoc network environments.

In this chapter, we present a new TCP variant that is able to deal with the most common data packet loss causes within wireless ad hoc network environment. To summarize, the key features of this new TCP variant (TCP WELCOME, TCP variant for Wireless Environment, Link losses, and COngestion packet loss ModEls) are:

1- Loss Differentiation Algorithm (LDA) that has the ability to distinguish among these different data packet loss causes: (i) Network congestion induced losses, (ii) Wireless channel related losses (interference and signal loss), and (iii) Link failure induced losses.

2- TCP Loss Recovery Algorithm (LRA) that is able to deal with each identified data packet loss cause accordingly by stopping data transmission when necessary or adjusting the TCP performance parameters (CWND and RTO) when needed. LRA should take into consideration network resources optimization through: (i) Minimizing radio communication energy consumption, (ii) Minimizing total TCP's node energy consumption, and (iii) Maximizing TCP throughput.

This chapter describes the necessity of developing such a new TCP variant and its main characteristics and algorithms. Improving TCP performance within wireless ad hoc networks is the major goal of this thesis and the work done in this chapter accomplishes this objective. The chapter is organized as follows: after presenting the motivation behind our work in the following section. Then, we present an overview of the related work done within this domain. After that, we describe our proposed TCP variant and its main characteristics showing how it could enhance TCP performance within wireless ad hoc network environments. The methodology used to evaluate the performance of our proposed TCP variant and the obtained results are described after. Finally, the last section summarizes our work in this chapter and gives some perspectives.

5.2 MOTIVATION AND PROBLEM STATEMENT

Before going through our point of view concerning the proposed solution, we will state again the problem of TCP within wireless ad hoc networks. The current implementation of TCP loss recovery algorithm is a congestion oriented, as indicates the name: congestion control algorithm. This congestion orientation in TCP makes it incapable to deal with other data packet loss situations that TCP may suffer from. The problem of TCP resides in its inability to recognize the main cause of data packet loss within the network. The proposed solution must resolve this problem by finding how TCP could be more intelligent to differentiate between the most common situations of data packet losses within wireless ad hoc network environments. At the same time, TCP must be able to react accordingly by taking the most appropriate loss recovery action that optimizes both the nodes' and network resources.

In wired networks, the main cause for data packet loss is network congestion due to nodes' buffer overflow. In such case, backing off and stopping data transmission is the right reaction. In addition, reducing data transmission rate (by decreasing the congestion window size) after a loss event helps avoiding repeated congestion conditions within the network. However, in wireless ad hoc network, we have three network situations that may lead to data packet losses:

(i) High Bit Error Rate (BER) over the wireless communication channels, (ii) Link failure due to nodes' mobility or power depletion, and (iii) Network congestion due to nodes' buffer overflow.

It is obvious that, each of these loss situations requires a different TCP reaction, as detailed below. Indeed, TCP implementation reacts similarly in front of all data packet losses as if they were due to network congestion events. Hence, improving TCP in order to efficiently handle the different data packet loss situations within wireless ad hoc network is of a great interest.

Next, we explain the importance of differentiating between these data packet loss situations, and our perspectives concerning the reaction of TCP in front of each data packet loss cause. This is demonstrated through some illustrative examples that show the undesirable behavior of TCP within wireless ad hoc network environment faced with different types of packet loss.

5.2.1 IMPERFECTIONS OF WIRELESS COMMUNICATION CHANNELS

Communications between wireless ad hoc network nodes are established via wireless channels. This is considered as a scarce, unreliable communication media type. It is affected by the weather conditions, geographical obstacles along the route between source and destination, and the distance between communicating nodes. These factors have a varying impact on the reliability of data transmission over wireless communication channels. Experiencing different levels of Bit Error Rates (BER) over communication channels, results in data packet loss episodes within TCP sessions. In this case, TCP backs off and stops data transmission for a period of time equals to a Retransmission Time Out (RTO) value. Then, it resumes data transmission after decreasing its data transmission rate. This reaction leads to waste of both nodes' and network resources (network bandwidth, and node's energy), especially with repeated losses due to high BER over the wireless links. Stopping data transmission and reducing TCP transmission rate is considered as unnecessary, even an aggressive action in this case. A more optimal reaction here would be to retransmit the lost data packets and to resume data transmission with the same transmission rate as before the loss event.

5.2.2 LINK FAILURE WITHIN THE NETWORK

Losing a route between two communicating end points or an intermediate link within the route between them requires an action from the routing protocol. The ad hoc routing protocol is responsible for link loss recovery in the network by finding an alternative route towards the destination. In such a situation, TCP reacts as if it were network congestion. TCP backs off and stops data transmission for a certain time (RTO) and then resumes the communication using reduced data transmission rate. A more appropriate reaction here is to resume the data transmission immediately after link loss recovery or after finding an alternative route towards the destination. If the route recovery time is smaller than the RTO, TCP remains in idle state although that there might be an available route towards the other end point. In addition, decreasing TCP data transmission rate may not be necessary and could be a wasting of network's bandwidth. It is better here that TCP adjusts its data transmission rate and RTO timer values according to the new route conditions (i.e. the length and the load of the new discovered route represented by the new RTT delay over this new route).

From the above, we can conclude that the efficiency of TCP to handle a data loss situation is highly dependent on its capability of identifying correctly the cause of this loss, since each data loss situation requires a different appropriate loss recovery action.

Our objective is to develop an end-to-end TCP enhancement solution that includes: loss differentiation and loss recovery algorithms. Unlike feedback-based solutions, end-to-end solutions do not require any support or interaction from intermediate nodes within the network. Hence, the protocol messages overhead is lower than that in the case of feedback-based solutions. This helps to avoid wasting the network's available bandwidth as well as saving the nodes' energy. Additionally, our proposed solution will be based on RTT measures at the sender side host, which does not require synchronizing clocks at both sending and receiving ends.

5.3 RELATED WORK

In this section, we discuss the main TCP congestion control enhancements proposed in order to improve its performance within wireless and ad hoc networks. We discuss mainly the loss classification algorithms developed.

Loss classification algorithms can be categorized into two categories [59]: (i) implicit or end-to-end, and (ii) explicit loss differentiation algorithms. Unlike implicit loss differentiation algorithms, explicit solutions use agents that are deployed on the network's intermediate nodes. End-to-end or implicit solutions could involve the sender side only (e.g. TCP Westwood) or both the sender and receiver sides (e.g. the 3 Duplicate Acknowledgements sent by the receiver to notify the sender of the loss).

5.3.1 IMPLICIT LOSS CLASSIFICATION ALGORITHMS

TCP Westwood is an example of implicit loss classification algorithms. As described before, TCP Westwood [60] [11] [61] is a sender-side modification of TCP New Reno that estimates the connection bandwidth based on the rate of the received acknowledgements. TCP Westwood uses the estimated bandwidth to adjust and set its congestion window and Slow-Start threshold parameters. This is in contrast to traditional TCP congestion control implementation, where both congestion window size and Slow-Start threshold are halved whenever a data packet loss is detected within the connection [6]. This bandwidth estimation algorithm enhances the performance of TCP Westwood in front of random, sporadic data packet losses (wireless link errors).

In [50] the authors illustrate that, in the link failure case, both TCP New-Reno and TCP Westwood recognize the packet loss with RTO expiration. Thus, both react the same way by backing off for a while entering Slow-Start phase. In the link failure case, the average goodput of TCP Westwood 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 causes, TCP Westwood relies on the received acknowledgments. In a situation where there are several acknowledgments lost, this may lead to a wrong estimation of the end-to-end bandwidth and consequently to a TCP Westwood misbehavior. The authors found that TCP Westwood has higher energy consumption per received bit than TCP New-Reno in most cases. It can also be noticed that TCP Westwood energy consumption gets worse when BER increase. Its dependence on RTT measurements to calculate the estimated bandwidth is also responsible of this effect. Similarly to the link failure case, as BER increases over the wireless channels, the returned ACKs become prone to loss and corruption. These lost or corrupted ACKs can yield to mistaken estimated bandwidth calculations.

5.3.2 EXPLICIT LOSS CLASSIFICATION ALGORITHMS

Explicit loss identification can be performed through different estimation techniques. Many researches attempted to classify data packet losses within the network using Loss Differentiation Algorithms (LDA). The main LDA schemes are discussed in the following.

In [62] a sender-side method of end-to-end loss differentiation and adaptive segmentation (Robin) is proposed, for enhancing TCP performance in heterogeneous[4] networks. This loss differentiation algorithm enables the TCP sender to distinguish congestion losses from wireless error losses. Moreover, in order to improve the error recovery phase during a non-congestive period, an adaptive segmentation algorithm is proposed. This algorithm enables the TCP sender, if a non-congestive packet loss is detected, to retransmit smaller packets, having aggregate payload equal to the payload of the lost packet. Decreasing segment size reduces Packet Error Rate (PER) [63]. The evaluation results of this algorithm show that, in the case of high

[4] Heterogeneous networks mean a mixed wired/wireless environment.

propagation delays over the network, the improvement is negligible. We have to note here that the proposed solution assumes that only the last hop is a wireless link. While in the case of wireless ad hoc networks, all the communication links are wireless channels and the propagation delay will be higher than a simple one hop wireless network.

In [64] the authors state that the loss differentiation is often performed at the receiver side and the congestion control at the sender side. Thus, they propose a simple checksum based approach for loss differentiation together with two loss notification schemes. The two proposed loss notification mechanisms based on a TCP option for the first one and based on sending three duplicate acknowledgements (3-dupack) for the second one. The TCP option explicitly signals non-congestion losses to the sender whereas the 3-dupack scheme instead implicitly influences the sender to retransmit faster. This solution is beyond the scope of our suggestions as we do not want to apply notifications mechanism. Indeed, applying notification mechanisms will increase the overhead of TCP execution (throughput and energy consumption).

In [65] the authors propose a cross-layer solution based on two LDA algorithms in order to classify the loss origin on an 802.11 link and react accordingly. The first LDA scheme, acting at the MAC layer, allows differentiating losses due to signal failure caused by displacement or by noise from other loss types. In this case, it adapts the behavior of the MAC layer to avoid a costly end-to-end TCP resolution. The objective of the second LDA scheme, which acts at the TCP layer, is to distinguish losses due to interferences from those due to congestions and to adapt, the TCP behavior accordingly. The work done here is for wireless networks with only the last hop as wireless communication channel.

In [66] the authors tried to augment the basic LIMD (Linear Increase/Multiplicative Decrease) [67] congestion control with additional mechanisms to predict the cause of packet losses and react accordingly. They present the LIMD/H algorithm, which has the following features:

(i) LIMD/H uses the "history" of packet losses and the evolution of transmission rate for a connection in order to distinguish between congestion-induced and non-congestion induced packet losses, and

(ii) LIMD/H reacts gently to non-congestion-induced losses and aggressively to congestion induced losses, thereby achieving high efficiency, fairness, as well as quick reaction to the onset of congestion.

Biaz and Vaidya [68] looked at two different end-to-end loss differentiation approaches for TCP connections. They first looked at a set of "loss predictors" based upon three different analytic approaches to congestion avoidance that explicitly model connection throughput and/or round-trip time (e.g. TCP Vegas). The results show that, these algorithms perform badly in case of wireless losses. They could not correctly classify errors due to wireless problems. In subsequent work [69], they proposed a new algorithm that uses packet inter-arrival time at the receiver-side to differentiate losses. It assumes that the last hop is a wireless connection and that is the bottleneck. If the time between received packets is close to minimum, then a lost packet in-between is assumed to be lost due to wireless channel errors and not network congestion (buffers overflow). Using simulations, they show that it works very well in a network where the last hop is wireless and is the bottleneck link. Once again, such a solution would be inadequate for wireless ad hoc network environments where all the communication links are wireless channels and where more complex loss situations may happen.

The Spike Scheme [70], at the receiver side, measures one-way delays. The receiver switches between congestion state and wireless state according to a certain threshold. If the delay exceeds this threshold, it is a congestion state. Otherwise, it is a wireless state. The ZigZag Scheme presented in [71], extends the Spike scheme to include both the mean and standard deviation values of the measured one-way delays as well as the number of packet losses when computing the delay threshold used. According to this calculation, the higher the number of packet losses, the greater the threshold beyond which a congestion state is assumed to be. In other words, the wireless errors state becomes more likely the cause of data packet losses over the network.

