COMMUNICATION PROTOCOLS FOR WIRELESS AD-HOC AND SENSOR NETWORKS

By

JIN DING

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

MAY 2006

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of JIN DING find it satisfactory and recommend that it be accepted.

_____
Chair

_____

_____

ACKNOWLEDGEMENT

First and foremost, I would like to thank Dr. Sirisha Medidi, my thesis advisor and mentor for the past 4 years. Dr. Medidi has been a wonderful advisor, providing me with invaluable advice, support. Her enthusiasm for research, breadth of knowledge inspires me. Her encouragement gives me confidence. I thank her for all the time and energy has invested into my research.

I would also like to thank Dr. Murali Medidi and Dr. Curtis Dyreson for agreeing to be on my committee. I thank them for reading my thesis and providing me helpful comments. I was fortunate enough to work with them. Their enthusiasm and optimism always amaze me. My thanks also to the folks at the SWAN research lab, Yong, Yuanyuan, Chris, Roger for helping each other and having fun together.

I would like to thank Dr. Bruce Kwan, my colleague in Broadcom Corporation for his generous assistance in my dissertation.

I would like to thank the graduate secretary Ms. Ruby Young, for all her help.

I cannot fully express my gratitude to my family. I am forever indebted to my parents for everything that they have given me. Their unconditional support and encouragement give me strengths to finish this work. I also would like to express my deep appreciation for my grandmother. For wonderful time I had with her when I was in China. For the value and joy of love and life she has shown me. I want to thank my sister Feng Ding for her love and support. I am very lucky to have such a wonderful family. I dedicate this work to them; to all the people who love me and whom I love.

Last, but far from the least, I thank my husband, Lin Xu from bottom of my heart. His suggestions were very helpful. I thank him for all his patience and his never-ending optimism when I frustrated and stressed. He makes my life full of joy and love. I cannot imagine that I could have completed this work without his support.

COMMUNICATION PROTOCOLS FOR WIRELESS AD-HOC AND SENSOR NETWORKS

Abstract

by Jin Ding, Ph.D.
Washington State University
May 2006

Chair: Sirisha Medidi

Demand for decentralized wireless ad-hoc systems, where hosts are free to leave or join, to replace wired communication systems has seen a phenomenal growth. The protocols developed for wired systems cannot handle the new problems that come with wireless networks. This inability lays our entire communications capability open to disruption, and requires entirely new protocols.

Traditional TCP (Transport Control Protocol), designed for wired networking, degrades significantly over wireless links and provides abysmal throughput. This performance degradation occurs because the TCP protocol is carefully tuned for wired networks, where most packet losses are primarily due to congestion loss. However, in wireless ad-hoc networks, packet loss could be because of several reasons such as link loss, node mobility and network misbehavior to name a few. Current techniques for improving TCP do not consider the malicious packet drop attack.

Energy-efficient information dissemination is a critical operation in wireless sensor networks. Conventional protocols like flooding or gossiping have problems such as data implosion, overlap, and resource blindness. SPIN (Sensor Protocols for Information via Negotiation), proposed to handle these issues uses negotiation with meta-data descriptors and resource-adaptation. However energy-efficiency was not considered in this design.

This dissertation addresses these challenges by providing two communication protocols, one that improves TCP performance over lossy links and another that provides energy-efficient data distribution.

First, TCP-Manet, a reliable transport protocol over wireless ad-hoc networks is introduced. TCP-Manet determines the characteristics of the packet loss based on current connection status and reacts effectively to the packet loss. Theoretical and simulation results show that TCP-Manet provides better

throughput performance than TCP-Reno.

Next, SPIN-G, an energy-efficient data dissemination protocol is developed. SPIN-G adapts to the remaining battery power at any sensor node there by extending network lifetime. This protocol also uses the negotiation, meta-data description, and resource adaptation features of SPIN and improves on the negotiation by explicitly accounting for the battery capacity. Simulation experiments confirm that SPIN-G dissipates less energy compared to SPIN and converges (all nodes in the network get all the data) slightly slower than SPIN.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER ONE

INTRODUCTION

The advent of wireless communications emerged as early as the beginning of the 20th century. Initial applications for wireless communications were focused on voice communications (i.e. cellular phone). As wireless technology matures, more and more people are enjoying the benefits of wireless networks, such as lower cost and increased mobility for users. As the technology gains popularity, users are developing more reliance on wireless access for data communications. In addition, users demand high performance from the wireless network [2].

Although the increasing popularity of wireless networks indicates that wireless links will play an important role in future inter-networks [3], wireless communication has two unique resource limitations - bandwidth and energy - as compared to current wired networks. This resource limitation constrains the application of wireless networks. Therefore, it requires innovative communication techniques to increase bandwidth utilization and innovative design techniques and protocols enable efficient energy utilization. Furthermore, wireless channels are inherently error-prone and time varying. These characteristics make it difficult to consistently obtain desired performance, which adds more challenges to the communication protocols designed for this dynamic environment.

This dissertation addresses these wireless networking challenges by providing two communication protocols, one that provides improved TCP performance over lossy links and one that provides energy efficient data distribution. The two protocols presented in this dissertation include the following:

1. TCP-Manet: A TCP enhancement for wireless ad-hoc networks
2. SPIN-G: An energy-aware data dissemination protocol in wireless sensor networks.

1.1 Wireless Ad-Hoc and Sensor Networks

A mobile ad-hoc network (MANET) is a self-configuring network made up exclusively of mobile hosts connected by wireless links to form an arbitrary topology. The network has no access points (APs) and operates in peer-to-peer operating mode. The mobile hosts are free to move randomly and organize themselves arbitrarily; therefore the network's topology may change rapidly and unpredictably.

Ad-hoc networks may operate in a standalone fashion, or may be connected to the larger Internet. The mobile hosts in ad-hoc networks usually have the entire protocol stack as the fixed hosts in wired network to provide inter-operative and compatibility with the Internet. As a consequence, solutions to any new protocols should consider the inter-operativity with the current Internet.

Wireless sensor networks can be considered as a subset of ad-hoc networks (MANETs). However, there are inherent differences between the two. For example, MANETs are associated with a high degree of mobility, unlike sensor networks which are stationary. Unlike in MANETs, addressing in a sensor network is not as important as data gathering. More importantly, sensor networks are usually application specific. Such applications may monitor a variety of environments that include home security, machine failure diagnosis, chemical/biological detection, medical monitoring, and surveillance. Therefore, rather than using a general-purpose protocol architecture, most sensor networks deploy application specific protocols that can exploit features of the application to achieve greater performance.

1.2 TCP Enhancement (TCP-Manet) for Wireless Ad-Hoc Networks

Transport layer is an essential part of the protocol hierarchy that provides reliable, cost-effective data transport from the source machine to the destination machine [4]. In theory, transport layer protocols should be independent of the technology of the underlying protocols. However, in practice, the transmission control protocol (TCP) is a transport protocol that is tuned primarily for wired networks. Typically, the TCP congestion control mechanism is triggered when packet loss occurs, which is detected based on a timeout mechanism or upon receipt of duplicate acknowledgements (ACKs). Because TCP assumes all packet drops are due to congestion, the sender reduces the congestion window size, thus reducing the sending rate.

This mechanism works well in wired networks due to the low packet error rate. However, it can perform quite poorly when used over wireless links, especially in wireless ad hoc networks. Figure 1.1 shows how TCP-Reno performs in an ad-hoc network while the packet error rate increases from 0% to 5%. The TCP throughput reduces by approximately 65%. This degradation occurs since the packet losses could be due to different reasons. Depending on the reason for the packet loss, the system should trigger different recovery tactics:

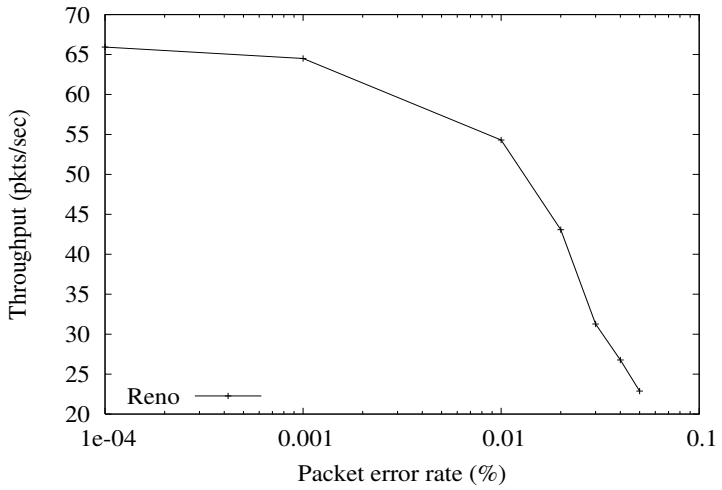


Figure 1.1: Performance of TCP-Reno over wireless network.

- **Congestion.** To keep connections in equilibrium, TCP uses congestion avoidance to probe the bandwidth available for the connection. Once the TCP sending rate is higher than the available bandwidth along the path, packet loss may occur. In a wired network, TCP periodically experiences packet losses that are assumed to be due to congestion. In this case, the TCP sender should adjust its congestion window size to reduce the sending rate, thereby reducing the network load and alleviating the congestion condition in the network.

- **Wireless Link Error.** Packets can also be lost due to a transient random loss. For this kind of packet loss, TCP should have a different recovery strategy rather than slowing down the sending rate.

- **Broken Link Error due to Node Mobility.** A mobile ad-hoc network (MANET) is infrastructureless. Any host in the network is free to join and leave, thereby resulting in a highly dynamic network topology. The broken link errors due to node mobility in ad-hoc networks can cause route errors. In this case, the sender should freeze the window and timeout and suspend data transmission until the route recovered, and then retransmit the packet as soon as possible.

- **Network Misbehavior.** In an ad-hoc network, some hosts may intentionally fail to execute their part of a network protocol in order to save its resources (such as battery power). For instance, selfish nodes may refuse to take part in forwarding any packets, and simply drop all the packets it receives

to save its own energy. This can lead to a TCP sender that keeps on retransmitting the packets and exponentially backs off the timeout. The TCP sender remains idle for a long time which reduces the throughput performance and may even result in the resetting of the connection. In this case, if the lower layer does not detect the misbehavior right away, the TCP sender should inform the lower layer with the error and trigger the lower layer to find a new route in the network, instead of being frozen and closing the TCP connection.

The objective of this study is to provide a transport layer protocol that is capable of performing efficiently in wireless ad-hoc networks. To provide the inter-operatability in new protocols, the new protocol design enhances the current TCP protocol rather than providing an entire new transport layer protocol.

1.2.1 Design Goals for Transport Layer Protocols in Wireless Ad-Hoc Networks

The new TCP enhancement should provide reliable and efficient data transfer in wireless ad-hoc networks with the following features:

- Improved Throughput. Traditional TCP is tuned for wired networks and suffers throughput degradation in wireless networks. TCP-Manet should provide better throughput than traditional TCP with the new enhancements.
- Sender Side Only Modifications. Since an ad-hoc network is a highly dynamic system, it is important to design an algorithm that does not require additional modifications to the nodes in the network.
- Cross Layer Design. In conjunction with the network layer, TCP-Manet should be able to gain more information about connection status. With this information, it can determine the nature of the packet loss, thereby triggering different recovery tactics.

1.2.2 Challenge: Meeting the Design Goals

Traditional TCP is designed for wired networks that are characterized with a low error rate. It detects packet loss by observing duplicate ACKs and timeouts, and assumes all packet losses are due to congestion error. However, when the wireless link comes into the picture in data communication, the error detection and correction does not meet the transmission requirement any more. The network suffers from high packet

loss rate due to its relatively dynamic nature (fading channel, prolonged and frequent burst errors). This leads to some undesirable patterns of behavior for TCP protocols, resulting in a performance degradation of the throughput. For example, when there are random or short bursts of link errors that lead to packet losses, TCP invariably interprets these events as resulting from congestion. TCP then reduces the window size, hence reducing the sending rate. Subsequently, the sender applies a conservatively gradual increase to its window size. During this phase, bandwidth and opportunities for error-free transmissions are wasted and the throughput is reduced. In addition, since an ad-hoc network has no fixed infrastructure to establish communication, the nodes can move freely, and each node can act as a host and a forwarding node. It is a highly dynamic and unpredictable network, and node mobility may trigger exponential back-off in the TCP protocol. However, the purpose of the timeout and exponential back-off scheme is only for avoiding major transmission errors at the cost of significantly degraded throughput. The TCP sender will be unnecessarily frozen due to packet drops for reasons other than congestion.

In summary, the central problem is that TCP suffers from degraded performance in wireless networks. This degradation occurs due to TCP's inability to correctly detect the nature of the error, and to respond in an appropriate manner [5, 6, 3, 7]. In addition, the traditional scheme of congestion control, which shrinks the congestion window in the event of a retransmission or timeout, does not necessarily suffice for wired/wireless networks. Although it has the merits of simplicity, it degrades the ability to rapidly detect error conditions and recover immediately.

1.2.3 Solution: TCP-Manet

Like TCP, TCP-Manet detects packet losses by observing duplicate ACKs and timeouts. Besides that, TCP-Manet tries to determine the nature of the packet losses based on the current connection status, and then invokes the corresponding recovery strategy. These packet losses may be due to congestion loss, wireless link error, and network misbehavior etc. Since we consider more types of errors, TCP alone can not handle all these packet loss errors, because TCP protocol has limited information about the network. The only information which it has access to is RTT (Round Trip Time) and the acknowledgement. Hence, TCP-Manet uses cross layer design strategy that asks for more information from the lower layer. TCP-Manet and the lower layer interact to enable higher layer to obtain network information such as routing message, and to provide reliable data transfer.

TCP-Manet monitors the trend of the “power” metric that is defined as the ratio of throughput and delay. When a packet loss is detected by duplicate ACKs, if the “power” is in an increasing trend, that means the network link is under utilized. The sender will only retransmit the packet without reducing the congestion window size. Otherwise the sender will trigger a new congestion avoidance algorithm designed in TCP-Manet. If a packet loss is detected by a timeout, TCP-Manet retransmits the packet while holding the congestion window size unchanged. If the sender gets a new acknowledgement, which means it is a congestion error, the TCP sender will set the congestion window size to 1. After four timeouts, TCP sender starts to send probe messages to the destination to identify if there is a selfish node in the connection.

In addition, we present a theoretical model for TCP Manet in terms of the throughput, and compare it with simulation results. In the simulation evaluation of the TCP-Manet, we study the throughput, drop rate, fairness, backward compatibility etc. We also compare our results with TCP-Reno. The simulation results show that TCP-Manet has better performance then traditional TCP over wireless ad hoc networks.

1.3 Data Dissemination Protocol (SPIN-G) in Wireless Sensor Networks

Rapid technological advances in wireless communication have made it possible to network low cost, low complexity miniature sensor devices to capture environmental and tactical data and disseminate them around the network. This brings the new application of wireless communication networks - sensor networks.

The sensors are equipped with a wireless communication transceiver and a reasonably powerful processor which is capable of signal processing and complex computations. The main functionality of these sensors is to monitor a variety of environmental events. In a sensor network that consists of a number of sensor nodes, the sensor nodes gather data and disseminate them throughout the sensing area via a wireless channel.

Data dissemination occurs when sensor nodes (source node), which detects an environment events (stimulus), distributes its observations to other sensors (sink nodes) that are interested in collecting this data [8] [9]. It has many potential applications in military and surveillance. For example, several hundred sensors can be scattered in a battle area to form a wireless sensor network. Sensors in the network detect the existence of enemies and disseminate their observations to other sensors. When soldiers enter this area, they can obtain this information from any sensor in the network.

1.3.1 Design Goals for Data Dissemination Protocols in Wireless Sensor Networks

The objective of this study is to present an energy-efficient data dissemination protocol (SPIN-G) for wireless sensor networks. This effort aims to meet the following requirements:

- Energy efficient to extend the network lifetime,
- Scalable to enable a large number of nodes in the system, and
- Allow timely distribution of information throughout the network with an acceptable latency.