Samaraweera proposes a Non-Congestion Packet Loss Detection (NCPLD) algorithm [72] in order to differentiate between congestion and non-congestion data packet loss causes within the wireless network. NCPLD is based on the “Knee Point” concept of the throughput-load graph at which the network reaches its optimum performance. Before the knee point, no congestion is present. Then, increasing data transmission rate increases the throughput while the round trip delay remains relatively constant. On the other hand, after the knee point, queuing delay at the routers results in round trip delay increase. If the current or measured round trip delay is less than the delay threshold at the knee point, the packet loss is assumed to be due to wireless errors. Otherwise, it is assumed to be a network congestion case within the network.

From the above, we can see that most of the work done in this domain addresses the problems of TCP within wireless infrastructure networks. All these contributions assume that the wireless channel connection is only the last network’s hop. However, in wireless multi-hop ad hoc networks, all the communication channels are wireless links. In addition, the proposed LDA that addresses the link failure problem is implemented at the MAC layer level and not at the Transport layer. The cross-layer solutions will naturally increase the nodes’ computations and energy consumption.

We will see in the next section that a link failure case within a wireless ad hoc network requires a specific reaction from TCP in order to recover from data packet losses. Link failure situation within such networks introduces burst data packet losses over the TCP connection. Although that burst losses could be the result of a network congestion event, the reaction of TCP in front of link failure data losses assuming that it is due to network congestion is an aggressive, inefficient reaction.

Within wireless ad hoc network environments, we have the effect of wireless communication channels (e.g. fading, multi-path routing, interference), the effect of ad hoc network environments (mainly link failure due to mobility or battery depletion) in addition to the network congestion effects due to buffers overflow. Hence, we have three different reasons (not only two as discussed in previous researches) to lose data packets within wireless ad hoc network environments. Therefore, we propose new TCP loss differentiation and loss recovery algorithms that can distinguish among the three loss situations mentioned above, within wireless ad hoc network environments.

5.4 TCP-WELCOME: TCP VARIANT FOR WIRELESS ENVIRONMENT, LINK LOSSES, AND CONGESTION PACKET LOSS MODELS

In order to enhance the performance of TCP in front of the different data packet loss situations within wireless ad hoc networks, TCP must be able to react differently when confronted with each of them [Figure 5.1]. TCP reaction should be as follows:

In the case of high BER over the wireless communication channels within the network, it is unnecessary to stop data transmission or to decrease TCP’s transmission rate after experiencing a loss event.

In case of link failure within the network (i.e. a broken route between the connection's end points), it will be sufficient to stop data transmission till an alternative route towards the destination is found. The transmission rate here will be adjusted according to the available bandwidth of the new route. It is obvious that, the length and the load of the communication path have an impact on the Round Trip delay Time (RTT) between the end points. Hence, it would be necessary, in this case, to recalculate both the TCP data transmission rate (CWND) and the Retransmission Time Out (RTO) values according to the characteristics of the new route (e.g. length and load of the new route).

When there is a congestion situation in the network, TCP keeps its normal behavior. It reacts according to how the congestion had been detected (3 Duplicate Acknowledgements or RTO). In all congestion cases, TCP stops data transmission during a certain period of time and resumes it after wards with a reduced data transmission rate.

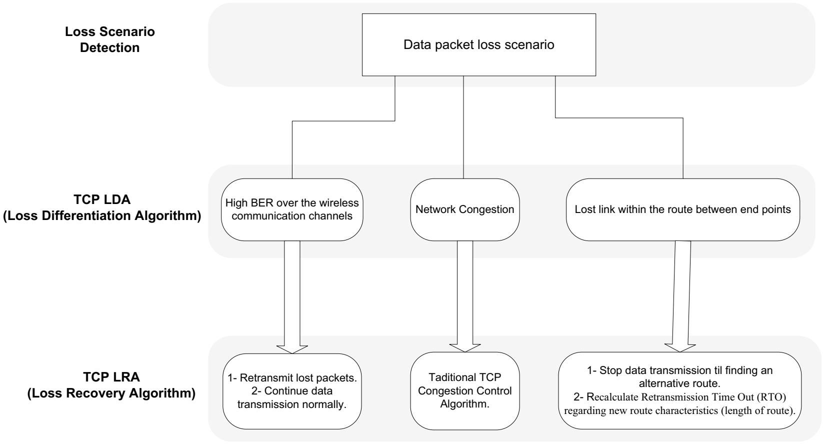

Loss Scenario Detection
Data packet loss scenario
TCP LDA (Loss Differentiation Algorithm)
High BER over the wireless communication channels
Network Congestion
Lost link within the route between end points
TCP LRA (Loss Recovery Algorithm)
1- Retransmit lost packets. 2- Continue data transmission normally.
Taditional TCP Congestion Control Algorithm.
1- Stop data transmission til finding an alternative route.
2- Recalculate Retransmission Time Out (RTO) regarding new route characteristics (length of route).

FIGURE 5.1. TCP PROPOSED LDA AND LRA ALGORITHMS

This section is organized as follows: we start by describing the rules that define how TCP-WELCOME will distinguish among the data packet loss causes defined above. Then, we present the main algorithms of TCP-WELCOME: the TCP Loss Differentiation Algorithm that is used to classify the data packet loss cause and the Loss recovery Algorithm used to recover from each type of loss.

5.5 TCP-WELCOME: LOSS DIFFERENTIATION ALGORITHM (LDA) RULES

With respect to all the concerns and suggestions above, we need to have an adapted LDA algorithm that enables TCP to correctly classify the cause of data packet losses within wireless ad hoc network environments. This algorithm should differentiate between the predefined data loss situations above.

In order to decrease the execution overhead of TCP algorithms and the interaction with the intermediate nodes within the network, our LDA and LRA algorithms are end-to-end, sender side modifications. In addition, TCP-WELCOME relies on the evolution of RTT samples of sent packets at the sender side in order to take its decisions. The evolution of RTT samples history is an efficient indication of the loss cause over the connection. We will see later how RTT samples evolution can be used to classify different data packet loss causes.

In order to detect the packet loss cause, let $q_{d,i}(t)$, $P_{d,i}(t)$ and $p_{d,i}(t)$ be the queuing, processing and propagation delays, respectively, of node i at a given time t.

Figure 5.2 illustrates the delays experienced by a TCP sent data packet over the connection until receiving the acknowledgements at the sender side.

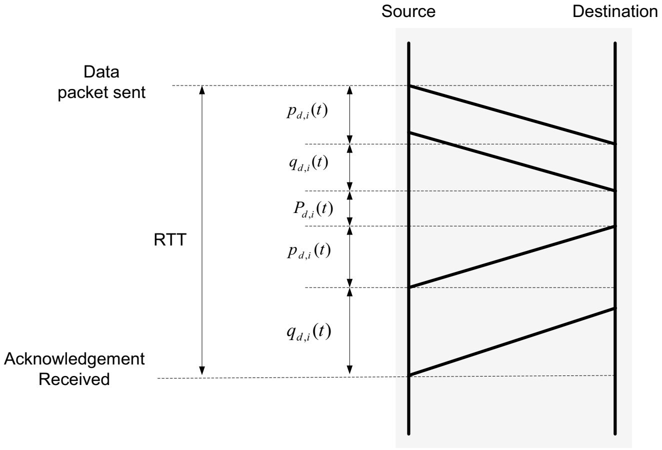


FIGURE 5.2. ROUND TRIP TIME (RTT) OF A TCP SENT DATA PACKET

Thus, RTT value of TCP connection over a route that contains m hops at time t is calculated as follows:

$$RTT(t) = 2\sum_{i=1}^{m}(q_{d,i}(t) + p_{d,i}(t) + P_{d,i}(t)) \tag{5.1}$$

The above equation describes the RTT value of a sent packet over network links towards the destination host. Henceforth, we will consider only the propagation and queuing delays as these values are highly affected by network changes. Whereas, processing delay depends solely on the communication node capabilities and not on network conditions. It is obvious that, when there is a link failure within the network, the propagation delay will change according to the new recovered route. Moreover, with network congestion, the queuing delay will increase. The next section describes the proposed Loss Differentiation Algorithm (LDA) for TCP enhancements which include the proposed RTT calculation and estimation algorithms to be used. As for how should TCP adjust its parameters (RTO, CWND, and SSThreshold) according to the packet loss cause identified by LDA, this will be discussed later.

5.6 TCP-WELCOME: LOSS DIFFERENTIATION ALGORITHM (LDA)

In this section, we identify the basic concept of our proposed LDA used to classify the different data packet loss causes over the TCP connection. The main idea is based on observing the history of RTT samples evolution within the network and the way in which TCP identifies the data packet loss [Figure 5.3].

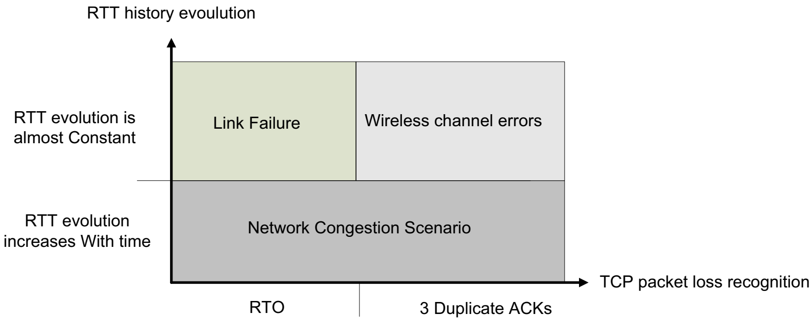

RTT history evoulution
RTT evolution is almost Constant
RTT evolution increases With time
Link Failure
Wireless channel errors
Network Congestion Scenario
TCP packet loss recognition
RTO
3 Duplicate ACKs

FIGURE 5.3. TCP PROPOSED LOSS DIFFERENTIATION ALGORITHM

According to the evolution of RTT samples over the connection, TCP should be able to identify the cause of data packet losses. Next, we will discuss how TCP can use RTT values as an indication of each type of data packet loss.

5.6.1 WIRELESS CHANNEL RELATED LOSSES

If the evolution of *RTT* samples over the connection is not highly fluctuating, staying around an average value and the data packet loss is identified through three duplicate acknowledgements, thus the data packet loss is due to wireless channel inefficiency on one of the links over the communication path. In wireless channel BER case, both queuing and propagation delays are almost constant, and the RTT samples over the connection should not experience high fluctuations with time. Additionally, when there is a valid route between the source and the destination, despite the presence of link errors, the source can always receive acknowledgements from the destination. The corruption of acknowledgements is of a lesser probability since the acknowledgement packet size is relatively small.

Thus, in the case of wireless channel induced losses, the RTT samples will stay around an average value (± *RTT_THRESHOLD*):

$$RTT(t) = 2\sum_{i=1}^{k}(q_{d,i}(t) + p_{d,i}(t)) \approx Const. \qquad (5.2)$$

Hence, three Duplicate Acknowledgements and RTT values that are almost constant mean that data packet losses over the connection are due to wireless channel errors.

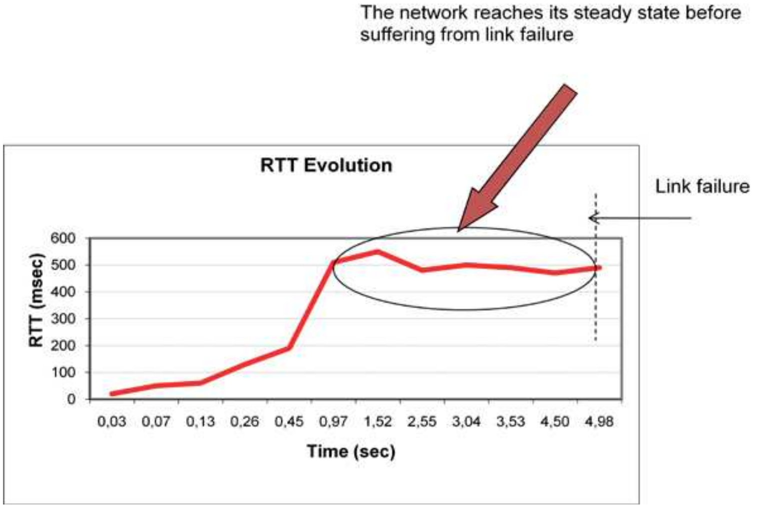
5.6.2 LINK FAILURE LOSS EVENT

If the evolution of *RTT* samples over the TCP connection is relatively constant and TCP recognizes data packet losses through Retransmission Time-Out (RTO) expiration, then data packet loss is due to a link failure situation along the route towards the destination.

In this case, after link loss recovery, the following observations could be noticed: both Propagation and Queuing delays change suddenly since the new discovered route might (i) not be having the same length (i.e. number of hops) as the lost route, (ii) be more charged than the lost one.