1.3.2 Challenge: Meeting the Design Goals

The design and implementation of a data dissemination protocol poses several significant and interesting challenges:

- Energy Efficiency. Sensor nodes are characterized by limited computation, memory storage, communication bandwidth, and battery power capability. In some scenarios, the sensors can not be recharged once their energy is drained. Hence, the lifetime of the network depends heavily on how efficiently the nodes are able to perform its duties of gathering, processing, and distributing information. This means that the data dissemination protocol should consume as little energy as possible, thereby extending the network lifetime.
- Scalability. The sensor network is usually densely deployed which consists of hundreds or even thousands of sensor nodes in the field. Blindly broadcasting data to other sensor nodes would generate a high network overhead. Avoiding flooding storm while designing a network protocols is also an important challenge.
- Timeliness. Data gathered from sensor nodes are typically time-sensitive. It is essential to receive the data in a timely manner. In addition, increased latency typically lead to data retransmission which leads to unnecessary power wastage. Hence, it is imperative to develop a protocol that has acceptable latency.

1.3.3 Solution: SPIN-G

We designed the SPIN-G protocol that is motivated by SPIN protocol (Sensor Protocol for Information via Negotiation) [10, 11]. It employs meta-data negotiation before initializing the real data operation to minimize the redundant data transmission to save energy over classical flooding. However, in SPIN, meta-data exchange is based on flooding, which introduces network overhead of meta-data exchange and could incur flooding storm problem that deteriorates performance in a high density network.

To further reduce the energy consumption of SPIN, SPIN-G employs a randomized algorithmic "gossip" and data aggregation scheme to reduce the network overhead. Like SPIN, SPIN-G is a meta-data negotiation based protocol. That is, before transmitting data, the sensor nodes will advertise its data using meta-data, wait for request from other sensor nodes, and then send data to the requesting node. Unlike SPIN, which uses flooding for data advertising, SPIN-G employs a gossip algorithm in which each sensor node only advertises data to a randomly chosen neighboring node. Gossiping, which informs only one neighbor instead of all neighbors, has the slowest distribution rate of data dissemination and introduces a latency penalty. To alleviate this penalty, we utilize a data aggregation scheme, in which each sensor aggregates old data with new data for advertising. Data aggregation not only can fasten the timeliness of the protocol, but also deal with packet losses in the network. Hence, combining gossip and a data aggregation scheme, we can achieve energy conservation at the expense of slightly increased latency, and improve the robustness of the protocol as well.

Energy efficient data dissemination not only means a low energy consumption level, but also means balanced energy consumption distribution throughout the network. This leads to the benefits of enhancing the networks ability to remain connected and extending the network functionality. To reach this goal, instead of responding with a data request packet immediately after receiving an advertisement as in SPIN, sensor nodes (upon receiving multiple data advertisement messages (ADVs) for the same data) in SPIN-G select advertising neighbor with the highest energy level.

Finally, from an energy savings point of view, an efficient mechanism is to place sensor nodes in sleep mode [12]. However, putting all nodes in sleep mode will reduce the responsiveness of the protocol. Specifically, it will increase the data dissemination protocol convergence time. SPIN-G only applies the sleep mode to the sensor nodes whose battery power levels are below a threshold, hence, preventing battery

poor nodes in the network from dying faster which can lead to network partition. The sensor node will sleep and wake up periodically during a sleeping-active cycle. This scheme aids by minimizing the need for nodes to participate in the dissemination process, thereby leading to an improvement in the life expectancy of a battery impoverished sensor device.

We conducted analytical and simulation-based analysis of proposed protocol, and compare with SPIN protocol [10]. The results show that although SPIN-G has a slightly higher protocol convergence time than SPIN, it consumes about 20% less energy than SPIN. With the increase of the network density, our protocol reduces the energy consumption by about 50%. Introducing a sleep cycle in SPIN-G may also substantially increase the network lifetime.

1.4 Key Contributions of this Research

The results of our research show that TCP-Manet can provide the high performance needed for wireless networks. Traditional TCP is designed using a layered approach that can provide necessary performance over wired networks. However, over a wireless networks, TCP suffers from performance degradations. To enhance the performance of TCP over wireless ad hoc networks, we found that using a cross-layer design, where network layer and transport layer are exposed to information from each other, produces improved performance. We define the appropriate information that should be passed across the layer. TCP-Manet provides the improved performance by effectively utilizing information from the lower layer.

The main contributions of TCP-Manet are as follows:

- Error detection capability that can determine the nature of packet loss while TCP-Manet is in operation.
- Cross layer design that overcomes the limitation of TCP that have limited access of the network information.
- Theoretical model that enable analysis of the behavior of the TCP-Manet.

In our research with data dissemination protocol SPIN-G, two primary performance metrics are studied - protocol convergence time that defines all the nodes receiving the data distributed by all the other nodes in the network and energy consumption of the data dissemination process. It is important that the data

dissemination protocol can converge in an acceptable time and consume as little energy as possible. This leads us to a design of SPIN-G with following features:

- Gossiping with data aggregation to further reduce network overhead introduced by meta-data negotiation.
- Make sensor nodes request data from most energetic neighboring node to balance the energy consumption throughout the network.
- Utilize a sleep/active cycle to battery impoverished nodes to protect them from depleting their energy and improving network lifetime.
- Achieving acceptable convergence time of the protocol.

1.5 Organization of the Dissertation

In the following chapters, both protocols, TCP-Manet and SPIN-G, are discussed in detail. Chapter 2 provides the necessary background on current wireless ad-hoc and sensor networks. Chapter 3 addresses the TCP enhancement (TCP-Manet) that improves TCP performance over wireless ad hoc networks by providing detection component for TCP protocol. We present the description of the proposed network protocol TCP-Manet. And then, we give the theoretical analysis, and simulation model and results of our research. In chapter 4, an energy-aware data dissemination protocol (SPIN-G) is developed that focuses on the data distribution on the sensor networks. SPIN-G is an improvement of SPIN [10]. After discussing the mechanisms used in SPIN-G to save energy, we give the theoretical and simulation evaluation of the protocol. This thesis ends with conclusions and discussions of the future work in Chapter 5.

CHAPTER TWO

BACKGROUND

2.1 Wireless TCP

TCP's behavior over wired networks, where congestion is a regular cause for packet loss was initially studied by Jacobson [13]. Recently, TCP behavior over wireless networks has become a focus of attention. Recent research results [4, 14, 15, 3, 16, 17, 18, 6, 19] have shown that TCP throughput (i.e. sending rate) degrades, in the presence of random/burst errors and long propagation delays. As a result, some researchers have tended to focus on the development of architectures (e.g., wireless proxies) that assist the protocol's operation over specific networks in order to introduce minor changes to the protocol itself. Therefore, a large number of suggested proposals aiming at improving TCP performance and avoid or control network congestion deal with the functionality of network devices that can assist the protocol operations. These works can be classified into two categories, (i) cross layer design; and (ii) layered design.

In this section, we review the research that has been proposed on TCP enhancement in wireless ad-hoc networks. First, we will discuss the techniques that have been proposed to improve TCP performance in wireless networks. Then, we will review the research in TCP protocol design in ad-hoc networks. We classify the proposals and research into two categories: (1) cross layer design and (2) layered design.

2.1.1 TCP in Cellular and Satellite Networks

When the wireless networks first came into existence, it usually meant the cellular networks or satellite networks, which we refer to as heterogeneous wired/wireless network. In this kind of network, there is a base-station in between the cellular/satellite network and wired network. Base-station can control the communication within the cell, and act as an interface between the mobile node and Internet. Research results [4, 14, 15, 3, 16, 17, 18, 6, 19] have shown that TCP throughput (i.e. sending rate) degrades, in the presence of random/burst errors and long propagation delays. As a result, some researchers have tended to focus on the development of architectures (e.g., wireless proxies) that assist the protocol's operation over wireless networks in order to introduce minor changes to the protocol itself. In these proposals, most commonly used approach for improving TCP performance between mobile nodes and fixed nodes in the Internet is proxy-based solution. In this mechanism there is a proxy implemented at the base-station at

the wired-wireless boundary to hide the effect of wireless error from wired network, avoid timeouts or fast retransmission at the sender, and avoid the exponential back-off of the timeout value.

The proxy-based solution usually buffers data segments at the proxy and retransmits them over the local wireless link if they get lost due to transmission error. Duplicated acknowledgements resulting from wireless losses are dropped to prevent from triggering the fast retransmission, therefore avoid window shrinkage at the TCP sender. For some explicit approaches, some ECN-like notifications are sent to the sender to shield the effect of wireless losses on the retransmission timeout maintained at the sender. Then the sender reacts to the congestion quickly and allows the proxy the chance to locally repair wireless losses.

The Indirect-TCP (I-TCP) [20] splits the TCP connection into two separate connections. The first connection goes from the sender to the base station. The second one goes from the base station to the receiver. Hence the base station maintains two TCP connections, one over the fixed network, and another over the wireless link. The base station simply copies packets between the connections in both directions to hidden the wireless link errors from the wired network. I-TCP does not maintain end-to-end TCP semantics. It relies on the application layer to ensure reliability. That means, if the application has the ability to provide reliability, it uses I-TCP, otherwise chooses TCP. This implies that the mobile hosts in the network must be aware of an application's ability to provide reliability for choosing a proper protocol for the transport layer. The base station should be informed about the mobile host's selection. The advantage of I-TCP is that both connections are now homogeneous. Parameters can be tuned separately for the different connections. The disadvantage of the scheme is that it violates the semantics of TCP. Since each part of the connection is a full TCP connection, the receipt of an acknowledgement does not mean that the receiver got the segment, only that the base station got it.

MTCP [21] is similar to I-TCP, except that the last TCP byte of the data is acknowledged to the source only after it is received by the mobile host. If the sender does not receive the acknowledgement for the last byte, it will resend all the data including those that may have already been received by the mobile host. The base station sends ZWA (zero window adjustment) to freeze the source during handoffs, so the window and timeout are not affected.

Explicit Bad State Notification (EBSN) [22] uses local retransmission from the base station to shield wireless link errors and improve the throughput. The sender timeout may be avoided by using explicit

feedback mechanism, while wireless link is in bad state. In EBSN approach, the base station sends an EBSN message to the source for every retransmission of a segment to the mobile host, and the sender will reinitialize the timer upon the received EBSN message. The main disadvantage of this approach is that it requires TCP code modification at the source to be able to interpret EBSN message.

WTCP [16] is also similar to I-TCP, with the exception the base station acknowledges a TCP segment to the sender only after that segment is acknowledged by the mobile host preserving the TCP end-to-end semantics. It hides the time spent by a TCP segment in the base station buffer to avoid affecting RTT estimates and timeout maintained at the sender. This is achieved by modifying the timestamp field in acknowledgement instead of using explicit feedback message. For the reliable connection from the base station to the mobile host, base station reduces its window size to one segment in case of timeout. It assumes that a typical burst loss will follow, rapidly reducing the window size to avoid the wasteful retransmissions and interference with other channels. Upon each acknowledgement, the WTCP sender will assume that an ACK indicates that wireless link is in good state and set its window size to advertised window size by the receiver (mobile host). For duplicate acknowledgment, WTCP does not alter the wireless transmission window assuming that the reception of the duplicate acknowledgement is an indication that the wireless link is in good state, and immediately retransmits the lost segment. The base station can decide the number of duplicated acknowledgment to the receiver to cease the retransmission at the sender side, therefore improving the utilization of the wireless channel.

Snoop [23] is similar to WTCP, except that it is implemented at the link layer of the base station. The base station sniffs the link interface for any TCP segments destined for the mobile host, and buffers them if buffer space is available. The retransmitted segments that have already been acknowledged by mobile host are not forwarded by the base station. The base station also sniffs into the acknowledgements from the mobile host. It detects the loss by duplicated acknowledgement. If it is wireless link error and the segment is buffered, the base station retransmits the lost segment and starts a timer. Although Snoop does not break the semantics of TCP, it makes several small modifications to the network layer code in the base station. One of the changes is the addition of a snooping agent that observes and caches TCP segments going out to the mobile host and acknowledgements coming back from it. When the snooping agent sees a TCP segment going out the mobile host but does not see an acknowledgement coming back before its (relatively

short) timer goes off, it just retransmits that segments, without telling the source that it is doing so. It also retransmits when it sees duplicate acknowledgements from the mobile host go by, invariably meaning that the mobile host has missed something. Duplicate acknowledgements are discarded on the spot, to avoid having the source misinterpret them as congestion.

Fast-retransmission [24] reduces the effect of mobile host handoff by delayed acknowledgments for controlling the sender's transmission rate at the receiver. During a mobile host hand-off from one base station to another, TCP segments can be lost or delayed, and the source can timeout. Because of the typical coarse granularity of the TCP clock, the timeout period is much higher than hand-off time, and the mobile host has to unnecessarily wait for a long duration to receive a retransmission of a lost TCP segment from the source. In the fast retransmit approach, immediately after completing the hand-off, the IP in the mobile host triggers TCP to generate a certain number of duplicate acknowledgments. Since most of TCP implementations now have fast-retransmit, these duplicate acknowledgments cause the source to retransmit the lost segment without waiting for the timeout period to expire.

There is also some additional work in proxy-based mechanism [25, 26, 27, 17]. A comparison of some of the above solutions is given in [28]. For a good quality wireless link, the throughput of I-TCP is better. When the wireless link quality degrades WTCP yields better throughput mainly because of its aggressive retransmission policy over the wireless link. WTCP achieves throughput values 4-8 times higher than TCP-Tahoe. However, the aforementioned policy degrades (slightly) the utilization of the wireless link. In addition, though Snoop achieves throughput values comparable to I-TCP and WTCP at low loss situations, the throughput of Snoop is poor when the wireless link is very bursty (in bad error state) for long durations. The reason is that Snoop triggers retransmission only after the base station receives a duplicate acknowledgment. When the wireless link is in a bad state, acknowledgments might be lost.

2.1.2 TCP in Ad-Hoc Networks

Ad-hoc network is different from the cellular and satellite network. Since it is infrastructureless, there is no base station operated in the network, shown in figure 2.1. In addition, since the ad-hoc network is highly dynamic, and all the links are wireless, it is not feasible to assign a proxy in the network to take in charge of the data communications. In the following part, we will review some TCP approaches used in ad-hoc networks.

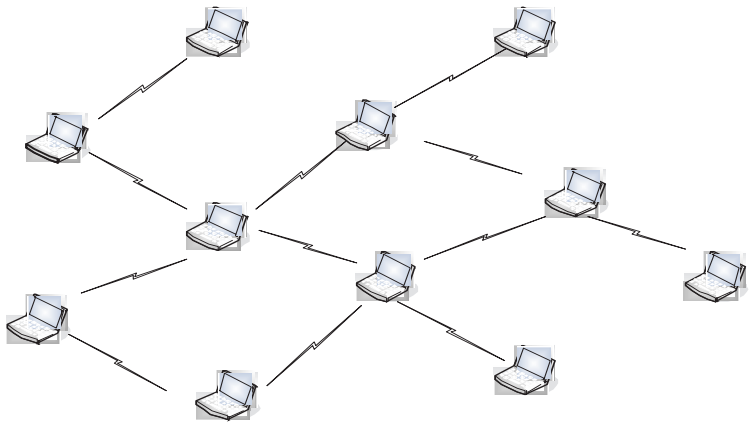


Figure 2.1: Wireless Ad-hoc Network.

Usually, the techniques can be classified into two categories: cross layer design and layered design.

Cross Layer Design

Cross layered design usually involves more than two OSI layers. The lower layer can provide sender with explicit information about the nature of the error or attempt to hide altogether the error from the sender.

An Explicit Congestion Notification (ECN) scheme is proposed in [29], which adds ECN to the IP protocol to trigger TCP congestion control. The intermediate nodes in the network send ECN to the sender to notify the sender its limited buffer space has been full. Upon receiving the ECN message, the TCP sender will trigger the congestion control algorithm and avoid congestion collapse. Therefore, it is beneficial to the sender to know that congestion is precisely about to happen. However, since ECN message could be lost, not receiving an explicit notification does not mean a detected drop was not caused due to congestion. On the other hand in an ad-hoc network, we cannot guarantee that all the intermediate nodes are ECN-capable.

TCP-Feedback [30] tries to handle the effect of the node mobility in ad hoc networks. It is a feedback-based scheme in which the TCP sender can distinguish between route failure and network congestion by receiving Route Failure Notification (RFN) from intermediate nodes. When an RFN message is received, TCP-Feedback will push the TCP sender into a “snooze state”, where TCP stops sending packets and freezes all its variables such as timers and CWND size. This makes sense since there are available routes to the destination temporarily due to the node mobility. When the new route to the destination is reestablished, a Route-Re-establishment Notification (RRN) will be received by the sender. The sender can leave the frozen

state and resume transmission using the same variable values prior to the interruption. In addition, a route failure timer is used to prevent infinite wait for RRN messages. It is triggered whenever an RFN is received, and in case it expires the frozen timers are reset allowing the TCP congestion control to be invoked normally. TCP-Feedback showed gains over standard TCP in conditions where the route reestablishment delays are high. It also performs better for scenarios with high rates. However, since RGN and RRN message should be carried by the routing protocol, no such protocol was considered in the evaluation.

ELFN-based approach [13] provides meaningful enhancements over standard TCP. It uses Explicit Link Failure Notification (ELFN) message to interact with the routing protocol in order to detect route failure and take appropriate actions when that is detected. When a wireless link error happens, the node detecting the failure will send the ELFN message back to the sender. The ELFN messages contain sender and receiver address and ports, as well as the TCP sequence number. The TCP-ELFN is able to distinguish losses caused by congestion from the route failure. When the TCP sender receives an ELFN message it enters a "stand-by" mode, which implies that its timers are disabled and probe packets are sent regularly towards the destination in order to detect the route restoration. Upon receiving an ACK packet the sender leaves the "stand-by" mode and resumes transmission using its previous timer values normally. This scheme was evaluated for the DSR routing protocol where the stale route problem was found to be crucial for the performance of this modified TCP as well. Additionally, the length of the interval between probe packets and the choice of which type of packet to send as a probe was also evaluated. Only the former showed to be really relevant. It suggests that a varying interval based on RTT values could perform better than the fixed probe interval used in this algorithm. Another interesting investigation performed by this study was the impact of ARP (Address Resolution Protocol) protocol on TCP efficiency, which calls for improvements.

ATCP (Ad-hoc TCP) protocol [14] does not impose changes to the standard TCP itself. It implements an intermediate layer between network and transport layers. In particular, this approach relies on the ICMP (Internet Control Message Protocol) protocol and ECN (Explicit Congestion Notification) scheme to detect network partition and congestion, respectively. In this manner, the intermediate layer keeps track of the packets to and from the transport layer so that the TCP congestion control is not invoked when it is not required. When three duplicate ACKs(Acknowledgements) are detected, indicating a lossy channel, ATCP puts TCP in "persist mode" and quickly retransmits the lost packet from the TCP buffer; after receiving the

next ACK the normal state is resumed. In case an ICMP “Destination unreachable” message arrives, pointing out a network partition, ATCP also puts the TCP in “persist mode” which only ends when the connection is reestablished. Finally, when network congestion is detected by the receipt of an ECN message, the ATCP does nothing but forward the packet to TCP so that it can invoke its congestion control normally.

Layered Design

Layered design detection mechanism does not need the information from information from other layer or intermediate nodes. It tries to impose minimal demands (if any) on any host other than the sender or the receiver. This approach has comparative properties in ad-hoc networks: (i) It does not need any intermediate node in the network to notify the reason of packet losses, therefore is no need to modify the intermediate node in the network. (ii) It does not induce network overhead at the end-host and network, since intermediate node assistance mechanism usually use explicit approach to inform the end-host, which may introduce new overhead in the network.

Fixed RTO [31] relies on the idea that routing failure recovery should be accomplished in a faster fashion by the routing algorithm. As a result, it disables exponential back-off mechanism when two timeouts indicating the route failure happens and make TCP sender to retransmit at regular intervals instead of increasingly exponential ones. This approach is based on the assumption that any disconnection should be treated as a transitory period and exponential back-off can cause unnecessary long recovery delay. By doing so, it allows the TCP sender to retransmit at regular intervals instead of at increasingly exponential ones. In fact, the TCP sender doubles the RTO once and if the missing packet does not arrive before the second RTO expires, the packet is retransmitted again and again but the RTO is no longer increased. It remains fixed until the route is recovered and the retransmitted packet is acknowledged. The authors evaluated this proposal considering different routing protocols as well as the TCP selective and delayed acknowledgements options. They report that significant enhancements were achieved using fixed-RTO with on-demand algorithms, and only marginal improvements were noticed regarding the TCP options mentioned. Nevertheless, as stated by the authors themselves, this proposal is limited to wireless networks only, which makes it somewhat discouraging as interoperation with wired networks seems to be really necessary in the future.

TCP door [32] focuses on the idea that out-of-order (OOO) packets can happen frequently in ad-hoc network environment as a result of node mobility, and it might be enough to indicate link failure inside

the network. In this way, TCP door detects OOO events and responds accordingly. Since not only data packet but also ACK packets can experience OOO deliveries, TCP door implements a precise detection at both sender and receiver. To achieve this goal, one-byte option for ACKs and two-byte option for data packets are added in TCP options. For every data packet the sender increments its own stream sequence number inside the two-byte option regardless of whether it is a retransmission or not (standard TCP does not increment sequence number of retransmitted packets). Thus the receiver can precisely detect OOO data packets and notify the sender via a specific bit into ACK packet. In addition, the receiver increments its own ACK stream sequence number inside one-byte option for every "retransmitted" ACK, so that the sender can distinguish the exact order of every (retransmitted or not) sent packet. Therefore, the explained mechanisms provide the sender with reliable information about the order of the packet stream in both directions, allowing TCP sender to act accordingly. TCP door sender can respond OOO event with two mechanisms: temporarily disabling congestion control and instant recovery during congestion avoidance. In the former, TCP sender keeps its state variables constant for a while (T1) after the OOO detection. The rational is that such condition might be short (route change) not justifying the invocation of the congestion avoidance mechanism. In the latter, when an OOO condition is detected TCP sender checks if the congestion control mechanism has been invoked in the recent past (T2). If so, the connection state prior to the congestion control invocation is restored, since such an invocation may have been caused by temporary disruption instead of by congestion itself. In terms of evaluation, different scenarios combining all the mechanisms above mentioned were simulated. Also, the effects of the route cache property of DSR routing protocol on TCP door performance were considered. The main results showed that: Only sender detection mechanism (ACK OOO detection) should suffice. Both responses mechanisms showed to be important and instant recovery during congestion avoidance performed better than temporarily disabling congestion control. In general TCP door improved TCP performance significantly, 50% on average.

A simple receiver-based scheme [33, 34] is implemented at the receiver to distinguish congestion losses from corruption losses. This scheme works in the case where the last hop to the receiver is a wireless link and has the small bandwidth among all links in the connection path. With such mechanism, the receiver attempts to detect the real cause of the packet loss and inform the TCP sender to take the appropriate actions. Specifically, if the loss is a transmission error, the receiver can speed up the recovery and avoid shrinkage

of sender's congestion window.

TCP-Probing [6] grafts a probing mechanism into standard TCP. It uses "Probe Cycle" which consists of a structured exchange of "probe" segments between the sender and receiver to monitor the network condition. When a packet loss is detected, the sender initiates a probe cycle during which data transmission is suspended and only probe segments (header without payload) are sent. A lost probe or acknowledgment re-initiates the cycle, hence suspending data transmission for the duration of the error. When the probe cycle is completed, the sender compares the measured probe RTTs (Round Trip Time) and determines the level of congestion. The protocol allows for three distinct tactics in response to the nature of the error detected: Slow Start (for congestion detected by timeout), Fast Recovery (for moderated congestion detected by three ACKs), and Immediate Recovery (for congestion-free path). A fourth tactic (i.e., conservative recovery due to frequent link errors) is not included in the recovery strategy. The reason is that probing cycle can be extended when a probe or an acknowledgment is missing. Hence, a "probing device" models two properties: (i) it inspects the network load whenever an error is detected and rules on the cause of that error and, (ii) it suspends data transmission for as long as the error persists, thereby forcing the sender to adapt its data transmission rate to the actual conditions of the channel. Probing can be more effective than Reno and Tahoe when the sending window is not too small. The Immediate Recovery mechanism of TCP-Probing avoids the Slow Start and/or the congestion avoidance phase of Tahoe and Reno. In this case, Probing immediately adjusts the congestion window to the recorded value prior to the initiation of the probe cycle. When congestion is indicated, the protocol complies with the congestion control principles of standard TCP.

TCP-real [35] proposes modifications to (i) enhance the real-time capabilities of the protocol over wired/wireless networks and (ii) tackle the problem of asymmetry by decoupling the size of congestion window from the timeout. It is receiver oriented and uses wave pattern for detecting errors at the receiver. In this mechanism, congestion window size is included in the TCP header for the receiver to communicate with the sender to direct the sender's congestion control. The receiver measures the number of successfully delivered segments within a wave and the wave delivery time respectively, and estimate the level of loss and changes in current conditions. A wave is a fixed 3 pattern of data exchange between sender and receiver that enables the receiver to measure the perceived level of congestion based on the time required for a wave to be delivered. The wave delivery time is the time difference between the reception of the first and the

last segment of the wave and it could be much smaller than the RTT. Congestion control in TCP-Real has therefore two additional properties: (i) it can avoid unnecessary congestion window adjustments due to path asymmetry, and (ii) it can determine the level of loss and jitter with better precision since the size of the sending window is known to the receiver (i.e., the wave pattern).

TCP-Westwood (TCPW) [36, 37, 38] is a simple modification of the TCP source protocol stack which allows the source to estimate the available bandwidth, and to use the bandwidth estimation to recover faster, thus achieving higher throughput. TCP Westwood exploits two basic concepts: the end-to-end estimation of the available bandwidth, and the use of such estimate to set the slow start threshold and the congestion window. TCPW does not require any intervention from network layer or proxy agents. TCPW source continuously estimates the packet rate of the connection by properly averaging the rate of returning ACKs. The estimate is then used to compute the "Permissible" congestion window and slow start threshold to be used after a congestion episode is detected, that is, after three duplicate acknowledgements or after a timeout. The rationale of this strategy is simple: in contrast with TCP Reno, which simply halves the congestion window after three duplicate ACKs, TCP Westwood (TCPW) attempts to make a more "informed" decision. It selects a slow start threshold and a congestion window that are consistent with the effective connection rate at the time congestion is experienced. The "Key innovation" of TCPW is to use the bandwidth estimate "directly" to drive the window, instead of using it to compute the backlog. The rationale is that if a connection is currently achieving a given rate, then it can safely use the window corresponding to that rate without causing congestion in the network.

Freeze-TCP [15] is another inactive method. It avoids timeouts at the sender during handoffs since a timeout shrinks the sending window to a minimum in all TCP versions. To this end, Freeze-TCP exploits the ability of the receiver to advertise a window of zero. The motivation of this paper is that, originally, the TCP protocol is conservative on the errors. It assumes all the errors are due to the congestion. Error recovery schemes such as the congestion window adjustment, the acknowledgement strategy, the timeout mechanism as well as other factors (e.g., receiver advertised window, slow start threshold) all contribute to the efficiency of the recovery process. The idea is that if the protocol is able to distinguish the nature of the error, the recovery strategy can be more aggressive. More precisely, the regular course of action of the recovery strategy is to shrink the congestion window and extend the timeout period due to congestion, or

freeze the window and timeout upon drops due to non-congestion error.

A receiver-based rate estimation is used in [39] for the proactive detection of incipient congestion in rate-based protocols. It calculates the receiving rate in receiver side instead of sender side (Usually, we use ACK to calculate received rate in sender side). The receiver can infer the average of the sending-rate over the measurement interval from the value of X_i contained in the TFRC (TCP-Friendly Rate Control) data packet: Let n denotes the number of packets received during the time interval T_{recv} preceding the arrival of the most recently received packet. Let X_1, ..., X_n denote the sending-rates reported in these n packets. For each data packet received, compute $X_{send} = \frac{n}{\sum_{i=1}^{n} \frac{1}{X_i}}$;$X_{recv} = \frac{n}{T_{recv}}$; if $X_{send} - X_{recv} > \epsilon$, then this is an indication of the congestion. By matching the actual and expected receiving-rate, congestion can often be detected before packet losses occur. Simulation results show that this modification can help in avoiding packet losses and in stabilizing the transmission rate quicker at session start-up. The authors also point out that the same principle can be used to avoid packet losses in other situations as well, such as the adaptation to rapidly changing link characteristics in wireless systems.

TCP Santa Cruz [40] replaces the round trip delay measurements of TCP with estimations of delay along the forward path, and uses an operating point for the number of packets in the bottleneck.

2.2 Data Dissemination Protocols in Wireless Sensor Networks

Several mechanisms have been proposed in the literature to address the data dissemination problem in sensor networks. We classify them as *all-to-all* data dissemination, in which all sensor nodes distribute their data to all the other sensor nodes in the network; and *some-to-some* data dissemination, in which some sensor nodes only disseminate their data to those sensor nodes that are interested in the data.

The conventional all-to-all data dissemination protocols are flooding and gossip. In flooding, each node sends data it receives to all its neighbors, except the neighbor that it received the data from. It is the fastest dissemination algorithm with a distribution speed of $O(d)$, where d is the diameter of the network. Gossip uses randomization, in which each node in the network only forwards data to a randomly selected neighbor. It disseminates data the slowest. The fastest possible distribution rate of gossip is one node/round.

Both flooding and gossip are relatively straightforward to implement; however, there are some deficiencies with both these two protocols. Flooding suffers data implosion and overlapping problem that a node

always sends data to its neighbor no matter if it already has it or needs it [10] [11]. Nodes in the network may get multiple data packets from multiple paths and even incur flooding storm problem resulting in high energy consumption. Gossip can mostly avoid the implosion problem with a very slow dissemination rate. However, while disseminating data to all nodes in the network, it is possible that gossiping nodes forward data back to the sender resulting in redundant data transmission [11] and energy wastage. Therefore, they may not meet energy-conserving and timeliness requirements of sensor networks.

To overcome the deficiencies of flooding and gossiping, several all-to-all data dissemination protocols were proposed. They try to complement the data dissemination protocol with application layer methods that are data centric. The objective of these protocols is to minimize the transmission of redundant data by taking the data semantics of applications into account.

SPIN (Sensor Protocol for Information via Negotiation) [10] [11] uses high-level data descriptors called meta-data in negotiation to determine if a node needs the data prior to real data exchange. Before initiating data transmission, a source node starts the meta-data transfer by sending an advertisement packet (ADV). Data packet (DATA) is sent to only those nodes which send back a request packet (REQ). Meta-data packets are much smaller than data packets in this three-way handshake. Therefore, compared to network overhead introduced for meta-data transfer, it conserves more energy by reducing redundant data packet transmissions.

An enhancement of SPIN, SPMS (Shortest Path Minded SPIN) [8] [9] assumes that each sensor node in the network can operate at multiple power levels. The source node defines its maximum transmission range as a zone, and uses meta-data to advertise the availability of data (ADV) using maximum power level. The remainder of the negotiation and data transfer (REQ and DATA) use multiple hop transmission via the shortest path using the Bellman Ford algorithm. Since SPMS relies on the relay nodes for data delivery, it is resilient to intermediate node and link failures.

LAF (Location-Aided Flooding) [41] is also an information dissemination protocol based on a variant of classic flooding (modified flooding). It uses location information to partition the network into virtual grids. Sensor nodes relate themselves with a virtual grid based on its location and divided into groups of gateway nodes and internal nodes. Gateway nodes forward the packets across virtual grids; internal nodes forward the packets within a virtual grid. LAF reduces the number of redundant transmission by modified

flooding that adds a special field in packet header called node list that containing the ides of all the nodes that already have the packet. Hence, avoid forwarding packet to those nodes is unnecessary.

Deluge [42] provides quick reliable dissemination of large data objects over a multi-hop, wireless sensor network. Each node occasionally advertises the most recent version of the data object it has available to its neighbors. The node that receives an advertisement of older version will respond its object profile of new version. From the object profile, the node determines which portions of the data need updating and requests them from the node that sending object profile. This process continues till all the nodes get new version of data. The density-aware and epidemic properties help to provide reliable data propagation in network.

INFUSE [43] is another reliable dissemination protocol for bulk data based on TDMA medium access layer. Although TDMA guarantees collision-freedom, unexpected channel errors (e.g., message corruption, varying signal strengths, etc) can cause random messages losses. INFUSE consider two recovery scheme that use implicit acknowledgements (received by listening to the transmissions of the successors of a sensor) to recover from lost messages.

Usually *some-to-some* dissemination utilizes the publish and subscribe model to distribute data to some of the nodes in the network: sink nodes have a specific subscription profile to indicate which data they are interested in, and source nodes only send corresponding data to those sink nodes that are interested in them.