If the new recovered route between the source and destination does not have the same number of hops as the old lost one, then TCP connection will experience a sudden change in propagation and queuing delays. Let k be the number of hops along the new recovered route. Then, RTT values of the new route can be calculated as follows:

$$RTT(t) = 2\sum_{i=1}^{k}(q_{d,i}(t) + p_{d,i}(t)) \tag{5.3}$$


The network reaches its steady state before suffering from link failure

Link failure

FIGURE 5.4. RTT VARIATIONS WITHIN WIRELESS AD HOC NETWORK (LINK FAILURE)

In the above simulated link failure situation [Figure 5.4], we can see that RTT evolution, after a certain time of the simulation's onset (during which the ad hoc routing protocol finds a route towards the destination), the connection enters in a steady state phase where the RTT evolution stays almost constant. The Figure shows that before the link failure event, the RTT evolution fluctuation is not high and can be considered within an average value.

Hence, Retransmission Time Out and RTT values that are almost constant mean that data packet losses over the connection are due to link failure.

Regarding the time required by the ad hoc routing protocol to recover from the link failure or to find an alternative one, we can see two situations:

1. This time is less than TCP's RTO. In this case TCP identifies data packet loss through duplicate ACKs. When the TCP sender side checks the evolution of RTT samples over the connection, it finds that it is relatively constant. Accordingly, TCP will classify this type of loss as a wireless channel BER, and will verify the CWND and modifies it to be relevant to the Slow-Start threshold (as will be seen below). This action will be less aggressive when compared to traditional TCP in which it is assumed that the losses are due to network congestion.

2. If this time is longer than TCP's RTO. In this case, the TCP sender identifies data packet losses through RTO. When the TCP sender checks the evolution of RTT samples and finds that the evolution is almost constant, TCP will classify the packet loss as a link failure related loss and reacts accordingly.

5.6.3 NETWORK CONGESTION EVENT

If the evolution of *RTT* samples at the sender side is increasing gradually. Then, the loss is due to network congestion. Regardless of how TCP recognizes the data packet losses: 3 Duplicate

Acknowledgments, or RTO expiration. In this case, the queuing delay increases gradually since the network nodes' buffers are filled with time.

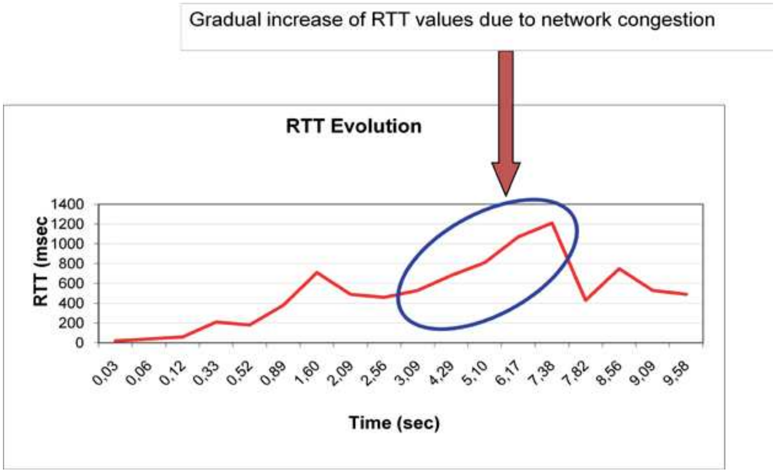


FIGURE 5.5. RTT VARIATIONS WITHIN WIRELESS AD HOC NETWORK (NETWORK CONGESTION)

It is clear from Figure 5.5, that before losing data packets due to a network congestion episode within the network, the evolution of RTT values over the TCP connection increases gradually. Thus, this gradual evolution is the indication of an imminent network congestion episode within the network.

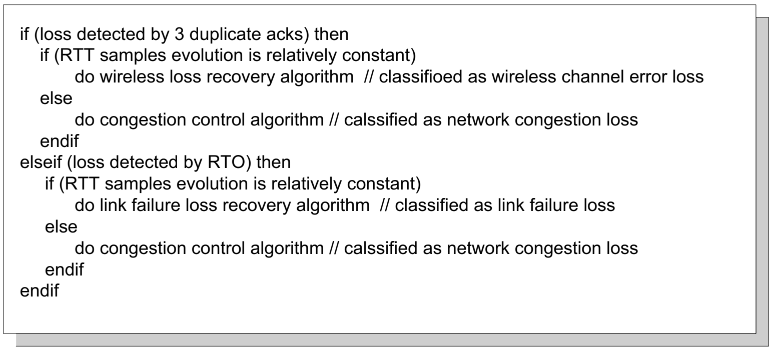
If the route between the source and the destination is constant (i.e. length of route is constant), the propagation delay stays almost constant over the connection. Let $q_i(t)$ be the queue length at node i; and c_i the link capacity at time t_i . The RTT value at time t can be calculated as follows:

$$RTT(t) = 2\sum_{i=1}^{m} \frac{q_i(t_i)}{c_i} + p_{d,i}(t) \qquad (5.4)$$

Hence, Retransmission Time Out and RTT values that are increasing gradually mean that data packet losses over the connection are due to network congestion.

5.6.4 LOSS DIFFERENTIATION ALGORITHM SUMMARY

As seen above, each cause of data packet loss can be characterized by its distinct RTT samples evolution history. Consequently, RTT estimation algorithm should be implemented to help TCP differentiate between the different data packet loss causes.



```
if (loss detected by 3 duplicate acks) then
   if (RTT samples evolution is relatively constant)
        do wireless loss recovery algorithm  // classifioed as wireless channel error loss
   else
        do congestion control algorithm // calssified as network congestion loss
   endif
elseif (loss detected by RTO) then
    if (RTT samples evolution is relatively constant)
        do link failure loss recovery algorithm  // classified as link failure loss
    else
        do congestion control algorithm // calssified as network congestion loss
    endif
endif
```

FIGURE 5.6. PSEUDO CODE OF TCP PROPOSED LDA RULES

From the Pseudo code presented in Figure 5.6, we can see clearly how TCP can differentiate between the different data packet loss causes over the connection. If TCP identifies the packet loss through three duplicate acknowledgements, it checks the evolution of RTT samples over the connection. If the RTT samples evolution is almost constant, then the data packet loss is due to wireless channel errors. Otherwise, if the RTT samples evolution increases gradually, the data packet loss is due to network congestion. Similarly, when TCP recognizes data packet loss through RTO, if the RTT samples evolution is increasing gradually, then the packet loss is due to network congestion. Otherwise (RTT samples evolution does not fluctuate much before the loss episode), the packet loss is due to link failure within the path between the source and the destination.

5.6.5 RTT CALCULATION ALGORITHM

Since our proposed solution will be based on RTT samples evolution history, it is important to be sure that the RTT samples over the connection are accurately measured. Thus, we must decide how the RTT samples will be calculated. There are many proposed algorithms in the literature describing different mechanisms for measuring RTT samples [73]. Among them:

1- Measuring from the first transmission [73].
2- Measuring from the most recent transmission [73].
3- Ignoring round-trip times for packets that have been retransmitted [73].
4- Karn’s algorithm [73].

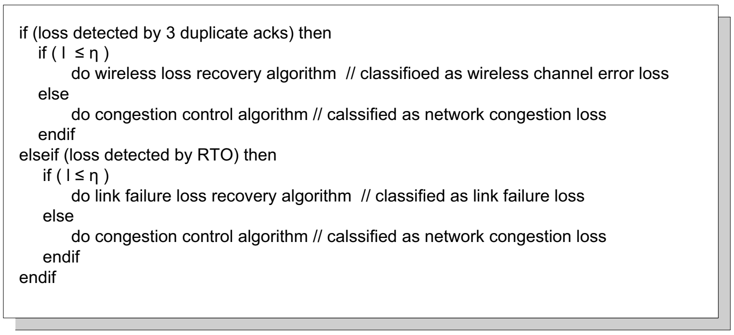
Karn’s algorithm accepts only good samples and uses the retransmission back-off strategy to ensure that good samples will eventually be available even if round-trip times increase dramatically [73]. The main idea of Karn’s algorithm is to use RTO in order to obtain accurate RTT measurements that are not affected by retransmission ambiguity. This algorithm does not take into consideration the acknowledgements of retransmitted data packets. Only the data packets that are acknowledged without retransmissions will be considered in RTO calculations. This action ensures that only accurate RTT measurements will be taken and used. Since Karn’s algorithm is recognized to be the best performing option [73], we decide to implement it in our proposed solution.

5.6.6 RTT ESTIMATION ALGORITHM

The RTT measurements have high-frequency characteristics that are desirable to detect. To be able to follow step changes in the RTT mean value due to increased network load, new competing traffic flows, or sudden path changes, more advanced algorithms for RTT estimation are needed. Currently most TCP versions implement the first-order linear filter. In mobile ad hoc networks, network parameters estimation is difficult because network observations are noisy. Current RTT estimator in TCP uses only one exponentially-weighted moving average (EWMA) static filter [6]. When a new observation is available, the EWMA filter produces a new estimate using linear combination of the old estimate plus the new observation, each given some weight. In traditional EWMA filters, the gain that determines the proportional weight assigned to the new observation and the old estimate is fixed. When old estimates are given more weight, the filter provides good stability; it resists noise in individual observations. However, when new observations are given more weight, the filter provides good agility; it is able to detect performance changes quickly. These filters are either able to detect true changes quickly or to mask observed noise and transients, but cannot do both at the same time. Ideally, one would like to have a filter that is agile when possible but stable when necessary, depending on current circumstances. Therefore, filters should be adaptive. In [74], an adaptive Flip Flop filter is proposed. The Flip Flop filter uses two EWMA filters, one is agile with a gain of 0.1, and the other is stable with a gain of 0.9. A controller selects between the two. The underlying principle of this controller is to employ the agile filter when possible, but to fall back to the stable filter when observations are noisy (RTT samples vary drastically and become noisy).

As previously discussed, our proposed LDA is based on the history of RTT variation over a TCP connection. If the network experiences a congestion condition, the variation of observed RTT samples will be noticeable. Otherwise, with wireless channel errors the variation of RTT samples will be relatively constant. We use a Flip Flop filter, and define an upper control limit, η *or RTT_G_THRESHOLD excess value*. Then, RTT samples that exceed the control limit, *l or RTT_G_COUNT_THRESHOLD*, are used as an indication of network congestion case within the network. In [59], the authors consider that much delayed packets (whose RTT exceeds the control limit) as "outliers"η.

Figure 5.7 shows the modified pseudo code of our proposed solution using the Flip Flop filter. In our proposed algorithm, we will keep η at a fixed value[5]. Furthermore, it is proved that Flip Flop filter fairness is competitive to regular TCP and its overhead is lower than that of TCP Westwood. Lowering overhead is an important issue for battery-operated devices.



```
if (loss detected by 3 duplicate acks) then
   if ( l  ≤ η )
        do wireless loss recovery algorithm  // classifioed as wireless channel error loss
   else
        do congestion control algorithm // calssified as network congestion loss
   endif
elseif (loss detected by RTO) then
    if ( l ≤ η )
        do link failure loss recovery algorithm  // classified as link failure loss
    else
        do congestion control algorithm // calssified as network congestion loss
    endif
endif
```

FIGURE 5.7 PSEUDO CODE OF TCPPROPOSED LDA

[5] This value will be defined empirically through simulations to find the most appropriate parameters.

5.7 TCP-WELCOME: LOSS RECOVERY ALGORITHM (LRA)

After identifying the data packet loss cause through using the proposed LDA, TCP-WELCOME should react accordingly using the most appropriate actions.

In the following sections, we explain how the TCP connection parameters (CWND and RTO) are adjusted after each data packet loss event over the connection. Before that, let us explain why we are interested in these two parameters. Indeed, the way in which TCP adapts its data transmission rate (CWND) has a direct impact on its performance in terms of throughput and energy consumption. More transmitted data packets and less retransmitted data packets over the connection leads to better exploitation of the available bandwidth. Also, more retransmitted data packets leads to higher energy consumption at the TCP node. In addition, the time that TCP waits after loss event and before resuming the communication session (RTO) has a severe impact on these performance parameters of the connection. For example, if the RTO is less than the RTT value over the connection, unnecessary data packet retransmissions will lead to TCP performance degradation in terms of the throughput and the energy consumption. Hence, efficient adaptation of both RTO and CWND over the connection is crucial to improving TCP performance.