Directed Diffusion (DD) [44] is a data-centric protocol. The data generated by sensor nodes is named by attribute value pairs. The sink node requests the data by periodically broadcasting an interest for the named data. Each node in the network that received the interest will set up gradients to its neighboring nodes from which it receives the interests. The intermediate nodes can cache or transform data. Once the source or intermediate node observes that the interest matches the available data, it will send data towards the sink along multiple paths. Then the sink will reinforce one or a small number of these paths to receive the rest of the data.

TTDD (Two-Tier Data Dissemination) [45] studies data dissemination in a large-scale sensor network from potentially multiple sources to potentially multiple mobile sinks. It assumes that sensor nodes are stationary and location-aware, and sinks could be mobile. TTDD builds and maintains a grid structure, and sets up forwarding information throughout the sensor network for each source node. Only sensor nodes located at or closest to the grid points (*dissemination nodes, DN*) need to forward the data. An interest from

a sink traverses two tiers to reach the source. One tier is the grid square of the sink called (cell) and the other one is the DN at grid points. The sink floods the interest within its own cell. When the nearest DN receives the interest, it forwards the interest to its adjacent DNs. This process continues until the query reaches the source or one of the DNs that has the corresponding data. The data path is established in interest-propagation period.

GRAB (Gradient Broadcast) [46] addresses the problem of robust data forwarding to a sink using unreliable sensor nodes with error-prone wireless channels. The sink builds and maintains a cost field by broadcasting the advertisement message containing its initial cost. Each intermediate node that hears the advertisement will calculate the receiving cost of the message. Then each node will keep the minimum cost for forwarding a packet from itself to the sink. When source sends the message to the sink, each message carries a "credit". The intermediate node which has a cost not greater than the "credit" plus the cost of the source will forward the message. Therefore, data can be forwarded along a band of interleaved mesh (multiple paths) from each source to the sink. The amount of the credit determines the width of the mesh, and controls the degree of robustness and overhead.

Bokareva et al. [47] conducted a simulation comparison of DD [44], TTDD [45], and GRAB [46] using the ns-2 [48] simulator. They studied average energy consumption, routing overhead, and packet delivery ratio. They observed that GRAB produces least routing overhead, while DD consumes least energy than other two protocols. TTDD and DD have very similar data delivery ratio close to the ideal one, whereas GRAB delivers on average six times more redundant data packets. All these protocols rely on a significant number of statically configured parameters. Because DD and GRAB use flooding to build up gradient and cost field for each node in the network, they provide little scalability and mainly target static networks only. TTDD relies on a priori geographical knowledge for routing.

Compared to deterministic algorithm, randomized algorithm naturally featured as robustness, simplicity and scalability. These properties are important for wireless sensor networks that characterized as resource constraints and high density. Therefore, recent year, many researchers shift there emphasis from deterministic algorithm to randomized algorithm while designing protocols for sensor networks [49]. Rumor Routing [50] is a logical compromise between flooding queries and flooding event notifications. When an interest is generated, it can be sent on a random walk instead of flooding until it finds the event path. As soon as

the interest discovers the event paths, it can be routed directly to the event. If the path cannot be found, the application tries re-submitting the query, or as a last resort, flooding it.

From the energy saving point of view, besides designing energy efficient protocols that can reduce the network operations, therefore reduce the sensors' energy expenditure, a widely employed technique is to place nodes in a sleep mode. In sleep mode, some parts of the sensor circuitry are turned off, such as transceiver etc. Hence, the sensors can save significant amount of energy by not transmit and receive data. However, there is a trade-off between node energy saving and the network performance in terms of throughput and data delivery delay. [51] develop an analytical model - Markov model of a sensor network whose nodes may enter a sleep mode, therefore, enables to explore this trade-off and to investigate the network performance as the sensor dynamics in sleep/active mode vary. [52] try to determine when certain nodes can sleep in order to reduce system-level energy consumption by developing a sleep discipline that allows nodes to minimize energy consumption by sleeping for the maximum amount of time. TD-DES (Topology-Divided Dynamic Event Scheduling) [53] organizes network as an event dissemination tree. Each node subscribes to the event type they are interested in. The root of the tree creates a data dissemination schedule that dynamically allocates and multiplexes upstream and downstream time slots for each event type and propagates it throughout the tree. Since each node can power off its radio while not transmitting data, power consumption is reduced. The event dissemination schedule can be determined in both centralized and distributed fashions.

CHAPTER THREE

TCP ENHANCEMENT (TCP-MANET) FOR WIRELESS AD-HOC NETWORKS

TCP is the most popular transport layer protocol on the Internet, which provides reliable data communications. It is simple, efficient, and operates very well in wired networks. TCP packets are cumulatively acknowledged as they arrive in order, with out of order packets causing duplicate acknowledgements.

As a trade-off of simplicity and efficiency, TCP gave up of having a mechanism to correctly and rapidly detect the nature of the error [5, 6, 3, 7]. Although it can detect packet loss by duplicated acknowledgement (fast retransmission) and timeout, TCP only assumes that all the packet losses are due to congestion and lower layer can handler all other errors. However, this assumption is not true in wireless networks that suffers high packet loss rate due to its highly dynamic property and relative persistent nature (i.e. fading channel, prolonged and frequent burst error). Therefore, TCP protocol usually has some undesirable patterns of behavior, resulting in a performance degeneration of the throughput in wireless ad hoc networks. For example, when there are random or short burst link error occurrences that lead to packet losses, the TCP will invariably interpret these events as resulting from congestion, reduce the window size, and reduce the sending rate. After then, the sender will apply a conservatively increase to its reduced window size. Bandwidth and opportunities for error-free transmissions are wasted during this phase inevitably, and the throughput will be reduced. In addition, in ad-hoc networks the node can move freely, the broken route would trigger exponential back-off mechanism in TCP protocol, which unnecessarily freezes the TCP sender.

TCP-Manet aims at adding detection functionality in TCP that can determine nature of the error and then invoke corresponding recovery scheme, hence improving the performance of TCP over wireless ad-hoc networks. When thinking of adding error detection component, we faced three design options:

- **At lower layer**. The error detection component is implemented in lower layer. In this case, TCP protocol does not need any modifications, and lower layer can handle those errors. [54, 55] provide an unobtrusive monitoring mechanism that offline monitors system logs and activity to detect wireless link failure, misroute packets. However, without end-to-end information from transport layer protocol, lower layer becomes week at the detection of some errors such as network misbehavior. In addition, the purpose of error control in lower layer is only used to optimize the function of the higher layer. The

modification in lower layer might exhibit a conflict behavior with TCP's mechanism. For example, a retransmission attempt at lower layer might result in extending the RTT estimates of TCP and hence its timeout. If the lower layer can not have enough information to perform the error detection functionality, we can not implement the whole functionality of the error control in the lower layer [35].

- **At transport layer**. Transport layer protocol provides the error control functionality. However, because traditional TCP protocol is carefully tuned for wired networks, it only has limited information about the network, such as RTT and acknowledgement. Therefore, it can not provide an error control mechanism that can detect errors rapidly and efficiently alone.

- **Using a cross layer design**. To provide a comprehensive error detection mechanism, a layered mechanism should be deployed to enforce cooperation between transport layer and lower layer to provide error control. Using this method, TCP can gather necessary information and detect errors at transport layer. Thus implementing an error detection component at transport layer with more precise and accurate information from the lower layer will be a good choice.

In following sections we will introduce the detection mechanism of TCP-Manet. First we describe how TCP-Manet detects wireless error by monitoring current connection status. Then, we present new congestion control and avoidance algorithm in TCP-Manet. Finally, we present how TCP-Manet detects network misbehavior.

3.1 Wireless Error Detection

The foundation of traditional TCP's congestion control is the principle of "conservation of packets" [56]. The "conservation of packets" means for a connection "in equilibrium": a new packet isn't admitted into the network until an old packet leaves, which is indicated by an arrival of ACK (acknowledgement). The basic idea of congestion control mechanism is that each TCP connection measures available bandwidth along the path between source and destination while in operation, and adjusts its date injection rate to make sure the flow make full use of the available bandwidth but not exceed it.

Consider a network path P is a sequence of H hops from the source S to the destination D. The link

capacity of each link i can transmit data with rate of C_i bps. The end-to-end capacity C is defined as "the maximum rate that the path can provide to a flow, when there is no other traffic in path P".

$$C \equiv min_{i=1...H} C_i \tag{3.1}$$

Suppose the average utilization of link i during time $(t_0, t_0 + \tau)$ is $u_i^\tau(t_0)$ with $0 \leq u_i^\tau(t_0) \leq 1$. The end-to-end available bandwidth A is defined as the maximum rate that the path can provide to a flow without reducing the rate of the rest of the traffic in path P [57].

$$A^\tau(t_0) \equiv min_{i=1...H} C_i(1 - u_i^\tau(t_0)) \tag{3.2}$$

Available bandwidth varies with time and exhibits high variability in a wide range of timescales. It is hard to measure the exact available bandwidth of the path while TCP is in operation. Therefore, traditional TCP uses probing to explore the available bandwidth, which is proved to be simple, effective, and practical. It linearly increases the congestion window size to intentionally create packet loss as a signal of congestion. Once there is a packet loss, sender assumes current sending rate has exceeded available bandwidth, reduce the sending rate by decreasing the congestion window size.

An alternative congestion control technique for TCP is end-to-end delay-based congestion avoidance algorithms (DCA). DCA is originally described by Jain [58], and is best represented by TCP-Vegas and Dual [59, 60], [61] focus on the DCA algorithm in high speed network. Unlike traditional TCP that is inactive and uses packet loss as an indicator that current sending rate has exceeded the available bandwidth, DCA introduces a preventive idea. It monitors some implicit feedback information such as increased packet round-trip times (RTTs), decreased throughput etc. to deduce the available bandwidth, and then determine the optimal sending rate to react to the increases in RTT in an attempt to avoid network congestion before it becomes significant.

TCP-Manet is a combination of two techniques. The packet loss is detected by duplicate packets and timeout. Because packet losses could be due to many reasons such as wireless link error, congestion etc.,

we could not simply assume current congestion window size reaches the available bandwidth. Therefore, we also introduce the preventive idea to monitor the network load. Unlike DCA that monitors the RTT, TCP-Manet monitors the trend of the "power" metric. Metric *power* is defined as the ratio of throughput over delay [62, 63].

Fig. 3.1 shows hypothetical graphs of round trip time, throughput and power as network load increases [1]. When the network load is small, increasing the load results in a comparable increase in the network throughput. After the load reaches the network capacity, throughput does not increase. If the load increases any further, the queues build up, potentially resulting in packets being dropped, and throughput dropping to zero. The round trip time behaves in a similar fashion. At first the round trip time has only a small increase as the load increases. When the queue starts to build up, the round trip time increases linearly. As the queues start to overflow, the round trip time increases to an extremely large value. As for power, when the network load is small, power increases. After the load reaches the network capacity, power decreases. The point at which throughput approaches zero and the round trip time approaches infinity is called the point of congestion collapse. The point with maximum throughput and minimal round trip time is called the point of knee.

By monitoring the trend of "power", TCP-Manet could gain some enhanced information to help itself be capable of telling if current sending rate (congestion window size) has exceeded the available bandwidth. If the trend of the "power" is increasing upon a packet loss, this loss probably is due to wireless error. The congestion window size should remain the same.

To compute "power", we use ACKs to compute throughput and delay. More precisely, the sender uses the following information: (i) the ACK arrival times, and (ii) the increment of data delivered to the destination. For instance, assume an ACK is received at the source at time t_k, notifying that all the segments before sequence number s_k have been received at the TCP receiver. That is d_k bytes have been received at the TCP receiver between interval t_{k-1} and t_k, where t_{k-1} is the time the previous ACK was received. The measured sample bandwidth used by that connection should be $b_k = d_k/(t_k - t_{k-1})$. Let $\Delta t_k = t_k - t_{k-1}$, then $b_k = d_k/\Delta t_k$. The sampled round trip time (RTT) is the round trip time for segment of sequence number s_k.

The pseudo-code is as follows:

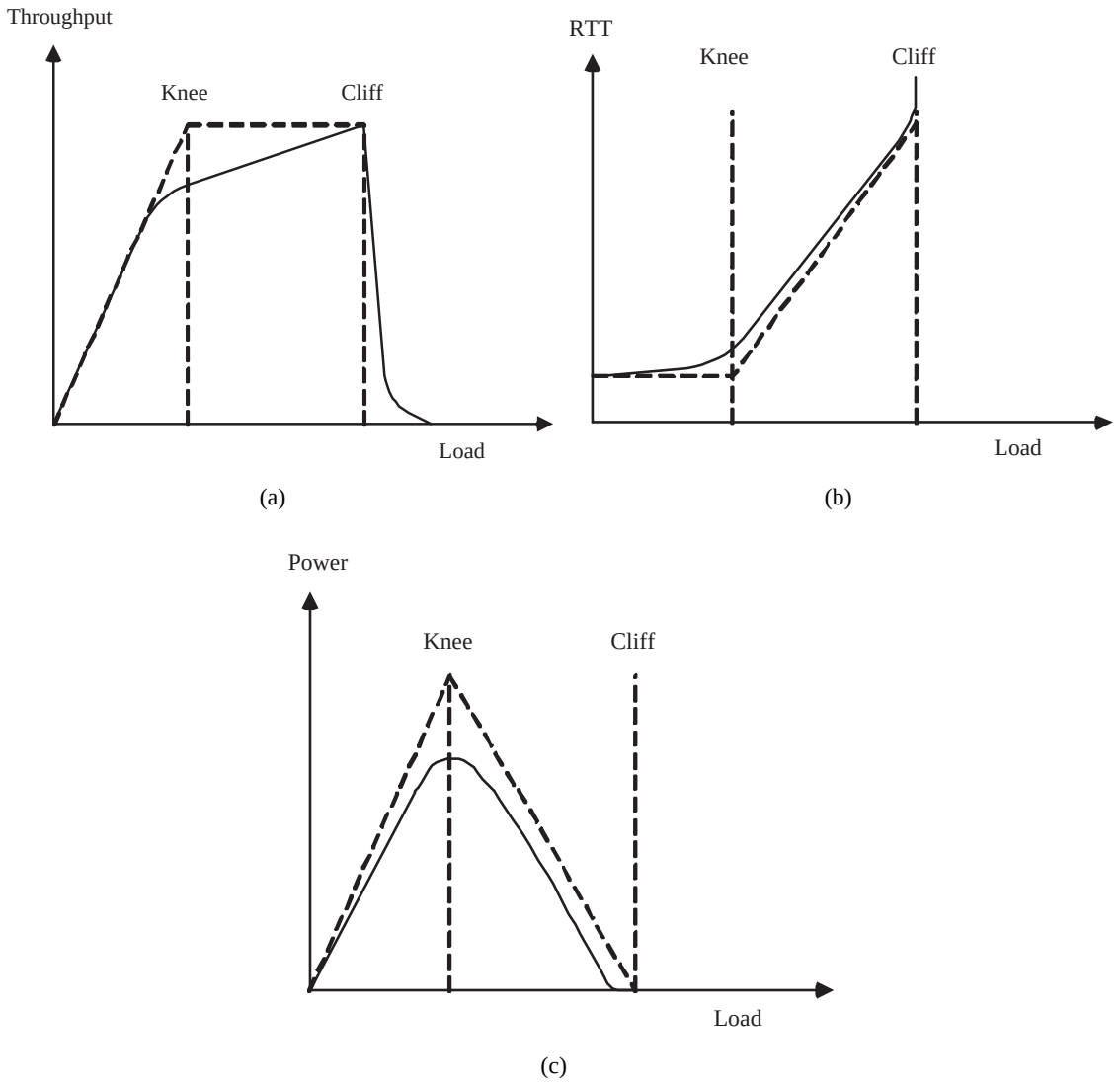


Figure 3.1: Network performance varying the load [1]. (a) Throughput vs. Load; (b) Round Trip Time (RTT) vs. Load, (c) Power vs. Load.

```
1.  if (ACK is received)
2.  {
3.      inverval[k] = now - last_sending_time;
4.      deliverd[k]= ACK_seq_no - last_ack_seq_no;
5.      rtt[k] = now - packet_sending_time;
6.      bw[k] = delivered[k] / interval[k];
7.      power[k] = bw[k] / rtt[k];
8.  }
9.
```

To compute the trend of power while the TCP is in operation, we use the algorithm proposed by Jain et al [57]. Suppose that the (relative) powers of a particular stream are P^1, P^2, ... P^k. First, we partition these measurements into $\Gamma = \sqrt{K}$ group of Γ consecutive powers. Then compute the median power $\hat{P^i}$ of each group. We get set $\hat{P^i}, i = 1, 2, ...\Gamma$. To check if this stream is in an increasing trend, the pairwise comparison test (PCT) metric of a stream is

$$S_{PCT} = \frac{\sum_{k=2}^{\Gamma} I(\hat{P^k} > \hat{P^{k-1}})}{\Gamma - 1} \tag{3.3}$$

Where $I(X)$ is one if X holds, and zero otherwise. PCT measures the fraction of consecutive power pairs that are increasing, and so $0 \leq S_{PCT} \leq 1$. If the powers are independent, the expected value of S_{PCT} is 0.5. If there is a strong increasing trend, S_{PCT} approaches one. In our current algorithm, if $S_{PCT} > 0.5$, power metric shows an increasing trend.

As we described above, upon packet loss, if current trend of power is not decreasing, TCP sender considers it as a wireless error and just retransmits the packet. Otherwise, TCP sender enters in congestion avoidance. TCP-Manet uses a new congestion control and avoidance algorithm. It reduces the congestion window size gradually rather than halve it like traditional TCP.

3.2 Congestion Control and Avoidance

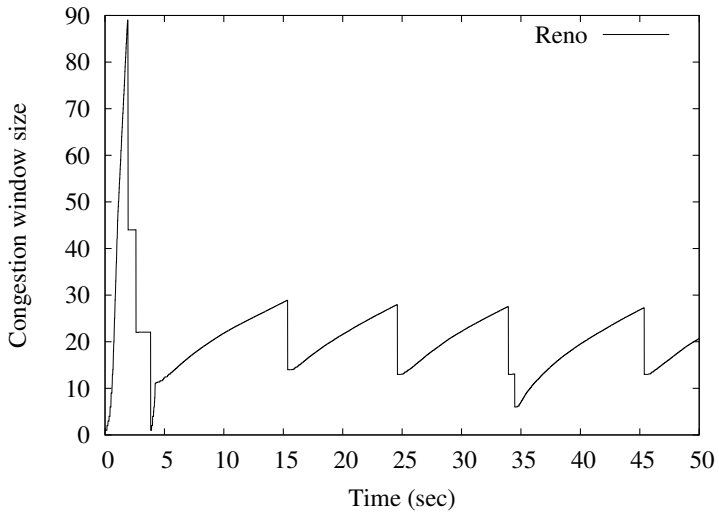
Conventional TCP uses AIMD (Additive Increasing Multiplicative Decreasing), which is based on the observation that queue length will increase exponentially under congestion condition. This aggressive window size reduction results in a saw tooth transmission pattern shown in figure 3.2(a). Therefore, TCP can not fully make use of the available bandwidth, especially, results an inferior performance in wireless networks. In wireless networks, random losses will increase the saw tooth pattern and reduce the total throughput shown in figure 3.2(b).

TCP-Manet deploys a new congestion control and avoidance algorithm to flatten the saw tooth pattern, in other word reduce the fluctuation to make bandwidth to be fully occupied.

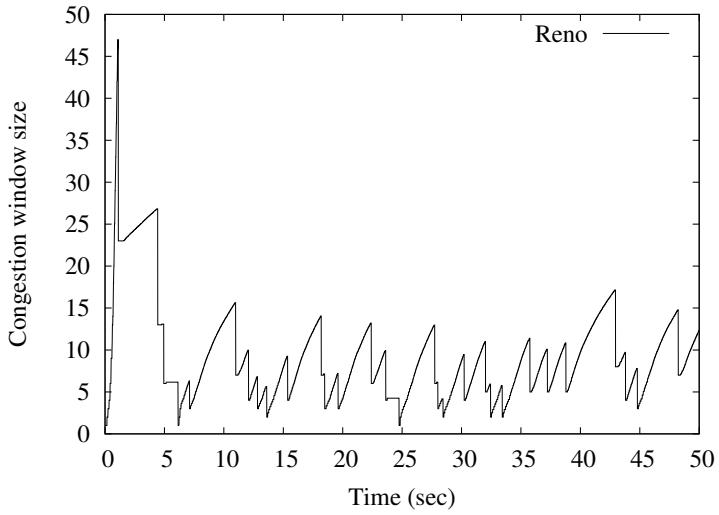
We define an "iteration" as the time between two packet losses with an increasing power trend. Within each iteration, TCP-Manet reduces the window size exponentially till half of the current window, say 0, 1, 2, ...2^i, ... $w/2$, here w is current window size. Let l denote size of the windows that should be reduced. If TCP-Manet sender detects a packet loss and "power" is in an increasing trend. l will be set to 0. When the next packet loss happens, if the "power" is still in an increasing trend, l will remain 0. Otherwise, we get $l = 2^1$, window size will be reduced to $w - 1$. If the next packet loss happens under an increasing power trend, l will be reset to 0. Otherwise $l = 2^1$, ... till $min(2^i, w/2)$.

TCP probes the available bandwidth by linearly increasing the congestion window size. Therefore, with the increase of the window size, the probability of congestion loss will increase. Assume sender detect a packet loss while power is in an increasing trend. The sender will just retransmit the packet without decreasing its window size. Under this condition, the probability that next packet loss is congestion loss with decreasing power trend will be high. Then sender should decrease more of its window size. Even this loss is a wireless link loss, because sender does not reduce its window size by half, it still can alleviate throughput degradation. Finally, while window size keeps decreasing exponentially, the traffic load in the network will decrease. Sender should be able to detect a packet loss with increasing power trend and reset l to 0. By operating window size conservatively and gradually, TCP-Manet can have higher throughput than traditional TCP.

The pseudo-code is as follows:



(a)



(b)

Figure 3.2: TCP-Reno congestion window size vs. time. a) Error rate = 0 (b) Error rate = 1%

```
1.  if (three duplicated packets) {
2.      // check the power trend between two packet drops
3.      trend = checkPowerTrend();
4.      if (trend == 1) { // increasing power trend
5.          iter = 0; // reset dropping size
6.      } else {
7.          if (2^iter < w/2) {
8.              iter++;
9.              reduce congestion window by 2^iter
10.         } else {
11.             reduce congestion window by w/2;
12.         }
13.     }
14. }
```

3.3 Selfish Nodes Detection

In this section, we describe how TCP-Manet detects network misbehavior using a cross layer design.

In traditional wired network, all the communication protocols in the protocol stacks such as medium access control (MAC) protocol, the routing protocol, and the transport layer protocol, were designed under the assumption that all hosts would follow the given specification. However, in a highly dynamic wireless ad hoc network, the host can diverge from the specification for its own purpose, which would have bad impact on global system performance. For instance, in wireless ad-hoc networks, some hosts may refuse to forward packets for other hosts to save its energy level. From this host itself point of view, it can save its energy. However, from the network system point of view, the performance of the network will degrade.

The nodes in an ad hoc network can be classified as [64]:

- Cooperative nodes, which comply with the protocol specification.
- Inactive nodes, which are unintentionally misconfigured or constrained.

- Selfish nodes, which ignore other nodes to optimize their own gain.
- Malicious nodes, which inject false information and/or remove packets from the network.

A selfish node is defined as a node not taking part in packet forwarding. Although a selfish node does not forward any data packets for other nodes except himself to maximize their own gain, it does need to assist the routing discovery to maintain an up to date routing table. This is because the selfish node itself need send and receive packet for its own purpose. This feature makes selfish node usually hard to be detected especially isolated. The existence of selfish node along the route triggers the exponential back-off mechanism, starves the TCP connection, and finally results in connection restoration.

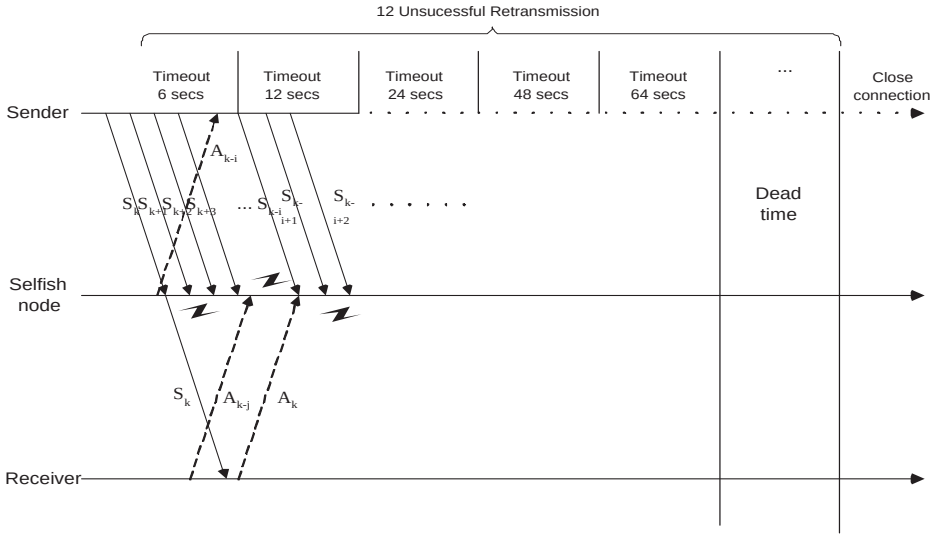


Figure 3.3: Impact of selfish nodes in TCP connection.

Figure 3.3 illustrates how a selfish node lead the TCP sender to a long idle period that is called "dead time", [19] and subsequently to the connection reset. Suppose a selfish node in the figure drops all the packets routed from and to the TCP sender, shortly after it forwards acknowledgement A_{k-i} and segment S_k. This places the sender/receiver in a status in which they can not receive any further ACK/segment from the receiver/sender, and the exponential back-off mechanism is triggered. At the first timeout that is a typical initial Retransmission Time-out (RTO) set to 6s, the sender retransmits from segment S_{k-i}. It then continues retransmitting until the timeout is doubled up to the limit of 64s (the maximum allowed timeout). If the new route is not reestablished between sender and the receiver, after 12 unsuccessful retransmissions, TCP sender

would consider the receiver is crashed or closed, give up retransmission and reset the connection. From the figure we can see that the TCP sender will have a dead time [19] which is the time duration from TCP sender setting its fifth timeout (64s) until it resets the connection. The dead time takes several minutes, during which TCP connection is frozen.

Some reputation-based systems [65] [66] have been proposed to deal with network misbehavior in ad-hoc networks. These approaches rely on direct network observation mechanisms so called watchdog. However, if a selfish node silently drops the packet, it is hard for network layer to detect it, since network layer has no information about end-to-end information about the flow. Therefore, it is necessary and possible for TCP to provide some assistance of detecting selfish nodes.

One of the important features of TCP-Manet is to identify and isolate the selfish nodes. The basic idea is similar to "traceroute" application. TCP-Manet sets a threshold as 4 timeouts. When 4 timeout occurs, sender starts to send some probe messages to identify the reason of timeouts. Sender first sends a probe message with a TTL (Time to Live) of 1 to the receiver. The first hop node along the path will handle the message by decrementing the TTL, discard the datagram, and send back the ICMP time exceeded message. Then the next message with a TTL of 2 is sent out from the sender. This continues until probe message can reach the destination that sends back an ICMP port unreachable message. If this occurs, the sender can conclude that there is no selfish node along the path. If TCP sender experiences a timeout for the probe message, and the sender node receives a "routing error message" during this period of time, the packet losses may due to the node mobility. Otherwise, we can conclude selfish node exists along the route.

This process is similar to traceroute. However, the purposes are different. In traceroute application, UDP diagrams are sent out to discover the route path from the source to the destination. In TCP-Manet, with the route information already in hand, sender needs to determine the existence of selfish node using the responses of the probe messages.

Besides, it is necessary for the sender to identify the suspicious selfish nodes along the path for isolation. As we described above, one of the characteristics of the selfish node is joining the route discovery and maintenance process. Without identifying the selfish nodes, even if TCP sender requests a new route for transmission, network layer may choose the same route as the current one. TCP-Manet considers two scenarios:

- Selfish node does not reply an ICMP error message to TCP sender

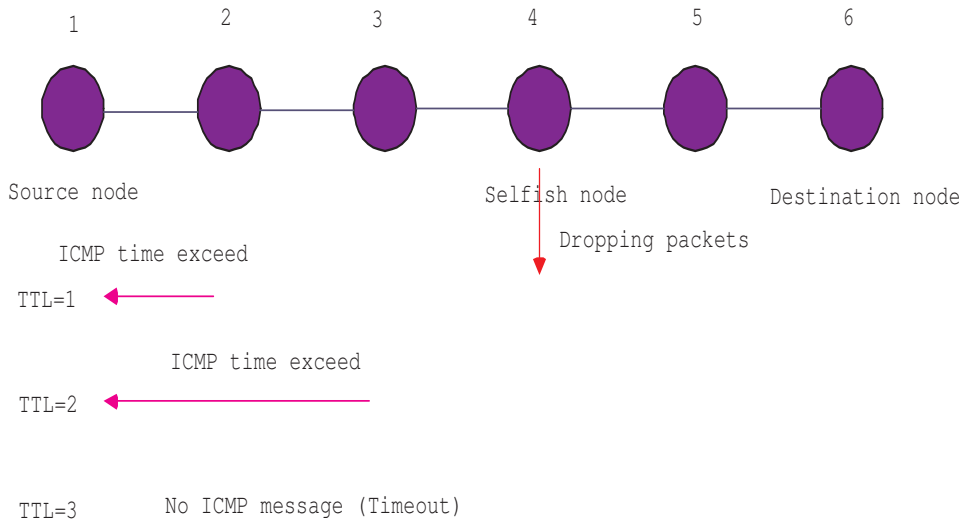


Figure 3.4: Case 1: Selfish node does not reply an ICMP error message to TCP sender.

Fig. 3.4 illustrates a connection with 6 nodes from sender to the receiver. Node 4 is the selfish node that drops all the packets. We assume that node 4 receives a probe message and does not send back an ICMP time exceed. Sender will receive ICMP time exceed packet from node 2 and 3, and get a timeout at node 4 (TTL = 3). In this case, sender does not know this timeout is because of node 3 dropping the probe message, or because of node 4 itself not replying the probe message.

- Selfish node replies an ICMP message to sender

  Fig. 3.5 illustrates the scenario that selfish node 4 drops the probe message and send back an ICMP message to the sender. The sender will get timeout on node 5. In this case, the sender can not determine this timeout is because of node 4 not forwarding the probe message, or because of node 5 dropping the probe message.

TCP-Manet chooses two suspicious selfish nodes for isolation – The last node from which sender received an ICMP error message, and the first node where sender encounters a timeout. Based on this strategy, for example, in scenario 1, we will choose node 3 and node 4 as suspicious selfish nodes; in scenario 2, we will choose node 4 and 5 as suspicious nodes. Then, TCP-Manet sender is able to request a new route which does not include the suspicious selfish nodes.

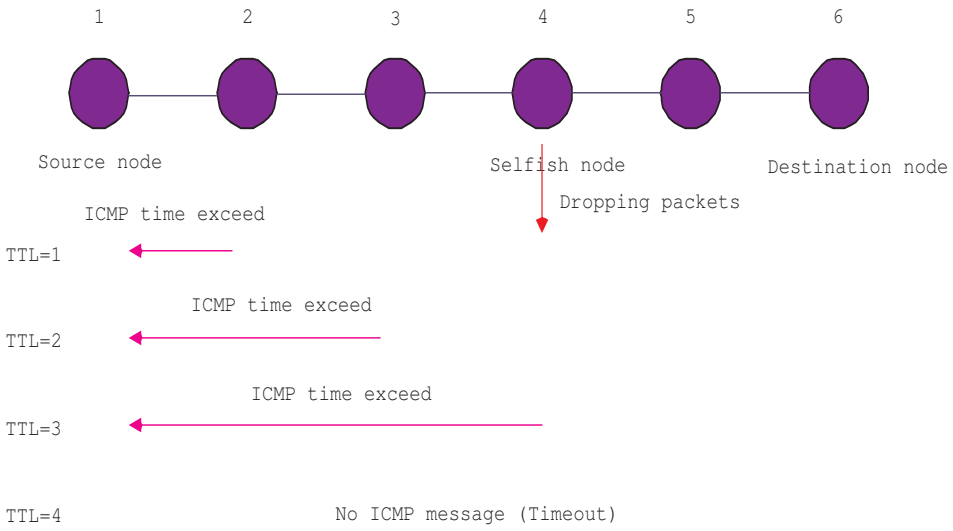


Figure 3.5: Case 2: Selfish node replies an ICMP message to sender.