5.7.1 TCP-WELCOME: RTO ADAPTATION ALGORITHM

RTO estimation differs from RTT estimation in three ways. First, the goal is not to accurately estimate the truly maximal possible RTT, but rather a good compromise that balances avoiding unnecessary retransmission timeouts due to low RTO value, versus being slow to detect that a retransmission is necessary. Second, the TCP sender needs to estimate the round-trip time of data packets, the time taken from the sender to the receiver plus the time required at the receiver side to generate an ACK. For example, a receiver employing the delayed acknowledgment algorithm may wait up to 500msec before transmitting an ACK. Thus, estimating a good value for the retransmission timer not only involves estimating a property of the network path, but also a property of the remote connection peer. Third, if loss is due to congestion, it might be necessary that the sender waits longer than the maximum RTT time, in order to give the congestion more time to diffuse from the network. If the sender retransmits as soon as the RTT time elapses, the retransmission could also be lost, whereas sending it later would be successful [75]. It has long been recognized that the setting of the retransmission timer cannot be fixed but needs to reflect the network path in use, and it generally requires dynamic adaptation because of the great extent to which RTTs could vary over the connection [76] [77].

TCP-WELCOME should adjust the RTO value according to the loss cause identified after each loss episode within the network. In the case of wireless channel errors, no RTO estimation will be done. Whereas, when there is a link failure case within the network, RTO value will be modified based on the length and the load of the new route recovered by the routing protocol. In this case, the estimation algorithm will depend on the new RTT value over the new recovered route. It is obvious that the number of hops within the route between the source and the destination of the TCP connection as well as the load of each link/node along this route affect the RTT value over that connection. The larger is the number of hops and/or the higher the load of the links/nodes, the longer the queuing delay will be experienced over the connection. The RTO value should be adapted in order to match the characteristics of the new route. A simple manner to do so is as follows:

Let RTT_{old} be the delay over the lost old route, and RTT_{new} be the delay over the new recovered route. Then, the new RTO could be calculated as follows:

$$RTO_{new} = \left(\frac{RTT_{new}}{RTT_{old}} \right) RTO_{old} \tag{5.5}$$

However, when there is congestion within the network, the RTO evolution stays the same as in traditional TCP New Reno.

5.7.2 TCP-WELCOME: TCP DATA TRANSMISSION RATE ADAPTATION

Bandwidth estimation algorithm is needed by TCP in order to well adjust its data transmission rate. Determining the available bandwidth of a new connection is a big issue in TCP. Clearly, if a transport protocol sender knows the available bandwidth, it would like to immediately begin sending data at that rate. But in the absence of knowledge about the available bandwidth, TCP must estimate it. In TCP, this estimation is currently made by exponentially increasing the sending rate until experiencing packet losses. The loss is taken as an implicit signal that the rate had grown too big, so the rate is effectively halved and the connection continues in a more conservative fashion [75]. TCP bandwidth estimation algorithm can be a sender-side or a receiver-side estimation algorithm [75]. However, our proposed solution will be based on the sender-side bandwidth estimation algorithm, in order to eliminate the impact of interaction between intermediate nodes over the connection on TCP performance.

Estimating TCP data transmission rate is dependent on the networks' links capacity and the queuing or buffering conditions within the network's nodes. Later, we will explain how the proposed TCP-WELCOME Loss Recovery Algorithm (LRA) should adjust its data transmission rate according to the data loss event (identified by LDA) within the network.

In the case of wireless channel related losses, there will be no modification of data transmission rate in TCP-WELCOME. However, in case of link failure along the route between the source and the destination, TCP-WELCOME should adjust its data transmission rate according to the characteristics of the recovered link. In this case, we have some ideas about how TCP can change the connection data transmission rate:

1. TCP-WELCOME can keep the actual data transmission rate as before the loss event, and we let TCP-WELCOME adjust it according to its congestion control algorithm, if necessary.

2. Second solution, TCP-WELCOME might decrease its data transmission rate automatically after the data loss episode over the connection. We may propose to half the actual data transmission rate before loss. This could be considered as a conservative action of TCP-WELCOME. In this way, we avoid having repeated congestion events over the links, in case the new route is more charged than the lost one. In this case, the CWND will be calculated as follows:

$$CWND_{new} = \frac{CWND_{old}}{2} \tag{5.6}$$

Where $CWND_{new}$ is the adjusted TCP-WELCOME data transmission rate after the data loss episode, and $CWND_{old}$ is the actual data transmission rate before.

At the same time the new Slow-Start threshold of TCP will be calculated as follows:

$$ssThreshold = Bw_{estimated} * RTT_{\min} \tag{5.7}$$

3- Third solution is to adjust TCP-WELCOME data transmission rate according to the proportion of the new RTT value over the new recovered route to that over the lost one. In this case, we follow also as in the previous solution, a conservative mechanism in order to prevent a network congestion episode over the network links. We assume that the new discovered route contains other competing data traffic. With this conservative algorithm we try to help enhancing TCP-WELCOME fairness within the network. The new values of CWND and SSThreshold will then be calculated as follows:

Let RTT_{old} be the delay over the lost old route, and RTT_{new} be the delay over the new discovered route. Thus, the new data transmission rate of TCP could be calculated as follows:

$$CWND_{new} = a.CWND_{old} \qquad \text{AND,} \qquad a = \left(\frac{RTT_{old}}{RTT_{new}} \right) \tag{5.8}$$

Where *a,* is the performance parameter modification factor.

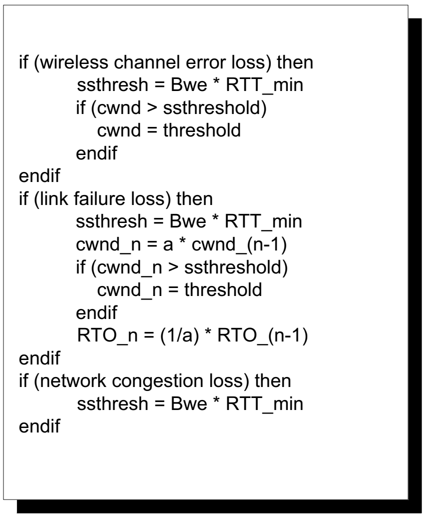
The number of the RTT samples needed by TCP in order to calculate its performance parameters *(RTO_NEW_RTT_SAMPLES)* is determined through simulations.

And the new Slow-Start threshold of TCP will be calculated as follows:

$$ssThreshold = Bw_{estimated} * RTT_{\min} \tag{5.9}$$

When there is network congestion over the TCP connection, TCP-WELCOME will keep its normal congestion control mechanism as in TCP New-Reno.

5.7.3 LOSS RECOVERY ALGORITHM SUMMARY



```
if (wireless channel error loss) then
        ssthresh = Bwe * RTT_min
        if (cwnd > ssthreshold)
           cwnd = threshold
        endif
endif
if (link failure loss) then
        ssthresh = Bwe * RTT_min
        cwnd_n = a * cwnd_(n-1)
        if (cwnd_n > ssthreshold)
           cwnd_n = threshold
        endif
        RTO_n = (1/a) * RTO_(n-1)
endif
if (network congestion loss) then
        ssthresh = Bwe * RTT_min
endif
```

FIGURE 5.8 PSEUDO CODE OF TCP PROPOSED LRA

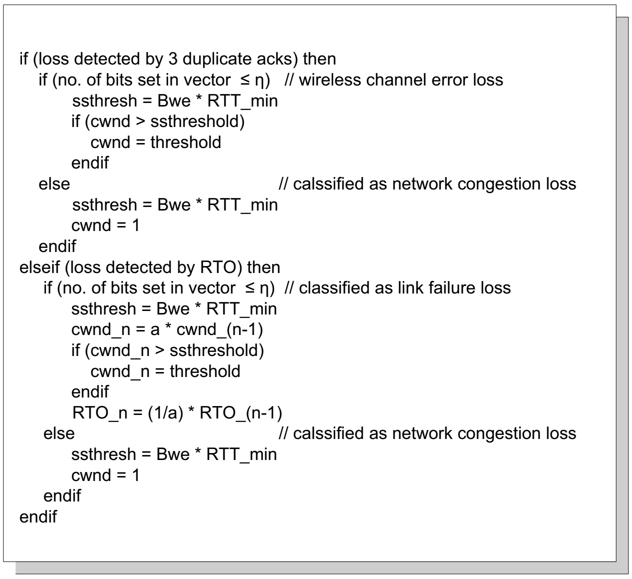
Figure 5.8, summarizes the behavior of TCP-WELCOME Loss Recovery Algorithm (LRA) in front of the different data packet loss cases within a wireless mobile ad hoc network environment. In case of wireless channel errors, TCP-WELCOME will calculate the slow-start threshold and verify that the congestion window is not greater than the allowed slow-start threshold over the connection. Yet, RTO will stay unchanged. Whereas, in case of network congestion induced data loss, TCP-WELCOME will decrease its congestion window in proportion to the calculated slow-start threshold (to avoid decreasing the congestion window to minimum as in traditional TCP), in order to enhance the TCP throughput and saves the node's energy.

When there is a link failure within the route between the communicating end points, TCP-WELCOME will adjust its parameters taking into consideration the characteristics of the lost and the new discovered route (RTT values).

5.8 SUMMARY OF THE PROPOSED ENHANCEMENT ALGORITHMS

As detailed above, TCP must be able to cope with different types of data packet loss situations within wireless mobile ad hoc networks. Our aim is to propose an algorithm that helps TCP to classify correctly the cause of data packet losses over the connection, and to react properly in front of each type of data loss situations. We define three different data packet loss situations; wireless channel errors over the communication links, link failure in the route between the communicating nodes, and network congestion due to nodes' buffer overflow. Then, we introduce a new Loss Differentiation Algorithm (LDA) which aims to classify these different loss causes. Our proposed LDA takes into consideration a new data packet loss case (that has not been discussed in previous work), which is the link failure within the network. This loss event is very common in mobile ad hoc network environments and cannot be neglected. Also, TCP must react correctly and differently according to the classified (by LDA) packet loss cause. We propose a new algorithm that deals with TCP loss recovery in such cases. Our proposed Loss Recovery Algorithm (LRA) takes into consideration the characteristics (length and load) of the route between the communicating nodes when dealing with link failure induced losses within the network.

Figure 5.9 illustrates the pseudo code of the proposed LDA and LRA algorithms to enhance TCP behavior within wireless mobile ad hoc network environments.



```
if (loss detected by 3 duplicate acks) then
   if (no. of bits set in vector  ≤ η)   // wireless channel error loss
        ssthresh = Bwe * RTT_min
        if (cwnd > ssthreshold)
           cwnd = threshold
        endif
   else                                  // calssified as network congestion loss
        ssthresh = Bwe * RTT_min
        cwnd = 1
   endif
elseif (loss detected by RTO) then
    if (no. of bits set in vector  ≤ η)  // classified as link failure loss
        ssthresh = Bwe * RTT_min
        cwnd_n = a * cwnd_(n-1)
        if (cwnd_n > ssthreshold)
           cwnd_n = threshold
        endif
        RTO_n = (1/a) * RTO_(n-1)
    else                                 // calssified as network congestion loss
        ssthresh = Bwe * RTT_min
        cwnd = 1
    endif
endif
```

FIGURE 5.9. PSEUDO CODE OF TCP-WELCOME LDA AND LRA ALGORITHMS

5.9 TCP-WELCOME IMPLEMENTATION

We implemented TCP-WELCOME within Linux kernel at first. More precisely, we implemented it over Linux kernel 2.6 since it supports pluggable congestion avoidance modules [78]. Pluggable congestion avoidance modules facilitate the introduction of new TCP congestion control mechanisms within Linux kernel. Then, we used "A Linux TCP implementation for NS2" patch [79] in order to import our Linux implementation code of TCP-WELCOME into the network simulator NS-2. In this way, we had the ability to test and validate our TCP-WELCOME implementation code through NS-2 simulations and realistic test-bed configuration measurements. The results of TCP-WELCOME through both approaches are detailed in the following sections.

5.10 TCP-WELCOME VALIDATION THROUGH NETWORK SIMULATIONS-NS2

In this section, we use the TCP-WELCOME implementation for NS-2 to validate its performance. We study both Tx/Rx energy cost and throughput. In addition, we compare TCP-WELCOME results with the results of the other TCP variants studied in the previous chapter in order to demonstrate the improvement in TCP performance parameters when using TCP-WELCOME within wireless mobile ad hoc networks.

5.10.1 SIMULATION SCENARIOS

In order to have a wide range of results that help to better understand the behavior of TCP-WELCOME in front of different packet loss situations, we compared it to different TCP variants using different simulation scenarios that describe multiple data packet loss cases. We use the same simulation scenarios that are described in the previous chapter: (i) network congestion, (ii) data packets interference, (iii) link failure, and (iv) signal loss. We build our scenarios using Ad-hoc On-demand Distance Vector (AODV) [52] as an ad hoc routing protocol. AODV is a reactive ad hoc routing protocol that triggers only route discovery process when the source has data to be transmitted toward the destination. This leads to low routing messages overhead. The scenarios are defined to be run using NS-2 [27] as a Network Simulator tool. In our simulations, all nodes communicate through identical wireless radio settings using the standard MAC 802.11 having a bandwidth of 2Mbps and a radio propagation range of 250 meters. The results are compared to those of the four other TCP variants (TCP New Reno, TCP SACK, TCP Vegas, and TCP Westwood) studied in the previous chapter.