3.4 Theoretical Analysis

In this section we give a theoretical analysis of TCP-Manet.

[67] gives the first comprehensive stochastic model of throughput for traditional TCP protocol.

$$\begin{aligned} B &= \frac{E[Y]}{E[A]} \\ &\approx \frac{1}{RTT\sqrt{2bp/3} + T_0 min(1, 3\sqrt{3bp/8})p(1+32p^2)} \end{aligned} \tag{3.4}$$

Where p is the packet loss rate; b is the number of packets that are acknowledged by an ACK in a round. b is usually 2, because many TCP receiver used delayed ACK, which sends one cumulative ACK for two consecutive packets received. RTT is the round trip time. T_0 is the initial timeout length. Throughput is measured by packet per unit time instead of bytes per unit time.

We extended the model in [67] to capture the features of TCP-Manet and developed the model in several steps corresponding to its operating regimes:

- First, we develop the model of a simplified TCP-Manet (TCP-simple) that only considers detection of wireless link error. TCP-simple determines wireless link loss by monitoring the trend of power metric,

described in 3.1. Upon receiving "three duplicated acknowledgements", if current power metric is in increasing trend, TCP-simple will keep the congestion window size unchanged. Otherwise, it will halve the windows size as the traditional TCP protocol.

- Then, we consider congestion control and avoidance algorithm in TCP-Manet described in 3.2. TCP-Manet uses a more conservative congestion avoidance and control algorithm than traditional TCP. Besides monitoring the trend of power metric to identity the wireless link error, it gradually decreases the congestion window size instead of using multiplicative decreasing algorithm.

- Finally, We consider selfish nodes detection in TCP-Manet described in 3.3.

3.4.1 System Parameters and Assumptions

We assume that there is no limit on the congestion window size. Like other papers [67], [68], [69], [70], [71], [72], we model TCP-Manet in terms of "rounds". A round starts with the back-to-back transmission of W packets, where W is the current size of the TCP congestion window size. Once all packets falling within the congestion window have been sent in this back-to-back manner, no other packets are sent until the first ACK is received for one of these W packets. This ACK reception marks the end of the current round and the beginning of the next round.

Besides the parameters described in [67], we also use (i) wireless link error probability (p_w); (ii)If the packet loss is due to wireless link error, we detect it with an increasing power trend with probability (p_1); (iii) If the packet loss is not due to wireless link error, we detect it with an increasing power trend with probability (p_2).

Let events A and B denote:

A = packet loss due to wireless link error

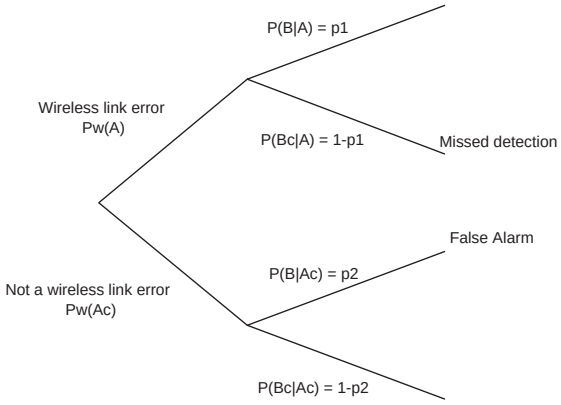
B = packet loss is detected with a presence of increasing power trend.

and their complements are:

A_c = packet loss is not due to a wireless link error

B_c = packet loss is detected without a presence of increasing power trend.

Fig. 3.6 gives the probabilities along the corresponding braches of the tree describing the sample space.



(a)

Figure 3.6: Tree structure of probability.

For each event of interest in corresponds to a leaf of the tree, its probability is equal to the product of the probabilities associated with the branches in a path from the root to the corresponding leaf. We get:

$P_{tp} = P(true\ positive) = P(A \cup B) = P(A)P(B|A) = p_w * p_1$

$P_{fa} = P(false\ alarm) = P(A_c \cup B) = P(A_c)P(B|A_c) = (1 - p_w) * p_2$

$P_{tn} = P(true\ negative) = P(A_c \cup B_c) = P(A_c)P(B_c|A_c) = (1 - p_w)(1 - p_2)$

$P_{md} = P(miss\ detection) = P(A \cup B_c) = P(A)P(B_c|A) = p_w * (1 - p_1)$

In following sections, we will develop a stochastic model of TCP-Manet in the presence of packet loss.

3.4.2 TCP-Simple

In TCP-Simple, sender detects wireless link loss by "three duplicate acknowledgements". Upon receiving "triple-duplicate" ACKs, if current power is in increasing trend, congestion window size will be kept the same. Otherwise, congestion window size will be half of the current size like traditional TCP and its variants.

We model the throughput assuming the packet loss are of "triple-duplicate" ACK in terms of p, p_w, p_1, p_2. As in [67], we define a TD period (TDP) (Triple Duplicate Period) to be a period between two TD loss indications. For ith TD period Y_i is the number of packets sent in the period, A_i the duration of the period, and W_i the window size at the end of the period. During the i-th TD period, the window size increases between $f(W_{i-1})$ and W_i. Because the increase is linear with slope $1/b$, we have:

$$w_i = f(w_{i-1}) + X_i/b \tag{3.5}$$

$$E[W] = E[f(W)] + E[X]/b \tag{3.6}$$

$$\begin{aligned} E[W] &= (p_w p_1 + (1 - p_w p_2)E[W] + ((1 - p_w)(1 - p_2) + p_w(1 - p_1))E[W]/2 + E[X]/b \\ &= (p_w p_1 + p2 - p_w p_2 + 1)E[W]/2 + E[X]/b \end{aligned} \tag{3.7}$$

$$\begin{aligned} Y_i &= \sum_{k=0}^{x_i/b-1} f(w_{i-1}) + k)b + \beta_i \\ &= f(w_{i-1}X_i + \frac{X_i}{2}(\frac{X_i}{b} - 1) + \beta_i \\ &= \frac{X_i}{2}(2f(w_{i-1} + \frac{X_i}{b} - 1) + \beta_i \\ &= \frac{X_i}{2}(f(w_{i-1}) + W_i - 1) + \beta_i \end{aligned} \tag{3.8}$$

$$E[Y] = \frac{E[X]}{2}(E[W](p_w p_1 + p_2 - p_w p_2 + 1)/2 + E[W] - 1) + \beta_i \tag{3.9}$$

Because we have:

$$(p_w p_2 - p_2 - p_w p_1 + 1)E[W]/2 = E[X]/b \tag{3.10}$$

$$E[Y] = \frac{1-p}{p} + E[W] \tag{3.11}$$

So we derive $E[W]$ as follows. Here β_i is the number of packets in the last round, which is uniformly distributed between 1 and w_i, and $E[\beta] = E[W]/2$.

$$\begin{aligned} \frac{1-p}{p} + E[W] = & \quad \tfrac{E[X]}{2} \quad (\frac{E[W]}{2}(p_w p_1 + p_2 - p_w p_2 + 1) + E[W] - 1) \\ & + \quad \tfrac{E[W]}{2} \end{aligned} \tag{3.12}$$

Let $\gamma = p_w p_1 + p_2 - p_w p_2$

$$\frac{1-p}{p} + E[W] = \frac{b(1-\gamma)}{4} E[W](\frac{E[W]}{2}(\gamma+1) + E[W] - 1) + \frac{E[W]}{2} \tag{3.13}$$

$$\frac{b(1-\gamma)(\gamma+3)}{8} E[W]^2 - \frac{b(1-\gamma)+2}{4} E[W] - \frac{1-p}{p} = 0 \tag{3.14}$$

$$\begin{aligned} \Delta &= \frac{(b(1-\gamma)+2)^2}{16} + 4\frac{b(1-\gamma)(\gamma+3)}{8}\frac{1-p}{p} \\ &= \frac{b^2(1-\gamma)^2 p + 4b(1-\gamma)p + 4p + 8b(1-\gamma)(\gamma+3)(1-p)}{16p} \\ &= \frac{b^2 p - 2b^2\gamma p + b^2\gamma^2 p + 4bp - 4b\gamma p + 4p - 8b\gamma^2 - 16b\gamma + 24b + 8bp\gamma_1 6bp\gamma - 24bp}{16p} \\ &= \frac{b^2(1-\gamma)^2 + 4}{16} + \frac{b(\gamma-1)(2\gamma+5)}{4} + \frac{b(1-\gamma)(\gamma+3)}{2p} \end{aligned} \tag{3.15}$$

$$
\begin{aligned}
E[W] &= \frac{\frac{b(1-\gamma)+2}{4}}{\frac{b(1-\gamma)(\gamma+3)}{4}} + \sqrt{\frac{b^2(1-\gamma)^2+4}{b^2(1-\gamma)^2(\gamma+3)^2} + \frac{4b(\gamma-1)(2\gamma+5)}{b^2(1-\gamma)^2(\gamma+3)^2} + \frac{8}{pb(1-\gamma)(\gamma+3)}} \\
&= \frac{1}{\gamma+3} + \frac{2}{b(1-\gamma)(\gamma+3)} + \\
&\quad \sqrt{\frac{1}{(\gamma+3)^2} + \frac{4}{b^2(1-\gamma)^2(\gamma+3)^2} + \frac{4b(\gamma-1)(2\gamma+5)}{b^2(1-\gamma)^2(\gamma+3)^2} + \frac{8}{pb(1-\gamma)(\gamma+3)}}
\end{aligned}
\tag{3.16}
$$

$$
E[X] = \frac{(1-\gamma)b}{2} E[W] \tag{3.17}
$$

As in [67], let r_{ij} denote the duration (round trip time) of the j-th round of TDP_i. Then the duration of TDP_i is $A_i = \sum_{j=1}^{X_i+1} r_{ij}$, here X_i is the number of round where packet is dropped during TDP_i. Then $E[A] = (E[X]+1)E[r] = RTT(E[X]+1)$

Then we can get throughput:

$$
\begin{aligned}
B(p,\gamma) &= \frac{E[Y]}{E[A]} \\
&= \frac{\frac{1-p}{p} + E[W]}{RTT(E[X]+1)}
\end{aligned}
\tag{3.18}
$$

3.4.3 TCP-Manet Congestion Control and Avoidance Algorithm

Then we discuss the model of new congestion control and avoidance mechanism. Traditional TCP uses additive increasing and multiplicative decreasing congestion avoidance algorithm, which halves the congestion window size on packet loss. This behavior will results in a saw tooth pattern for the transmission pattern. In wireless ad-hoc networks, random packet losses can highly increase the saw tooth pattern; therefore reduce the total throughput of the TCP connection. Hence, in TCP-Manet, we introduce a new congestion avoidance mechanism (section 3.2), which reduce the window size gradually. The purpose is to alleviate

the throughput loss due to high packet error rate along the path from the source to the destination.

Let TI period (TIP) to be the period between two iterations. One iteration is defined as the time between two consecutive packet losses with an increasing power trend. For i-th TI period we define P_i to be the number of packets sent in the period. T_i the duration of the period. W_i the window size at the end of period. We have

$$B = \frac{E[P]}{E[T]} \tag{3.19}$$

Each TI period can be considered as a series of TD period. Let n_i be the number of TD periods in interval TI_i. For j-th TD period of interval TI_i, Y_{ij} denotes the number of packets sent in the period, A_{ij} the duration of the period, X_{ij} the number of rounds in the period, and W_{ij} the window size at the end of period.

In order to derive an expression for B, the long-term steady-state TCP throughput, we must next derive expressions for the mean P and T.

$$P_i = \sum_{i=1}^{n_i} Y_{ij} \tag{3.20}$$

$$E[P] = E[\sum_{i=1}^{n_i} Y_{ij}] = E[n]E[Y] \tag{3.21}$$

$$T_i = \sum_{i=1}^{n_i} A_{ij} \tag{3.22}$$

$$E[T] = E[\sum_{i=1}^{n_i} A_{ij}] = E[n]E[A] \tag{3.23}$$

To derive E[n] observe that, during TI_i, there are n_i TDPs, where each of the first n_{i-1} with decrease power trend, and the last TDP end with an increasing power trend.

$$\begin{aligned} P[n=k] &= (p_w(1-p_1)+(1-p_w)(1-p_2))^{k-1}(p_w p_1+(1-p_w)p_2) \\ &= (1-\gamma)^{k-1}\gamma \end{aligned} \tag{3.24}$$

$$E[n] = \sum_{k=1}^{\infty} 1-\gamma)^{k-1}(\gamma)k = \frac{1}{\gamma} \tag{3.25}$$

From 3.5, we infer that

$$W_{ij} = \begin{cases} W_{i,j-1} - 2^{j-1} + \frac{X_{i,j}}{b}, & \text{if } 2^{j-1} < \frac{W_{i,j-1}}{2} \\ \frac{W_{i,j-1}}{2} + \frac{X_{i,j}}{b}, & \text{if } 2^{j-1} >= \frac{W_{i,j-1}}{2} \end{cases} \tag{3.26}$$

$$\begin{aligned} Y_{ij} &= \sum_{k=0}^{x_{i-1,j-1}/b-1} (f(w_{i-1,j-1}+k)b + \beta_{ij} \\ &= f(w_{i-1,j-1}X_{ij} + \frac{X_{ij}}{2}(\frac{X_{ij}}{b}-1) + \beta_{ij} \\ &= \frac{X_{ij}}{2}(2f(w_{i-1,j-1}) + \frac{X_{ij}}{b} - 1) + \beta_{ij} \\ &= \frac{X_{ij}}{2}(f(w_{i-1,j-1}) + W_{ij} - 1) + \beta_{ij} \end{aligned} \tag{3.27}$$

$$E[Y] = \frac{E[X]}{2}(E[f(w_{i-1,j-1})] + E[W] - 1) + \beta_{ij} \tag{3.28}$$

Case 1: $2^{E[n]-1} < E[W]/2$

$$\begin{aligned} E[W] &= E[W] - 2^{E[n]-1} + \frac{E[X]}{b} \\ E[X] &= b2^{E[n]-1} \end{aligned} \tag{3.29}$$

$$\begin{aligned} \sum_{i=1}^{E[n]} E[Y] = \frac{E[X]}{2}(E[n]E[W] &- (1 + ... + 2^{E[n]-1}) + E[n]E[W] - E[n]) + E[n]\frac{E[W]}{2} \\ E[n](\frac{1-p}{p} + E[W]) &= \frac{E[X]}{2}(2E[n]E[W] - (2^{E[n]} - 1 + E[n])) + E[n]\frac{E[W]}{2} \\ \frac{1-p}{p\gamma} + \frac{E[W]}{2\gamma} &= b2^{E[n]-2}(\frac{2}{\gamma}E[W] - (2^{E[n]} - 1 + E[n])) \\ \frac{1-p}{p\gamma} + \frac{E[W]}{2\gamma} &= \frac{b2^{E[n]-1}}{\gamma}E[W] - b2^{E[n]-2}(2^{E[n]} - 1 + \frac{1}{\gamma}) \\ \frac{(b2^{E[n]} - 1)E[W]}{2\gamma} &= \frac{1-p}{p\gamma} + b2^{E[n]-2}(2^{E[n]} - 1 + \frac{1}{\gamma}) \\ E[W] &= (\frac{1-p}{p\gamma} + b2^{E[n]-2}(2^{E[n]} - 1 + \frac{1}{\gamma})\frac{2\gamma}{b2^{E[n]} - 1} \\ E[W] &= \frac{2(1-p)}{p(b2^{E[n]} - 1)} + b2^{E[n]-2}(2^{E[n]} - 1 + \frac{1}{\gamma})\frac{2\gamma}{b2^{E[n]} - 1} \end{aligned} \tag{3.30}$$