Table 5.1 defines the variables and default values used in our simulated TCP-WELCOME variant.

Variable	Value
RTT_THRESHOLD	10
RTT_G_THRESHOLD	5
RTT_G_COUNT_THRESHOLD	5
RTO_NEW_RTT_SAMPLES	4

TABLE 5.1. TCP-WELCOME IMPLEMENTATION VARIABLES AND VALUES

5.10.2 SIMULATION RESULTS

In this section, we discuss the results of TCP-WELCOME validation process through simulations. In order to show the performance gains of TCP-WELCOME, its performance results are compared to the performance of the other TCP variants studied in the previous chapter (TCP New Reno, TCP SACK, TCP Vegas, and TCP Westwood).

5.10.2.1 NETWORK CONGESTION SITUATION RESULTS

The results demonstrate that, in the event of network congestion losses, TCP-WELCOME has almost the same performance compared to the other variants, in terms of energy consumption (Figure 5.11). This is expected as TCP-WELCOME reacts to congestion in the same manner (as in TCP New Reno). For the average throughput, we notice in Figure 5.10 that TCP-WELCOME has a comparable performance with most studied variants (TCP New Reno, TCP SACK, and TCP Westwood). This result confirms that TCP-WELCOME is able to classify correctly the losses induced from network congestion and takes the right actions to recover from these losses.

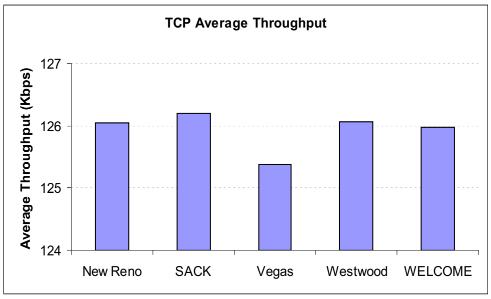


FIGURE 5.10. TCP AVERAGE THROUGHPUT

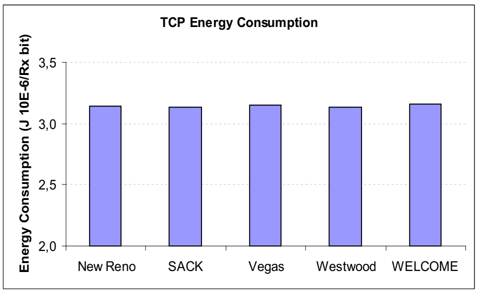


FIGURE 5.11. TCP TX/RX ENERGY CONSUMPTION

5.10.2.2 INTERFERENCE SITUATION RESULTS

Figure 5.12 and Figure 5.13, show clearly that in front of interference, TCP-WELCOME outperforms all the other variants both in terms of average throughput and energy consumption. The ability of TCP-WELCOME to classify the cause of a data packet loss, as wireless signal related problems, and not decreasing the data transmission rate improves its performance compared to other TCP variants which decrease their data transmission rate (halving data transmission rate in most cases). We notice also that TCP-WELCOME outperforms TCP Westwood, which was developed for wireless networks, and has the ability to differentiate between wireless and congestion induced losses, in both terms of throughput and energy consumption. The fact that TCP-WELCOME does not decrease its data transmission rate or modify it in the case of wireless induced losses as in TCP Westwood is the main difference between these two variants.

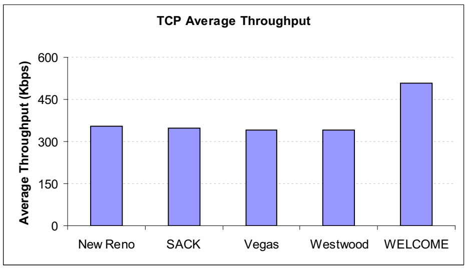


FIGURE 5.12. TCP AVERAGE THROUGHPUT

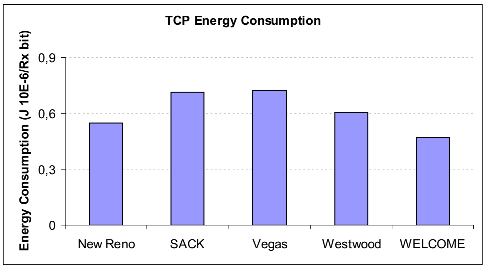


FIGURE 5.13. TCP TX/RX ENERGY CONSUMPTION

5.10.2.3 LINK FAILURE SITUATION RESULTS

In MANETs, it is obvious that the communication paths between the communicating end points can break (due to mobility or depletion of nodes' batteries). Figure 5.14 and Figure 5.15, show that the average throughput of TCP-WELCOME and its energy consumption are improved significantly compared to those of other TCP variants. The ability of TCP-WELCOME to detect that the packet losses are due to link failure and to react with the most appropriate action leads to a much better performance compared to all other TCP variants which react assuming that losses are due to congestions and decrease data transmission rate to minimum, thus, consuming more energy and leading to low throughput. In TCP-WELCOME, adjusting data transmission rate according to the new discovered route's characteristics; helps conserving node's energy and maximizing average throughput.

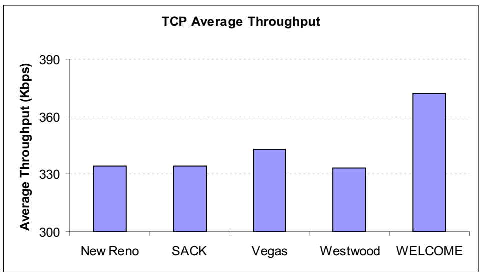


FIGURE 5.14. TCP AVERAGE THROUGHPUT

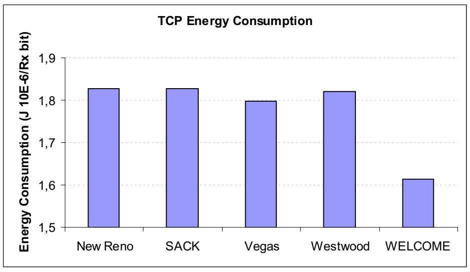


FIGURE 5.15. TCP TX/RX ENERGY CONSUMPTION

5.10.2.4 SIGNAL LOSS SITUATION RESULTS

Losing the radio signal might be considered as another reason to get disconnected from the other communicating end. In link loss, both nodes (sender and receiver) would search for another route to complete the session. While in signal loss, this is not possible. After signal loss recovery, most TCP variants' sender will start the communication session from the beginning, starting from Slow Start phase. This will be the case, each time the communicating nodes get disconnected due to absence of wireless signal. TCP-WELCOME, however, recognizes this data packet loss as link failure and reacts accordingly. While the energy consumption of most variants is almost the same (Figure 5.17); TCP-WELCOME outperforms them all in terms of average throughput (Figure 5.16). Depending on the duration of the signal loss, the packet loss is detected through RTO or through 3 duplicated acknowledgements. In both cases, TCP-WELCOME does not decrease its data transmission rate after data packet loss (as in TCP New Reno) leading to the observed throughput gain and to an optimum usage of wireless channel bandwidth resources. Although TCP Vegas has the least energy consumption among the others; its performance in terms of average throughput is mediocre.

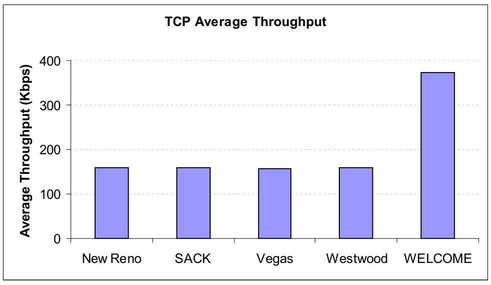


FIGURE 5.16. TCP AVERAGE THROUGHPUT

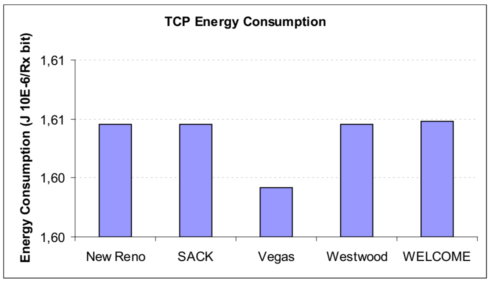


FIGURE 5.17. TCP TX/RX TX/RX ENERGY CONSUMPTION

5.11 TCP-WELCOME VALIDATION THROUGH REALISTIC TEST-BED

After validating TCP-WELCOME implementation code through NS-2, we evaluate its Linux kernel implementation through a realistic test-bed configuration. We study the computational energy consumption. The congestion control algorithm implemented in TCP involves running a number of complex mathematical operations to calculate the values of the different timers and other performance parameters. These are CPU-intensive operations that, in turn, lead to an increase in energy consumption at the TCP Node's CPU. Thus, in order to understand the effect of TCP congestion control algorithm in the case of different data packet loss cases, we need to study the TCP computational energy cost in the event of each data packet loss situation.

In this section, we use the same test-bed configuration as in the previous chapter in order to evaluate the computational energy cost of TCP-WELCOME implementation.

Figure 5.18 recalls this configuration. We use the same packet loss events used before to study the performance of the different TCP variants (congestion, interference, link loss, and signal loss). Here again, we compare our TCP-WELCOME computational energy cost to the four studied variants.

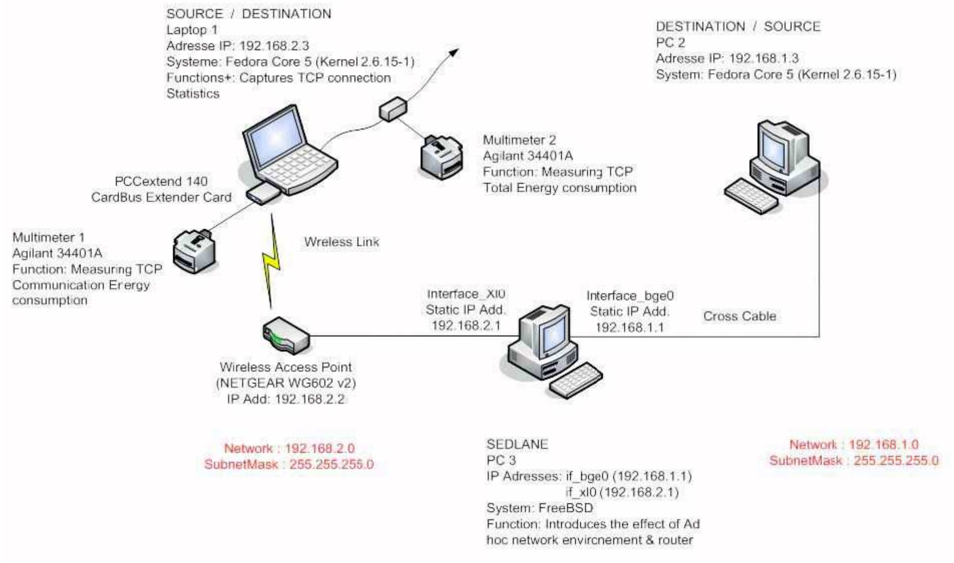


FIGURE 5.18. TCP-WELCOME ENERGY CONSUMPTION MEASUREMENTS TEST-BED

5.12 TEST-BED RESULTS (TCP-WELCOME COMPUTATIONAL ENERGY COST)

In this section, we show and analyze the results of TCP –WELCOME evaluation experiments. The results are discusses according to different data packet loss situations studied.

5.12.1 NETWORK CONGESTION SITUATION RESULTS

We can see from Figure 5.19 that TCP-WELCOME computational energy cost is slightly higher than that of TCP New Reno in the event of network congestion. This is due to the fact that TCP-WELCOME verifies the cause of data packet losses (Loss Differentiation Algorithm) before triggering the most appropriate data loss recovery action (Loss Recovery Algorithm). While in TCP New Reno, it stops data transmission without searching the cause behind data packet losses. This loss classification process in TCP-WELCOME necessitates more CPU calculations which lead to more computational energy cost. Also, we notice from the same figures that TCP-WELCOME still outperforms both TCP SACK and TCP Vegas in terms of computational energy cost. Indeed, these two variants use more complex algorithms without necessarily getting better results in terms of goodput. Similarly, recall that TCP-WELCOME sends more data than both TCP New Reno and TCP Westwood.

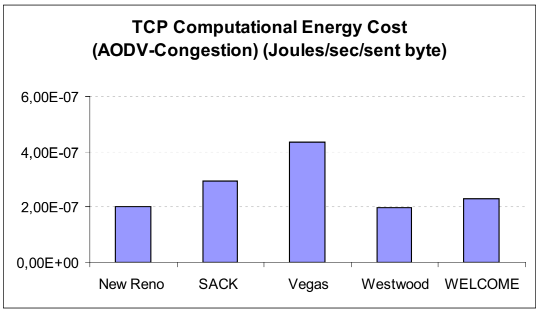


FIGURE 5.19. TCP COMPUTATIONAL ENERGY CONSUMPTION

5.12.2 INTERFERENCE SITUATION RESULTS

Figure 5.20 shows that TCP-WELCOME has almost the same performance in term of computational energy cost as TCP Westwood. In the case of data interference loss event, both TCP-WELCOME and TCP Westwood have the ability to classify data losses as due to wireless link problems. While, other TCP variants will misinterpret data packet loss and consider it as if it were due to congestion. Also, the high computational energy cost of TCP Vegas is due to the fact of high computational processing of its performance parameters at the reception of each acknowledgment.

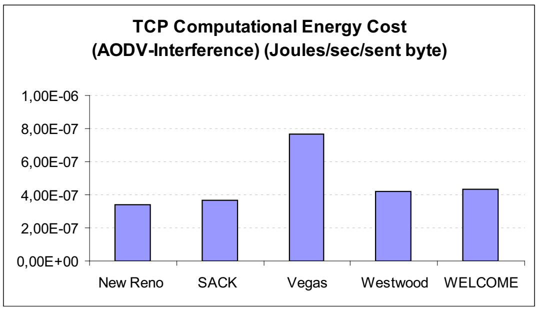


FIGURE 5.20. TCP COMPUTATIONAL ENERGY CONSUMPTION

5.12.3 LINK FAILURE AND SIGNAL LOSS SCENARIOS RESULTS

For both Link failure and Signal loss situations, Figure 5.21 and Figure 5.22, we notice the high computational energy cost of TCP-WELCOME compared to other TCP variants. Actually, as none of the other variants has the ability to classify and recognize the data packet loss cause over the connection, they all react by stopping data transmission and enters slow-start phase. On the other hand, TCP-WELCOME classifies the data packet losses cause and then reacts by calculating and adapting its performance parameters (RTO, and CWND). This leads to more computational energy cost for TCP-WELCOME.

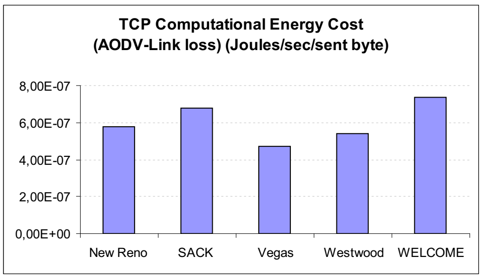


FIGURE 5.21. TCP COMPUTATIONAL ENERGY COST

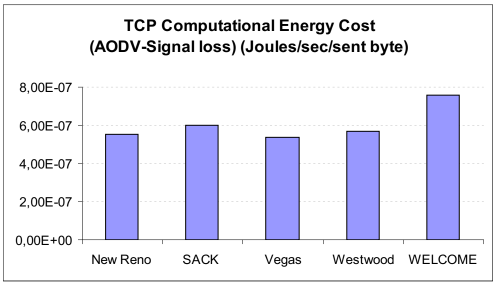


FIGURE 5.22. TCP COMPUTATIONAL ENERGY COST

5.13 COMPARISON OF TCP'S TOTAL ENERGY COST

The TCP total energy consumption contains both the communication energy consumption, at data packet transmission, reception, and forwarding, and the computational energy cost of the TCP's algorithms execution at the nodes' CPU unit. The Network Simulator (NS-2) simulation results give only the communication (Tx/Rx) energy cost, and not the computational energy cost. Hence, as a result, to obtain the computational energy cost of TCP, a realistic test-bed implementation is needed. In order to get the total energy consumption of TCP using NS-2, we integrate the computational energy cost obtained through the test-bed experiments within the energy model of NS-2 code. With this contribution, we are able to get the TCP's total energy cost through simulations using NS-2.

This section provides the final TCP-WELCOME validation results through the modified version of NS-2 that incorporates the TCP computational energy cost. We compare the performance of TCP-WELCOME with the four other studied TCP variants in terms of total energy cost (i.e. Tx/Rx energy cost + computational energy cost).

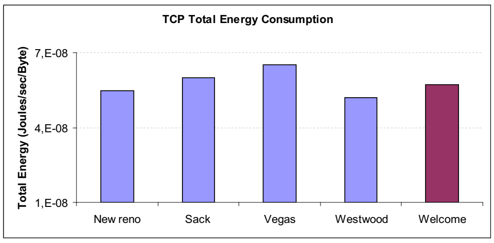


FIGURE 5.23. TCP TOTAL ENERGY CONSUMPTION (NETWORK CONGESTION)

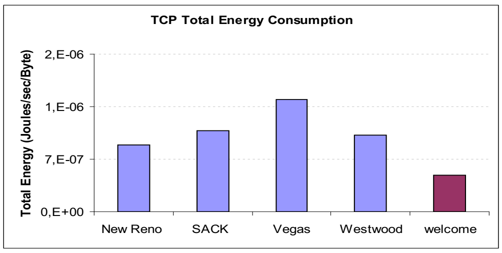


FIGURE 5.24. TCP TOTAL ENERGY CONSUMPTION (INTERFERENCE)

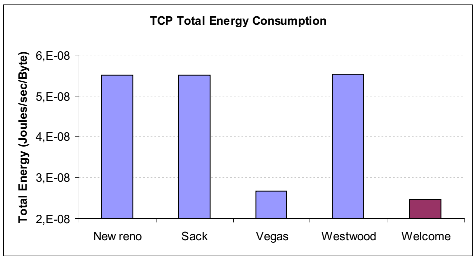


FIGURE 5.25. TCP TOTAL ENERGY CONSUMPTION (LINK FAILURE)

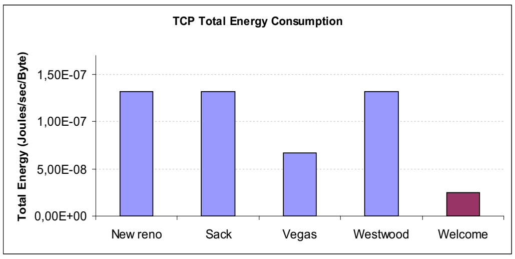


FIGURE 5.26. TCP TOTAL ENERGY CONSUMPTION (SIGNAL LOSS)

The above Figures show that TCP-WELCOME outperforms the other TCP studied variants, especially in interference, link loss, and signal loss cases (Figure 5.24, Figure 5.25 and Figure 5.26). Figure 5.23 shows that TCP-WELCOME, in network congestion case, outperforms TCP Vegas and TCP SACK, but has slightly higher total energy consumption than TCP Westwood and TCP New Reno. We have to mention that, our aim of developing TCP-WELCOME was to deal with data packet losses that can be found within wireless ad hoc networks, faced to which existent TCP variants performance is negatively affected.

The ability of TCP-WELCOME to adapt its data transmission rate and its RTO values if needed helps to better deal with data losses due to both link loss and signal loss situations within the network. Also, TCP-WELCOME does not decrease its data transmission rate after data losses due to wireless channel related problems (such as interference). This helps reducing the energy consumed to adapt the congestion window, as in the case of most other TCP variants. This as well leads to better usage of the available bandwidth.

5.14 CONCLUSION

TCP suffers from drastic degradation in performance when deployed within wireless ad hoc networks. This is due to the fact that TCP cannot differentiate between the different data packet loss situations over the connection. Misinterpreting the data packet loss cause and reacting as if it is due to congestion leads to waste of network and nodes resources (such as bandwidth and energy consumption). Hence, the ability of TCP to classify correctly the data packet loss cause over the connection helps to improve its performance within wireless ad hoc network environments.

In this chapter we introduced TCP-WELCOME, a new TCP variant that is suitable for wireless mobile ad hoc network environments. Unlike other TCP variants, it uses a Loss Differentiation Algorithm (LDA) that is able to identify accurately the three common data packet loss causes within such network: network congestion, wireless channel errors, and link losses. Moreover, TCP-WELCOME adopts a new Loss Recovery Algorithm (LRA) that reacts efficiently to each identified data packet loss cause with the most appropriate action.

In order to show the performance improvement of TCP-WELCOME we implemented it into both Linux Kernel and the Network Simulator version 2 tool (NS-2). We compared its performances to different TCP variants under different data packet loss scenarios (congestion, interference, link failure, and signal loss). This comparative study showed that both TCP average throughput and total energy consumption have been significantly improved. We also showed that TCP-WELCOME outperformed other TCP variants in most cases thanks to its ability to identify correctly the type of data packet loss through its loss differentiation algorithm and to take the most appropriate reaction to recover from data losses (loss recovery algorithm).

As a future work, we opt to enhance the decision making process of the TCP loss differentiation algorithm through using the utility functions. This enhancement intends to minimize the grey area between the different data loss event borders in order to get more accurate decisions.

CHAPTER 6. CONCLUSION AND PERSPECTIVES

6.1 CONCLUSION

Wireless ad hoc networks gained a lot of interest in the last decade due to the increase of ubiquitous communications technologies. Wireless ad hoc nodes are connected through wireless channels. These nodes are independent and battery operated, since such networks do not have a fixed infrastructure or centralized administration. Each node acts as a client, a server, and a router for the other nodes within the network. Wireless ad hoc networks inherit the characteristics of wireless networks such as wireless related problems (e.g. fading and wireless link errors). In addition, such networks have specific characteristics such as nodes' mobility and battery depletion events. Our objective in this thesis was to improve the performance of TCP within such environment.

The problem with TCP within wireless ad hoc networks results from its inability to distinguish between different data packet loss causes over the connection. TCP reacts as if all data packet losses are due to congestion. Many researches where conducted recently in order to improve the performance of TCP within wireless ad hoc networks. It was proved that the performance of TCP degrades dramatically when implemented within such environments. The main cause of this degradation is that TCP was originally developed for wired networks and is, thus, not capable to cope with data packet loss causes other than congestion. Many TCP variants were developed in order to improve the performance of TCP within wireless networks in general. These TCP variants have the ability to distinguish between non-congestion wireless related losses and congestion induced losses. We have to mention here, that within wireless ad hoc network environments, there would be data packet losses due to link failures as well. Link failure within an ad hoc environment could be the result of either mobility of nodes or networks devices' energy depletion. Although that data packet loss caused by link failures is a very common data loss cause within wireless ad hoc networks, in our knowledge no TCP variant was developed to address this cause of data packet losses. Indeed, as we showed through this thesis, the reaction in front of such losses should not be the same as in the case of congestion induced or wireless related data packet loss.

A first step towards improving TCP performances in wireless ad hoc network environments, we have to be able to realize a complete evaluation implying all the performance metrics that are of interest in such environment. These metrics are two: the throughput and the total energy consumption. This latter, in turn, is comprised of the communication energy used for transmission, reception and forwarding of data packet as well as the computational energy cost that is due to the different algorithms and storage operations involved in TCP. This second parameter is only measurable through experiments. These experiments should be able to introduce the effect of the ad hoc network environment in order to get the computational energy cost of TCP when used in such environment. So, the first contribution of this thesis was to develop SEDLANE, a new wireless ad hoc network emulator that is able to emulate wireless ad hoc networks of any scale. More precisely, SEDLANE allows reproducing the end-to-end effects of wireless ad hoc environment through the emulation of the end-to-end data packets delays and losses over the connection. This is obtained while using a low number of physical nodes. SEDLANE has many interesting design feature. First, it is easy and simple. It has the ability to use non-complex (easy to extract) characteristics of the connection (such as losses and delays) to emulate many ad hoc network configurations and parameters (such as ad hoc routing protocols, node mobility, and connection throughput). Second, it has the ability to emulate multi-hop wireless ad hoc network of any size using a small number of physical machines, and without the need for specific or expensive networking hardware. Finally, it allows performing controlled and repeatable tests. The validation results of SEDLANE show its capability to emulate ad hoc networks of any scale and that it respects accurately the ad hoc network emulated parameters over the connection.