Case 2: $2^{E[n]-1} >= E[W]/2$

Then we get:

$$E[X] = \frac{bE[W]}{2} \tag{3.31}$$

$$\sum_{i=1}^{E[n]} E[Y] = \frac{E[X]}{2}(E[n]E[W] - (1 + ... + 2^{logw/2} + (E[n] - 1 - logw/2) \times w/2 + E[n]E[W] - E[n]) + E[n]\beta_{ij}$$

$$E[n](\frac{1-p}{p} + E[W]) = \frac{bE[W]}{2}(2E[n]E[W] - (2^{\frac{logE[W]}{2}+1} - 1 + (E[n] - 1 - \frac{logE[W]}{2})\frac{E[W]}{2} + E[n])) + \frac{E[n]E[W]}{2}$$

$$\frac{1-p}{p\gamma} + \frac{E[W]}{2\gamma} = \frac{bE[W]}{2}(\frac{2}{\gamma}E[W] - (2^{logE[W]/2+1} - 1 + \frac{(E[n]-1)E[W]}{2} - \frac{E[W]log(E[W]/2)}{2} + E[n]))$$

$$\frac{1-p}{p\gamma} + \frac{E[W]}{2\gamma} = \frac{bE[W]}{2}(\frac{2}{\gamma}E[W] - E[W] + 1 - \frac{(1-\gamma)E[W]}{2\gamma} + \frac{E[W](E[W]-2)}{ln2(E[W]+2)} - \frac{1}{\gamma})$$

$$\frac{1-p}{p\gamma} + \frac{E[W]}{2\gamma} = \frac{bE[W]}{2}(\frac{3-\gamma}{2\gamma}E[W] + \frac{E[W](E[W]-2)}{ln2(E[W]+2)} - \frac{1-\gamma}{\gamma})$$

$$\frac{1-p}{p\gamma} + \frac{E[W]}{2\gamma} = \frac{b(3-\gamma)}{4\gamma}E[W]^2 + \frac{bE[W]^2(E[W]-2)}{2ln2(E[W]+2)} - \frac{b(1-\gamma)}{2\gamma}E[W]$$

$$\frac{b(3-\gamma)}{4\gamma}E[W]^2 + \frac{bE[W]^2(E[W]-2)}{2ln2(E[W]+2)} - \frac{b(1-\gamma)+1}{2\gamma}E[W] - \frac{1-p}{p\gamma} = 0$$

$$\begin{aligned} b(3-\gamma)pln2(E[W]+2)E[W]^2 &+ 2p\gamma bE[W]^2(E[W]-2) \\ -2pln2(E[W] &+ 2)(b(1-\gamma)+1)E[W] - 4(1-p)ln2(E[W]+2) = 0 \end{aligned}$$

$$\begin{aligned} b(3-\gamma)ln2E[W]^3 + 2b(3 &- \gamma)pln2E[W]^2 + 2p\gamma bE[W]^3 - 4p\gamma bE[W]^2 \\ &- 2pln2(b(1-\gamma)+1)E[W]^2 - 4pln2(b(1-\gamma)+1)E[W] \\ &- 4(1-p)ln2E[W] - 8(1-p)ln2 = 0 \end{aligned}$$

$$\begin{aligned} (3bpln2 &- b\gamma pln2 + 2p\gamma b)E[W]^3 + (6bpln2 - 2b\gamma pln2 \\ &- 4pb\gamma - 2pbln2 + 2pb\gamma ln2 - 2pln2)E[W]^2 \\ &- (4pbln2 - 4pb\gamma ln2 + 4pln2 + 4ln2 - 4pln2)E[W] - 8(1-p)ln2 = 0 \end{aligned}$$

$$\begin{aligned} (2bpln2 - b\gamma pln2 + 2p\gamma b)E[W]^3 + (4bpln2 - 4p\gamma b - 2pln2)E[W]^2 \\ -(4pbln2 - 4pb\gamma ln2 + 4ln2)E[W] - 8(1-p)ln2 = 0 \end{aligned} \tag{3.32}$$

Then we can resolve throughput B, based on above formulas.

3.4.4 Selfish Nodes Detection

Then we consider the throughput with the existence of selfish node along the path. Since TCP-Manet determines the existence of selfish node while TCP is in time-out period, we show how TCP-Manet impacts the TCP throughput performance. [67] shows that:

$$B = \frac{E[Y] + Q * E[R]}{E[A] + Q * E[Z^{TO}]} \tag{3.33}$$

Here $E[Y]$ is the expected value of the number of packets sent in a TD period. $E[A]$ is the expected value of the duration of the TD period. $E[R]$ denotes the expected value of the total number of packets sent during time-out sequence. $E[Z^{TO}]$ is the expected value of the duration of the time-out sequence. Q is the probability that a loss indication ending a TDP is a TO. Then we determine Q, $E[R]$ and $E[Z^{TO}]$.

Let q_s the fraction of selfish nodes in the network. We use denotation in [67]: A(w, k) the probability that the first k packets are ACKed in a round of w packets, given there is a sequence of one or more losses in the round.

$$A(w,k) = \frac{(1-p)^k p}{1-(1-p)^w} \tag{3.34}$$

Also C(n, m) to be probability that m packets are ACKed in sequence in the last round (where n packets were sent) and the rest of the packets in the round, if any, are lost.

$$C(n,m) = \begin{cases} (1-p)^m p, & m \leq n-1 \\ (1-p)^n, & m = n \end{cases} \tag{3.35}$$

So the probability that a loss in a window of size w is a timeout (TO) is given by

$$Q(w) = \begin{cases} 1, & w \leq 3 \\ \sum_{k=0}^{2} A(w,k) + \sum_{k=3}^{w} A(w,k) \sum_{k=0}^{2} C(k,m) + q_s - & \\ \quad (\sum_{k=0}^{2} A(w,k) + \sum_{k=3}^{w} A(w,k) \sum_{k=0}^{2} C(k,m) q_s & otherwise \end{cases} \tag{3.36}$$

Since TO occurs if there is a selfish node or the number of packets successfully transmitted is less than three or if the number of packets successfully transmitted in the last round is less than three.

Then we can get

$$\begin{aligned} Q(w) &= min(1, \frac{(1-(1-p)^3(1+(1-p)^3(1-(1-p)^{w-3}))}{1-(1-p)^w} + q_s + \\ &\quad \frac{(1-(1-p)^3(1+(1-p)^3(1-(1-p)^{w-3}))q_s}{1-(1-p)^w} \end{aligned} \tag{3.37}$$

$Q \approx Q(E[W])$, where E[W] is given in previous sections.

Then we consider $E[R]$, and $E[Z^{TO}]$. First we consider the probability of the number of timeouts in a TO sequence, given that there is a TO. Since only one TCP packet is transmitted between two time-outs in sequence. A sequence of k timeouts occurs when there are k-1 consecutive losses followed by a successfully transmitted packet.

$$P[R=k] = \begin{cases} p^{k-1}(1-p), & k \leq 6 \\ p^{k-1}(1-p)(1-q_s) + q_s p^{k-6}(1-p), & k > 6 \end{cases} \tag{3.38}$$

$$\begin{aligned} E[R] &= \sum_{k=1}^{\infty} kP[R=k] \\ &= \sum_{k=1}^{6} kp^{k-1}(1-p) + \sum_{k=7}^{\infty} kp^{k-1}(1-p)(1-q_s) + \sum_{k=6}^{\infty} kp^{k-6}(1-p)q_s \end{aligned}$$

$$
\begin{aligned}
&= \frac{1-q_s}{1-p} + \frac{q_s}{1-p} + \frac{q_s}{1-p} + \frac{5}{1-p} \\
&= \frac{1+q_s+5}{1-p}
\end{aligned} \tag{3.39}
$$

For $E[Z^{TO}]$, the average duration of a timeouts is the same as [1]. For the first six time-outs in one sequence have length $2^{i-1}T_0$, i = 16. With all immediate following timeouts having length $64T_0$

$$
L_k = \begin{cases} (2^{k-1}-1)T_0 & fork \leq 6 \\ (63+64(k-6)T_0 & fork \geq 7 \end{cases} \tag{3.40}
$$

Then the mean of Z_{TO} is:

$$
\begin{aligned}
E[Z_{TO}] &= \sum_{k=1}^{\infty} L_k P[R=k] \\
&= T_0(\sum_{k=1}^{6}(2^k-1)p^{k-1}(1-p) + 63(1-p)(1-q_s)\sum_{k=7}^{\infty}p^{k-1} \\
&\quad +64(1-p)(1-q_s)\sum_{k=7}^{\infty}(k-6)p^{k-1} + 63q_s(1-p)\sum_{k=7}^{\infty}p^{k-6} \\
&\quad +64q_s(1-p)\sum_{k=7}^{\infty}(k-6)p^{k-6} \\
&= T_0((1-p)(1+3p+7p^2+15p^3+31p^4+63p^5) + 63p^6(1-q_s) + \\
&\quad \frac{64(1-q_s)p^6}{1-p} + 63q_sp^6 + \frac{64q_s}{1-p}) \\
&= T_0((1+2p+4p^2+8p^3+16p^4+32p^5) + \frac{64(p^6-q_sp^6+q_s)}{1-p}
\end{aligned} \tag{3.41}
$$

Then we can get throughput is

$$
B(p,q_s) = \frac{\frac{1-p}{p} + E[W] + Q(E[W])\frac{q_s+6}{1-p}}{RTT(E[X]+1) + Q(E[W])T_0(f(p)+g(p,q_s))} \tag{3.42}
$$

Here

$$f(p) = 1 + 2p + 4p^2 + 8p^3 + 16p^4 + 32p^5 \tag{3.43}$$

$$g(p, q_s) = \frac{64(p^6 - q_s p^6 + q_s)}{1 - p} \tag{3.44}$$

3.5 Performance Evaluation

This section presents the performance evaluation of TCP-Manet protocol. In this section, we will first give the simulation study of TCP-Manet, compare the performance results of TCP-Reno and TCP-Manet, and validate the analytical model with the simulation results. The simulation experiments are conducted using NS-2 simulator [73].

Since the traditional TCP protocol deployed, there is dramatic change of the network environment. Many improvements were proposed to original protocols under different network environments. In the meanwhile, some works have been presented to evaluate the performance of these TCP protocols. [74, 75, 76, 61]

Generally, following requirements should be met for a deployable TCP enhancement [61].

- Throughput Improvement: The new TCP enhancement should improve the throughput.
- Fairness: The TCP enhancement should make better use of available bandwidth without reducing the performance of other competing TCP flows.
- Simple: Ideally, TCP enhancement only requires changes to a TCP sender.

Therefore we consider following core metrics to capture the characteristics of the TCP protocols.

- Throughput:

  The long-term steady-state TCP throughput B is

$$B = lim_{T->\infty} \frac{1}{T} \int_0^T x(t)dt \tag{3.45}$$

Here $x(t)$ denotes the packets transferred per unit time.

- Drop Rate

$$L = \sum_{h \in P} p_h \tag{3.46}$$

Here p_h is the drop rate of hop h, $p_h = lim_{T->\infty} \int_0^T \frac{p(t)}{x(t)} dt$. $p(t)$ denotes the packets dropped per unit time. $x(t)$ denotes the packets traffered per unit time.

- Fairness.

There are many different definition of fairness [77, 78, 79, 80, 81]. The fairness index in [80] is defined as:

$$F = \frac{(\sum_{i=1}^n x_i)^2}{n \sum_{i=1}^n {x_i}^2} \tag{3.47}$$

Here n is the number of flows considered, and x_i is the throughput of flow i. We not only consider the fairness between the same kinds of protocols, but also between different protocols.

- Backward Compatibility

Backward compatibility means the new TCP protocol will not have negative impact on users operating original TCP protocol. To measure the backward compatibility, we will study the throughput while TCP-Manet operating with traditional TCP.

- Responsiveness

Responsiveness measures how fast a protocol reacts to a change in the network configuration. It can react how the packet losses impact TCP protocols. Responsiveness can be assessed by measuring the average throughput over different packet loss probability.

- Effectiveness of Selfish Nodes' Detection [82].

  To determine the detection effectiveness of TCP-Manet, we consider the sensitivity and specificity used in binary classification. Binary classification is the task of classifying the members of a given set of objects into two groups on the basis of whether they have some property or not, such as medical diagnostic test for a certain disease.

  Sensitivity is defined as the ratio of true positive to the total number of positive cases. It measures the detection ability to correctly identify the presence of the selfish nodes. A high sensitivity means few false negatives.

$$sensitivity = \frac{number\ of\ true\ positives}{number\ of\ true\ positives\ +\ number\ of\ false\ negatives} \tag{3.48}$$

  A sensitivity of 100% means that all misbehavior nodes are recognized as such.

  Specificity is defined as the ratio of true negative to the total number of negative cases. It measures the detection ability to correctly identify the absence of the selfish nodes. A high specificity means few false positive.

$$sepecificity = \frac{number\ of\ true\ negatives}{number\ of\ true\ negatives\ +\ number\ of\ false\ positives} \tag{3.49}$$

  A specificity of 100% means that all non-selfish nodes are labeled as non-selfish.

  Here,

  True positives: those who detect positive for a condition and are positive (have the condition);

False positives: those who detect positive for a condition and are negative (i.e. do not have the condition);

True negatives: those who detect negative and are negative;

False negatives: those who detect negative and are positive.

Sensitivity or specificity alone does not tell us about the performance of the detection, because 100% sensitivity can be trivially achieved by labeling all the nodes are selfish nodes and specificity of 100% can be achieved by labeling all the nodes are non-selfish nodes. Therefore, we need know both the sensitivity and specificity. High sensitivity and specificity has high detection performance.

3.5.1 Simulation Topology Setup

To focus on the performance metrics above and capture the features of the TCP protocol, we design two wireless network topologies.

- Dumbbell Topology.

  The topology is shown in Figure 3.7. All the TCP flows are run between source Src and destination Dest. Src_N ($N \in 1, 2$) are TCP senders that run TCP-Reno or TCP-Manet. Dest_N ($N \in 1, 2$) are TCP receivers. The links are labeled with bandwidth B and propagation delay T (which is small and can be ignored). We consider bandwidths ranging from 1Mbps to 50Mbps. All the queues are configured as first-in-first-out(FIFO) queues. Although this topology is a constrained topology, it is a typical topology used by most researchers to study the fundamental properties of TCP protocol. The behaviors of TCP protocols are well studied in this topology, it is a good benchmark while we studying TCP-Manet.

- Random Network.

  The topology is shown in Figure 3.8 with 40 nodes in a $1000m \times 1000m$ area. We consider a random network where hosts are placed at random on a two-dimensional area with the additional constraints that the network is being connected initially. This type of random network is appropriate for modeling the wireless ad-hoc networks. We randomly establish TCP connections among these hosts, and then study the detection effectiveness of TCP-Manet in ad-hoc networks.

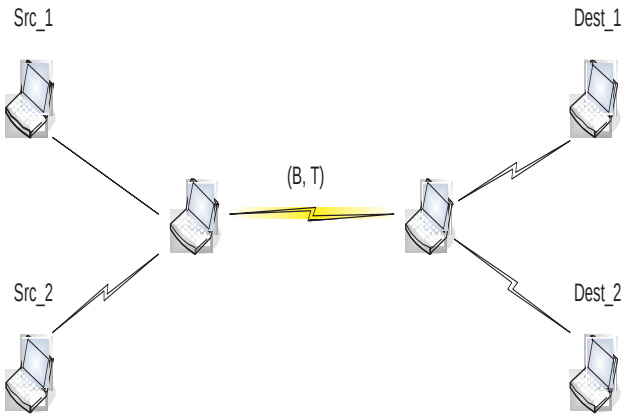


Figure 3.7: Dumbbell topology.

3.5.2 Throughput

We conducted experiments over dumbbell topology (section 3.5.1). The ns-2 simulator is modified to add TCP-Simple and TCP-Manet agents. Figure 3.9(a) and (b) show the throughput of TCP-Manet over low bandwidth of 2Mbps and high bandwidth of 20Mbps networks. We define TCP flow from source node 1 to destination node 1 and UDP flow from source node 2 to destination node 2. The packet arrival rates are constant bit rate (CBR) of 100 packets/sec in a 2Mbps network and 1000 packets/sec in a 20Mbps network. The segment size is 1500 bytes.

When the packet error rate is low, TCP-Reno, TCP-Simple and TCP-Manet have similar throughput. With the increase of the packet error rate, TCP-Manet has highest throughput and TCP-Reno has lowest throughput. However, as the packet error rate further increases to 5%, all of the three protocols have low throughput. At very low loss rates, random loss is not a significant factor. The likelihood that the congestion window size is impacted by random loss is small. Also, under heavy random loss situation, none of these three protocols can maintain TCP self-clocking mechanism, and they all experience frequent timeouts. In a low speed network, when packet error rate is 1%, TCP-Manet has throughput of about 62 packets/sec. Compared with TCP-Reno of 54 packets/sec, it is 15% of performance improvement. Correspondingly, in a high speed network, TCP-Manet has throughput of 537 packets/sec at packet error rate of 0.1%. Compared with TCP-Reno of 412 packet/sec, it is a 30% of performance improvement.