To better understand the behavior of TCP within wireless ad hoc networks. We conducted a comparative performance study of different TCP variants (TCP New Reno, TCP SACK, TCP Vegas, and TCP Westwood). The objective of this performance evaluation was to study the effect of different TCP congestion control algorithms on the performance of TCP within wireless ad hoc networks. We studied TCP's throughput, communication energy cost and TCP's computational energy cost through simulations and a realistic test-bed configured through SEDLANE. We studied the performance of TCP while facing the most common data packet loss scenarios that can be found within wireless ad hoc networks, such as interference, link failure, signal loss, and congestion. Each data packet loss case was studied in details and the performance of each of the studied TCP variants was analyzed. Using the test-bed, we also measured, the computational energy cost of the TCP's congestion control algorithms, such as Slow-Start, Fast Retransmit/ Fast Recovery, and Congestion Avoidance. We used the obtained results to enhance the NS-2 energy model by incorporating the calculated computational energy consumption of TCP into this energy model. Hence, we are now able to get the total TCP energy consumption (both communication and computational) through NS-2 simulations.

The results obtained from the detailed comparative study of TCP variants within wireless ad hoc networks show that the inability of TCP to distinguish between different data packet loss causes over the connection is the main reason behind the performance degradation of TCP within such networks. According to the obtained results, we designed and implemented a new TCP variant for wireless ad hoc networks. The new developed variant, which we called TCP-WELCOME, comprises two main algorithms:

1. Loss Differentiation Algorithm that can differentiate between the three common data packet losses within wireless ad hoc network environment (wireless channel errors, link failures, and congestions).

2. Loss Recovery Algorithm that has the ability to recover, differently, from each of the data packet losses classified by the loss differentiation algorithm and react accordingly in such a way to optimize the performance of TCP, both in terms of throughput and energy consumption.

We evaluated the performance of TCP-WELCOME and we compared the results with the other studied variants using both simulations (through NS-2) and realistic test-bed configuration experiments (configured through SEDLANE). The results show that TCP-WELCOME is the most adapted variant for wireless ad hoc network environments. TCP-WELCOME is able to identify correctly the cause of data packet losses over the connection, and to optimize TCP performance by triggering the most suitable reaction to recover from each data packet loss. The results demonstrate that TCP-WELCOME loss recovery algorithm is able to decrease the communication (Transmission and reception) energy consumption and increase the connection throughput. This is achieved while reducing unnecessary data packet retransmissions. Also, the adaptation of TCP-WELCOME performance parameters (CWND, and RTO) in case of link failure induced losses according to the characteristics of the new recovered route leads to better usage of the available bandwidth as well as the nodes' energy resources. Hence, TCP-WELCOME optimizes both the TCP total energy consumption and the throughput compared with the other four studied TCP variants.

6.2 PERSPECTIVES

In this thesis, two of our major contributions are: the SEDLANE emulator and our new TCP variant, TCP-WELCOME, for wireless ad hoc networks. Our future work concerns both of them.

Even if it had been designed in order to allow evaluating the computational energy cost of TCP, it is clear that SEDLANE can be used to help evaluating the performances of any client-server application or protocol when used in ad hoc networks. The client-server communication model was the only possible end-to-end communication model for more than two decades. However, at the end of the nineties appeared a new communication paradigm, known as Peer-to-Peer (P2P), allowing multiple parties to interact at the application-level to perform a single communication. So, this new communication model that involves multiple parties should be considered. Evaluating the performances of the network services using such communication model over wireless ad hoc networks is not an easy task. One perspective may be to extend SEDLANE in order to facilitate such evaluation. As the P2P communication model involves multiple parties, it also involves multiple connections between peers within the wireless ad hoc network. It is clear that each of these connections can be emulated through SEDLANE. However, building a test-bed using a set of machines (representing the different peers) that should be properly interconnected, then running SEDLANE manually to configure each link may rapidly become a tremendous task. A system that allows extracting from simulations traces the different parameters necessary to automate all this process may be of great help. First, this control-board system should be able to help SEDLANE users to set up the evaluation test-bed for their application (i.e. guidelines of peers interconnection and configuration). When this is done, it should be able to easily deploy and configure the SEDLANE within the network. Obviously, these configurations are extracted from more complex simulation traces than the currently used. The design and evaluation of such a system is the target of our future work. We strongly believe that such a system, when designed and validated, would be of great help to the research community.

As our second perspective, we propose to study the potential benefit of using the utility theory in order to further optimize the performance of TCP WELCOME within wireless ad hoc networks. Indeed, the limits or borders between the data packet loss classification areas as defined in CHAPTER 5 may be less clear sometimes. So, in some extreme and rare cases some classification errors may happen. The performance of TCP WELCOME, as designed in this thesis, may then be improved in these rare and extreme cases. This improvement may be realized through the use of the utility theory in order to adapt the parameters used by the classification rules. This rules will then became patterns to identify instead of fixed rules. Utility functions will then be used in the loss differentiation algorithm to govern the switching between different recovery-algorithms. The design and evaluation of these utility functions will be the target of our future work. The discussion of the interest to go deeper in the classification process, according to the obtained percentage of improvement, will also be the target of this future study.

[1] M. Zorzi and R. Rao, "Energy Efficiency of TCP in a local wireless environment," *In Proceedings of 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," *In 4th European Personal Mobile Communications Conference*, Febreuary 2001.

[3] H. Singh and S. Singh, "Energy consumption of TCP Reno, New Reno, and SACK in multi-hop wireless networks," *In Proceedings of ACM SIGMETRICS'02*, June 2002.

[4] G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobile Ad hoc Networks," *In Proceddings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 219-230, August 1999.

[5] M. Allman, V. Paxon, and W. Stevens, "TCP Congestion Control," *RFC 2581, IETF*, April 1999.

[6] V. Jacobson, "Congestion Avoidance and Control," *ACM SIGCOMM'88 symposium on communications architectures and protocols*, vol. 18, no. 4, August 1988.

[7] V. Jacobson, "Modified TCP Congestion Avoidance Algorithm," *end2end-interest mailing list, (ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail)*, April 1990.

[8] K. Fall and S. Floyd, "Simulation-based comparison of Tahoe, Reno, and sack TCP," *ACM Computer Communications Review*, vol. 5, no. 3, July 1996.

[9] S. Floyd, T. Henderson, Gurtov, and A., "The NewReno Modification to TCP's Fast Recovery Algorithm," *RFC 3782, IETF*, April 2004.

[10] M. Michel, L.S. Nelson, and F José, "On the Performance of TCP Loss Recovery Mechanisms," *In Proceedings of IEEE International Conference on Communications*, vol. 3, pp. 1812- 1816, May 2003.

[11] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and 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.

[12] L. S. Brakmo, S. W. O'Malley, and Larry L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," *ACM SIGCOMM'94*, pp. 24-35, August 1994.

[13] S. Kopparty, S. Krishnamurthy, M. Faloutous, and S. Tripathi, "Split TCP for mobile ad hoc networks," *In Proceedings of IEEE GLOBECOM*, November 2002.

[14] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM/Baltzer Wireless Networks Journal*, vol. 1, no. 4, pp. 469-481, December 1995.

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

Networking, vol. 5, pp. 756-769, 1997.

[16] P. Sinha, Venkitaraman N., R. Sivakumar, and V. Bharghavan, "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks," *In ACM Mobicom '99*, August 1999.

[17] J. Liu and S. Singh, "ATCP: TCP for Mobile Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 7, July 2001.

[18] Ola Westin, "TCP Performance in Wireless Mobile Multi-hop Ad Hoc Networks," *SICS Technical Report T2003:24, Swedish Institute of Computer Science ISSN 1100-3154*, 2003.

[19] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based Performance Analysis of Routing Protocols for Mobile Adhoc networks," *In Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, p. 195–206, August 1999.

[20] T. Dyer and R. Boppana, "A comparison of TCP performance over three routing protocols for mobile ad hoc networks," *In Proceedings of ACM MOBIHOC*, p. 56–66, 2001.

[21] F. Wang and Y. Zhang, "Improving TCP performance over mobile ad hoc networks with out-of-order detection and response," *In Proceedings of ACM MOBIHOC*, p. 217–225, June 2002.

[22] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback based scheme for improving TCP performance in Ad-Hoc wireless networks," *In Proceedings of the Personal Communications, IEEE*, vol. 8, no. 1, pp. 34 - 39, February 2001.

[23] V. Anantharaman, S.-J. Park, K. Sundaresan, and R. Sivakumar, "TCP performance over mobile ad hoc networks: A quantitative study," *Journal of Wireless Communications and Mobile Computing*, vol. 4, no. 2, p. 203–222, March 2004.

[24] J. Monks, P. Sinha, and V. Bharghavan, "Limitations of TCP-ELFN for ad hoc networks," *In Proceedings of Mobile and Multimedia Communications*, October 2000.

[25] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," *RFC 3168, Category: Standards Track*, September 2001.

[26] V. Ramarathinam and 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.

[27] NS-2. http://www.isi.edu/nsnam/ns/.

[28] S. Floyd and F. Kevin, "Router mechanisms to support end-to-end congestion control," *Technical report*, February 1997, available at : ee.lbl.gov/nrg-papers.html.

[29] T. Ott, J. Kemperman, and M. Mathis, "Window size behavior in TCP/IP with constant loss probability," *In The Fourth IEEEWorkshop on the Architecture and Implementation of High Performance Communication Systems (HPCS97)*, June 1997.

[30] Dummynet. http://info.iet.unipi.it/luigi/ip_dummynet/.

[31] W. Jiang and C. Zhang, "A portable real-time emulator for testing multi-radio MANETs," *In Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium, IPDPS*, p. 145, 2006.

[32] J. Dike, "A User-Mode Port of the Linux Kernel," *5th Annual Linux Showcase and Conference*, vol. 5, pp. 2 - 2, 2001.

[33] M. Engel, M. Smith, S. Hanemann, and B. Freisleben, "Wireless Ad-Hoc Network Emulation Using Microkernel-Based Virtual Linux Systems," *In Proceedings of the 5th EUROSIM Congress on Modelling and Simulation, EUROSIM Publis*, pp. 198-203.

[34] Y. Zhang and W. Li, "An Integrated Environment for Testing Mobile Ad Hoc Networks," *In Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 104-111, 2002.

[35] A. Vahdat et al., "Scalability and Accuracy in a Large-Scale Network Emulator," *In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, December 2002.

[36] P. Mahadevan, A. Rodriguez, and D., Vahdat, A. Becker, "MobiNet: A Scalable Emulation Infrastructure for Ad Hoc and Wireless Networks," *Mobile Computing and Communications Review*, vol. 10, no. 2, 2004.

[37] J. P. Macker, W. Chao, and J. W. Weston, "A Low-Cost, IP-based Mobile Network Emulator (MNE)," *MILCOM 2003 - IEEE Military Communications Conference*, pp. 481-486, 2003.

[38] P. Zheng and L. Ni, "EMWIN: Emulating a Mobile Wireless Network using a Wired Network," *In Proceedings of WOWMOM*, September 2002.

[39] M. Puˇzar and T. Plagemann, "NEMAN: A network emulator for mobile ad-hoc networks," *In Proceedings of the 8th International Conference on Telecommunications, ConTEL 2005*, 2005.

[40] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," *RFC3626, IETF*, 2003.

[41] M. Pužar, J. Andersson, T. Plagemann, and Y. Roudier, "SKiMPy: A Simple Key Management Protocol for MANETs in Emergency and Rescue Opera-tions," *Technical Report, Department of Informat-ics, University of Oslo*, February 2005.

[42] M. Engel and B. Freisleben, "Wireless Ad-Hoc Network Emulation Using Microkernel-Based Virtual Linux Systems," *Proceedings of EuroSIM 2004*, p. 198–203, 2004.

[43] M. Hohmuth, "The Fiasco Kernel: Requirements Definition," *TU Dresden Technical Report TUDFI98-12*, 1998.

[44] M. Bateman, C. Allison, and A. Ruddle, "A Scenario Driven Emulator for Wireless, Fixed and Ad Hoc Networks," *In Proceedings of PGNet2003*, pp. 273-278, June 2003.

[45] J. Flynn, H. Tiwari, and D. O'Mahony, "A Real-Time Emulation System for Ad Hoc Networks," *In Proceedings of the Communication Networks and Distributed Systems Modelling and Simulation Conference*, January 2002.

[46] G. Judd and P. Steenkiste, "Using Emulation to Understand and Improve Wireless Networks and Applications," *In Proceedings of NSDI*, May 2005.

[47] TTCP. http://www.pcausa.com/Utilities/pcattcp.htm.

[48] TCPDump. http://www.ethereal.com/docs/manpages/tcpdump.8.html.

[49] TCPTrace. http://jarok.cs.ohiou.edu/software/tcptrace/.

[50] A. Seddik-Ghaleb, Y. M. Ghamri Doudane, and S.-M. Senouci, "A Performance Study of TCP variants in terms of Energy Consumption and Average Goodput within a Static Ad Hoc Environment," July 2006.

[51] A. Seddik-Ghaleb, Y. M Ghamri Doudane, and 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.

[52] C. E. Perkins and 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.

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

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

[55] T. Clausen, "Comparative Study of Routing Protocols for Mobile Ad-Hoc NETworks," *Research Report, RR-5135, INRIA*, March 2004.

[56] P. Gauthier, D. Harada, and M. Stemm, "Reducing power consumption for the next generation of pdas: It's in the network interface," *In Proceedings of MoMuC'96*, Septembre 1996.

[57] Sycard technologies, "Sycard technologies, pccextend 140 cardbus extender," July 1996.

[58] B. Wang and S. Singh, "Computational energy cost of TCP," *In Proceedings of IEEE INFOCOM'04*, March 2004.

[59] D. Barman and I. Matta, "Effectiveness of Loss Labeling in Improving TCP Performance in Wired/Wireless Networks," *Boston University, Technical Report*, 2002.

[60] L. A. Grieco and S. Mascolo, "TCP Westwood and Easy RED to Improve Fairness in High-Speed Networks," *In Seventh International Workshop on Protocols For High-Speed Networks (PfHSN'2002)*, April 2002.

[61] R. Wang, M. Valla, M. Y. Sanadidi, B. K. F. Ng, and M. Gerla, "Efficiency/Friendliness Tradeoffs in TCP Westwood," *In ISCC 2002: Seventh IEEE Symposium on Computers and Communications*, July 2002.

[62] M. Mancuso, "A Novel Scheme of Loss Differentiation and Adaptive Segmentation to Enhance TCP Performance over Wireless Networks," *Politecnico di Milano, Dept. of Electronics and Information*.

[63] G. Xylomenos, G.C. Polyzos, Mahonen P., and M. Saaranen, "TCP Performance Issues over Wireless Links," *IEEE Communications Magazine*, vol. 39, no. 4, pp. 52-58, April 2001.

[64] J. Garcia and A. Brunstrom, "Transport Layer Loss Differentiation and Loss Notification," *In Proceedings of First Swedish National Computer Networking Workshop (SNCNW),* September 2003.

[65] L. Stéphane, Y. Ghamri Doudane, and G. Pujolle, "Cross-Layer Loss Differentiation Algorithms to improve TCP Performances in WLANs," *in the 11th IFIP International Conference on Personal Wireless Communications (PWC'06)*, September 2006.

[66] T. Kim, S. Lu, and V. Bharghavan, "Improving Congestion Control Performance Through Loss Differentiation," *International Conference on Computers and Communications Networks '99*, October 1999.

[67] D.-M. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," *Journal of Computer Networks and ISDN Systems*, vol. 17, no. 1, pp. 1-14, June 1989.

[68] S. Biaz and N. Vaidya, "Distinguishing congestion losses from wireless transmission losses: A negative result," *In Proceedings of Seventh International Conference on Computer Communications and Networks*, pp. 722-731, October 1998.

[69] S. Biaz and N. H. Vaidya, "Discriminating Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver," *In ASSET 1999: IEEE Symposium on Application-specific Systems and Software Engineering & Technology*, p. 10–17, March 1999.

[70] Y. Tobe, H. Aida, Y. X Aida, and H. Tokuda, "Detection of Congestion Signals from Relative One-Way Delay," *IPSJ Journal*, vol. 42, no. 12, 2001.

[71] S. Cen, P. C. Cosman, and G. M. Voelker, "End-to-End Differentiation of Congestion and Wireless Losses," *In MMCN2002: SPIE Multimedia Computing and Networking*, vol. 4673, pp. 1-15, January 2002.

[72] N.K.G. Samaraweera, "Non-Congestion Packet Loss Detection for TCP Error Recovery using Wireless Links," *In IEEE Proceedings Communications*, vol. 146, no. 4, pp. 222-230, August 1999.

[73] P. Karn and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols," *In Proceedings of ACM SIGCOMM '87*, August 1987.

[74] M. Kim and B. Noble, "Mobile Network Estimation," *In Mobicom 2001: The Seventh Annual International Conference on Mobile Computing and Networking*, July 2001.

[75] M. Allman and V. Paxson, "On Estimating End-to-End Network Path Properties," *ACM SIGCOMM*, vol. 29, no. 4, pp. 263-274, October 1999.

[76] J. Nagle, "Congestion Control in IP/TCP Internetworks," *RFC 896, IETF*, January 1984.

[77] D. Dykeman, M. Kaiserswerth, B.W. Meister, H. Rudin, and R. Williamson W. Doeringer, "A Survey of Light-Weight Transport Protocols for High-Speed Networks," vol. 38, no. 11, p. 2025–2039, November 1990.

[78] Pluggable congestion avoidance modules. http://lwn.net/Articles/128681/.

[79] A Linux TCP implementation for NS2 Linux. http://netlab.caltech.edu/projects/ns2tcplinux/ns2linux/.

[80] P. Karn and C. Partridge, Improving Round-Trip Time Estimates in Reliable Transport Protocols, August 1987.

[81] K. Jacobsson, H. Hjalmarsson, and K. H. Johansson, Unbiased bandwidth estimation in communication protocols, July 2005.

[82] A. Capone and F. Martignon, "Bandwidth Estimates in the TCP Congestion Control Scheme," *In Proceedings of Tyrrhenian Int'l Workshop Digital Comm.*, pp. 614 - 626, September 2001.

[83] Chuang-Yueh Chen, Sandra I. Woolley, Andrew J. Forgham, and Keith P. Jones, "A QoS Dynamic Bandwidth Partitioning (Q-DBP) Using Fermi-Utility Functions," *INC2002 3rd International Network Conference*, July 2002.

[84] X. Gao, T. Nandagopal, and V. Bharghavan, "Achieving application level fairness through utility-based wireless fair scheduling," *In Proceedings of IEEE GLOBECOM*, pp. 3257-3261, November 2001.

[85] J. Hoe, "Start-up Dynamics of TCP's Congestion Control and Avoidance Scheme," *Master's Thesis, MIT*, June 1995.

[86] Srisankar S. Kunniyur and Rayadurgam Srikant, "End-to-end congestion control schemes: utility functions, random losses and ECN marks," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 689-702, 2003.

[87] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options," *RFC 2018, IETF*, Octobre 1996.

[88] D. C. Montgomery, "Introduction to Statistical Quality Control," *John Wiley & Sons*, (1st edition, 1985, 2nd edition, 1991).

[89] V. Rakocevic, J. M. Griffiths, and G. Cope, "Dynamic Partitioning of Link Bandwidth in IP/MPLS Networks," *In Proceedings of IEEE ICC2001*, vol. 9, pp. 2918-2922, 2001.

[90] K. Ravindran, "Dynamic Protocol –level Adaptations for Performance and Availability of Distributed Network Services," *In 2nd IEEE International Workshop on Modelling Autonomic Communications Environments , MACE'07*, October 2007.

[91] A. Seddik-Ghaleb, Y. M Ghamri Doudane, and 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.

[92] H. Singh, S. Saxena, and S. Singh, "Energy Consumption of TCP in Ad Hoc Networks," *Journal Wireless Networks*, vol. 10, no. 5, September 2004.

APPENDIX ONE: SEDLANE AVAILABLE VERSIONS

We have developed different versions of SEDLANE: command line tool, graphical user interface version and SEDLANE distributed tool. The command line tool can be used by the bash expert users. In this version, the user should have the necessary knowledge about all the SEDLANE arguments and options and their meanings. These arguments and options include:

The source IP address and the destination IP address;

NS TCP trace file from which SEDLANE will get the required parameters needed for Dummynet configuration (RTT Classes, BW, Loss Rates);

The number of pipes corresponding to the maximum number of Dummynet's pipes or communication channels to be configured on the SEDLANE machine. It is used to group the packets together. The higher the number of pipes used, the most accurate the results can be found;

Data flow direction corresponding to the direction on the local machine. It can be "out" or "in". If "out", Dummynet rules will be applied before the data packets being sent, and if "in", Dummynet rules will be applied on the incoming data packets;

The interface option is used to specify the interface on which the *ipfw* rules will be applied. Naturally, this must be the interface that is used to send, receive, or relay data packets between the source and the destination. This is helpful when there are many interfaces within the machine;

SEDLANE operation mode option (simultaneous or sequential), when this option is not used, SEDLANE acts in simultaneous mode (the default mode);

In the Verbose mode option, if used, SEDLANE produces detailed display about the rules' configuration and asks for user's confirmation before launching *ipfw* configuration. Verbose mode can be skipped, simply do not supply [-v] as a command line argument and SEDLANE will pass directly to the configuration phase;

The Flow type option enables the user to use SEDLANE with a specified data packet type (TCP, UDP, and ICMP). The user may simply ignore this option, and the default will be any IP traffic.

The second available version of SEDLANE is the graphical user interface. In this version, the user does not need to know the argument names as the SEDLANE interface screen specifies the needed arguments according to the desired operation mode. The user only fills in the variables in the window and presses Launch. This version is simple to use and does not require any additional knowledge with the underlying configuration of SEDLANE. The arguments used by the graphical user interface version have the same significance as in the command line tool. Only "verbose" mode is not available when using SEDLANE graphical user interface.

Finally, a distributed tool is also available. SEDLANE is used between any two communicating end points in order to introduce the effect of ad hoc network characteristics over the connection. When we think about having a realistic test-bed with more than one connection, having many SEDLANE emulators within the network would be a good idea (e.g. testing applications like video games over such networks). The idea of the distributed mode is as follows: instead of moving around to install SEDLANE on each desired node, we propose using a centralized tool to distribute the required configuration and the NS-2 files to be used over the network. Through a centralized machine we can install SEDLANE remotely over all the other machines over the network. This is done through graphical user interface tool; the user opens a remote session with the selected machines to install SEDLANE emulation tool on them. After installing SEDLANE over these machines, the user will be able to configure the *ipfw* rules used by SEDLANE (remotely by running SEDLANE with all the necessary arguments for each machine separately; such as in simple graphical user interface mode). The distributed version of SEDLANE facilitates the configuration of SEDLANE over large scale networks where more than one emulation node is needed. In addition, the user can configure each SEDLANE instance over the network with different connection conditions by specifying different NS-2 trace file for each connection and he can also vary all the SEDLANE arguments according to the emulation requirements needed.

PATENTS

1- A. Seddik Ghaleb, Y. M. Ghamri Doudane , S. M. Senouci, and N. Agoulmine, "*TCP Loss Recovery Algorithm in Ad hoc Networks*". French Patent 0703391, issued on 11st Mai, **2007**. (Extension to international patent is in progress).

INTERNATIONAL CONFERENCES

2- A. Seddik Ghaleb, Y. M. Ghamri Doudane and S. M. Senouci, "*Computational Energy Cost of TCP in MANETs*", in the 7th ACS/IEEE International Conference on Computer Systems and Applications, AICCSA'09, Rabat, Morocco , Mai, **2009**.

3- A. Seddik Ghaleb, Y. M. Ghamri Doudane and S. M. Senouci, "*TCP WELCOME: TCP variant for Wireless Environment, Link losses, and COngestion packet loss ModEls*", in the 1st international Conference on COMmunication Systems and NETworks, COMSNETS'09, Bangalore, India, January, **2009**.

4- A. Seddik Ghaleb, Y. M. Ghamri Doudane and S. M. Senouci, "*TCP Computational Energy Consumption within Wireless Mobile Ad Hoc Network*", in the 33rd IEEE Conference on Local Computer Networks, IEEE LCN'08, Montreal, Quebec, Canada, October, **2008**.

5- A. Seddik Ghaleb, Y. M. Ghamri Doudane and S. M. Senouci, *"Emulating End-to-End Losses and Delays for Ad Hoc Networks"*, In 43rd IEEE International Conference on Communications 2007, ICC'07, Glasgow, Scotland, June, **2007**.

6- A. Seddik Ghaleb, Y. M. Ghamri Doudane and S. M. Senouci, "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", In 2nd ACM International Wireless Communications and Mobile Computing Conference, IWCMC'2006, Vancouver, Canada, July, **2006**.

7- A. Seddik Ghaleb, Y. M. Ghamri Doudane and 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, New York, June 28-30, **2006**.

NATIONAL CONFERENCES

8- A. Seddik Ghaleb, Y. M. Ghamri Doudane and S. M. Senouci, *"Études de la consommation d'énergie des versions TCP dans les réseaux ad hoc mobiles"*, In De Nouvelles Avancés pour les Communications, DNAC'2004, (Paris, France), December, **2004**.