To better understand the throughput and verify the theoretical model of the TCP, we also compare the

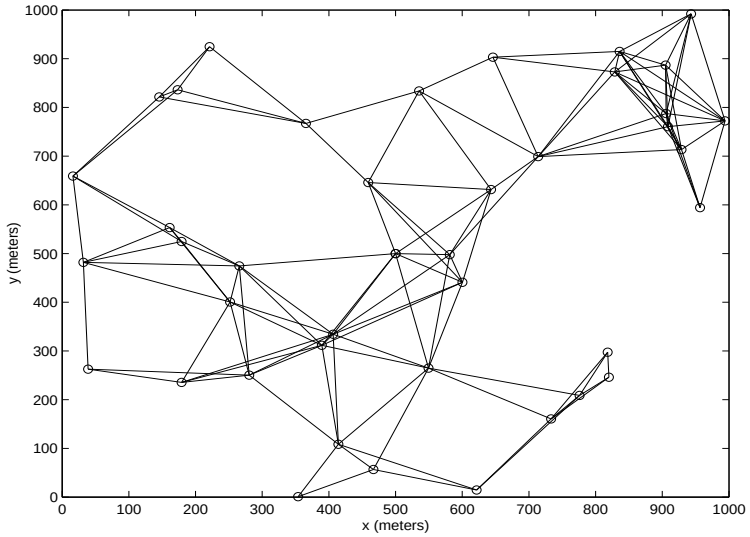


Figure 3.8: Wireless ad-hoc experimental network. (40-node random network in $1000m \times 1000m$ area)

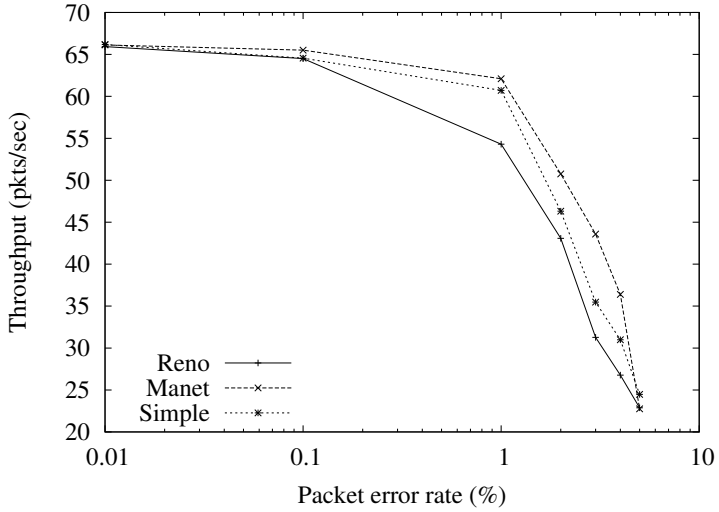
simulation results with the theory results (section 3.4).

3.5.3 Fairness

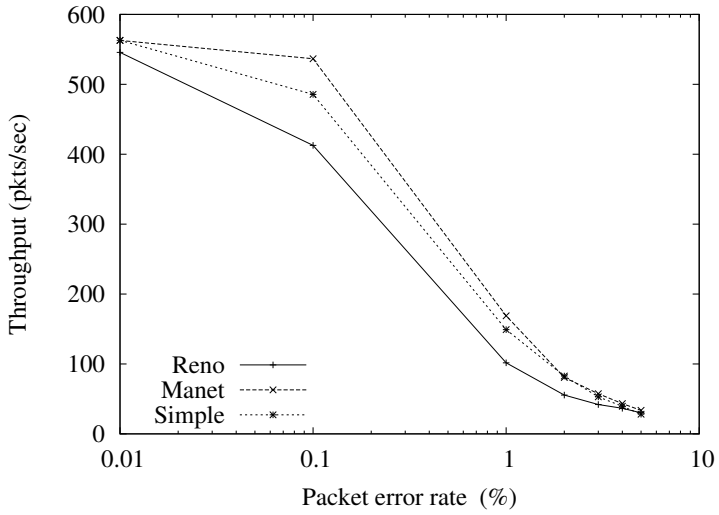
To evaluate the performance of TCP-Manet, one important feature is fairness and friendliness. Figure 3.10 and 3.11 show the fairness ratio of TCP-Reno, TCP-Simple, and TCP-Manet in low bandwidth of 2Mbps and high bandwidth of 20Mbps networks. We define two TCP flows from source node 1 to destination node 1, and source node 2 to destination node 2. The packet arrival rates are constant bit rate (CBR) of 100 packets/sec in 2Mbps network and 1000 packets/sec in 20Mbps. The segment size is 1500 bytes.

Figure 3.10 shows the fairness ratio of TCP-Reno, TCP-Simple, and TCP-Manet, when both the TCP connections are suffer the same loss rate. For both low bandwidth and high bandwidth network, all three protocols have similar high ratio fairness ranging from 98% to 100%. That implies TCP-Manet does not hurt the fairness of the original TCP protocol. When error rate is low, the TCP-Manet has slightly better fairness ratio than TCP-Reno and TCP-Simple. However, with the increase of the error rate, TCP-Manet have slightly worse fairness ratio than TCP-Reno and TCP-Simple.

Figure 3.11 shows the fairness ratio of TCP-Reno, TCP-Simple, and TCP-Manet of low bandwidth and high bandwidth networks, where only first TCP connection suffers random packet loss. With the increase of the packet error rate, the fairness ratio decreases. This is expected. If one connection suffers higher



(a)



(b)

Figure 3.9: Comparison of throughput of TCP-Reno, TCP-Simple and TCP-Manet.

Table 3.1: Throughput comparison of theoretical and simulation results of TCP-Reno (Bandwidth = 2Mbps)

Error rate (%)	Packet delivered (pkts)	Num timeouts	Num retrans	Loss frequency (p) (%)	avgRTT (sec)	Theoretical (pkts/sec)	Simulation (pkts/sec)
0	12941	4	32	0.278	0.335	47.544	65.047
0.01%	13113	4	34	0.29	0.339	45.966	65.927
0.1%	12836	2	45	0.366	0.293	46.909	64.512
1%	10854	8	115	1.133	0.177	39.398	54.306
2%	8606	15	148	1.894	0.151	32.874	43.077
3%	6243	31	169	3.2036	0.148	22.862	31.267
4%	5344	39	165	3.817	0.159	18.999	26.776
5%	4568	41	178	4.794	0.148	16.617	22.8815

Table 3.2: Throughput comparison of theoretical and simulation results of TCP-Simple (bandwidth = 2Mbps)

Error rate (%)	Packet delivered (pkts)	Num timeouts	Num retrans	Loss frequency (p) (%)	avgRTT (sec)	Theoretical (pkts/sec)	Simulation (pkts/sec)
0	13088	2	31	0.252	0.341	55.883	65.772
0.01%	13166	2	26	0.213	0.336	55.857	66.177
0.1%	12841	5	34	0.3037	0.321	49.985	64.552
1%	12131	3	129	1.088	0.199	36.893	60.712
2%	9257	12	180	2.074	0.169	28.403	46.313
3%	7078	27	196	3.15	0.161	25.204	35.482
4%	6180	36	198	3.786	0.166	18.650	30.982
5%	4879	46	188	4.796	0.171	15.213	24.471

Table 3.3: Throughput comparison of theoretcal and simulation results of TCP-Manet (bandwidth = 2Mbps)

Error rate (%)	Packet delivered (pkts)	Num timeouts	Num retrans	Loss frequency (p) (%)	avgRTT (sec)	Theoretical (pkts/sec)	Simulation (pkts/sec)
0	13018	5	34	0.299	0.352	61.347	65.622
0.01%	13125	5	44	0.373	0.372	50.813	66.127
0.1%	13009	4	41	0.346	0.346	57.284	65.517
1%	12370	4	120	1.002	0.238	42.497	62.102
2%	10116	15	190	2.026	0.192	29.717	50.748
3%	8656	25	209	2.703	0.195	23.066	43.588
4%	7242	34	221	3.521	0.165	20.022	36.392
5%	4525	49	173	4.906	0.160	14.624	22.756

Table 3.4: Throughput comparison of theoretical and simulation results of TCP-Reno (bandwidth = 20Mbps)

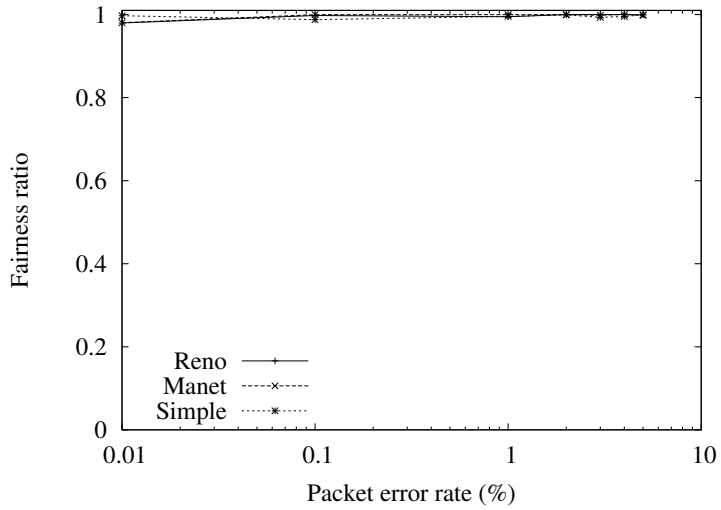
Error rate (%)	Packet delievered (pkts)	Num timeouts	Num retrans	Loss frequency (p) (%)	avgRTT (sec)	Theoretical (pkts/sec)	Simulation (pkts/sec)
0	113052	5	52	0.05	0.079	482.933	568.495
0.01%	108510	3	55	0.0534	0.077	478.017	545.748
0.1%	82035	2	83	0.104	0.070	370.089	412.695
1%	20153	12	207	1.087	0.084	77.469	101.633
2%	11011	33	211	2.216	0.101	40.017	55.455
3%	8335	33	236	3.227	0.096	30.468	41.927
4%	7273	42	234	3.822	0.102	25.423	36.605
5%	5959	52	223	4.615	0.117	19.858	30.027

Table 3.5: Throughput comparison of theoretical and simulation results of TCP-Simple (bandwidth = 20Mbps)

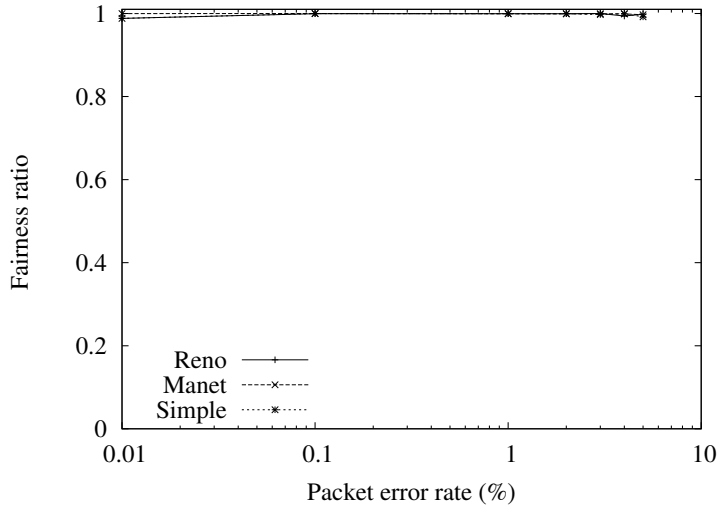
Error rate (%)	Packet delivered (pkts)	Num timeouts	Num retrans	Loss frequency (p) (%)	avgRTT (sec)	Theoretical (pkts/sec)	Simulation (pkts/sec)
0	113104	5	47	0.0459	0.079	525.361	568.771
0.01%	111920	4	52	0.05	0.078	504.362	562.891
0.1%	96539	3	101	0.108	0.074	375.205	485.587
1%	29613	6	265	0.915	0.078	92.579	149.166
2%	16445	18	296	1.909	0.092	48.076	82.728
3%	10532	35	295	3.13	0.101	30.505	53.018
4%	7818	47	242	3.697	0.131	22.392	39.479
5%	5575	61	210	4.86	0.127	18.187	28.162

Table 3.6: Throughput comprison of theoretical and simulation results for TCP-Manet (bandwidth = 20Mbps)

Error rate (%)	Packet delivered (pkts)	Num timeouts	Num retrans	Loss frequency (p) (%)	avgRTT (sec)	Theoretical (pkts/sec)	Simulation (pkts/sec)
0	108021	8	55	0.06832	0.079	589.359	543.341
0.01%	111898	6	62	0.0608	0.079	631.609	562.816
0.1%	106702	7	153	0.15	0.083	358.339	536.647
1%	33553	7	328	1	0.083	95.150	168.967
2%	16003	25	301	2.037	0.104	43.703	80.682
3%	11391	36	293	2.888	0.117	29.210	57.495
4%	8601	42	273	3.662	0.103	24.169	43.304
5%	6648	51	272	4.86	0.121	16.640	33.615

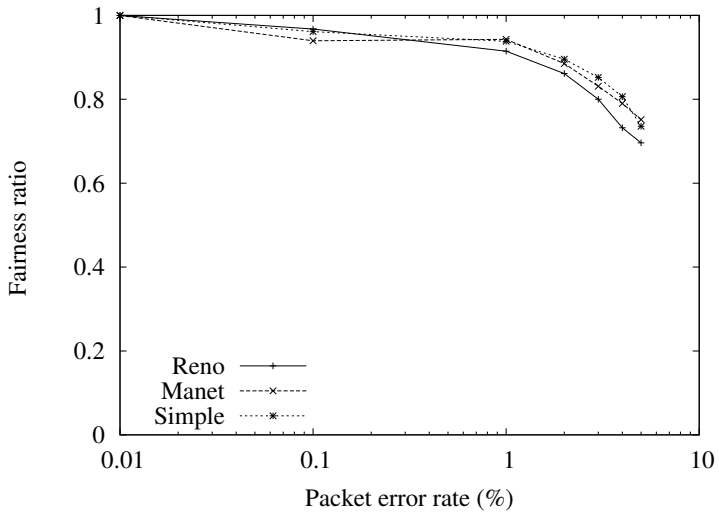


(a)

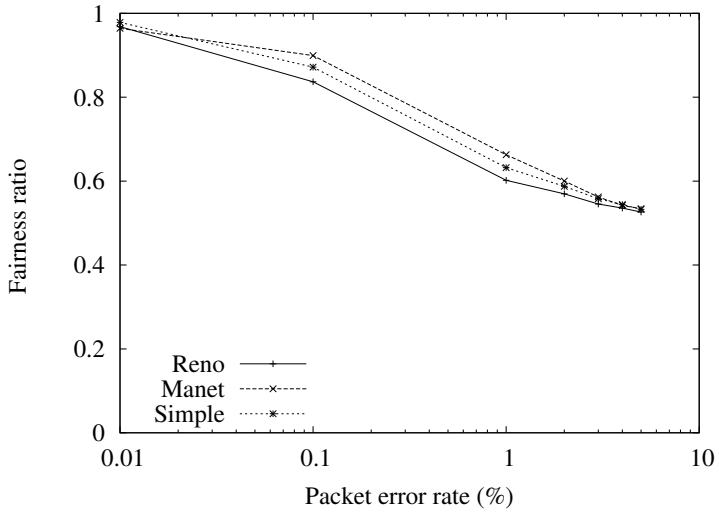


(b)

Figure 3.10: Comparison of fairness ratio of TCP-Reno, TCP-Simple and TCP-Manet under symmetrical condition. (a) Bandwidth = 2Mbps (b) Bandwidth = 20Mbps



(a)



(b)

Figure 3.11: Comparison of fairness ratio of TCP-Reno, TCP-Simple and TCP-Manet under unsymetrical condition. (a) Bandwidth = 2Mbps (b) Bandwidth = 20Mbps

packet error rate, the throughput of this connection will reduce. This allows the other connection get more bandwidth. Therefore the fairness ratio will reduce. In low speed network, TCP-Reno is fairer than the other protocols, when packet error rate is low. With the increase of the packet error rate, TCP-Simple and TCP-Manet are fairer than TCP-Reno. In high speed network, TCP-Simple and TCP-Manet are fairer than TCP-Reno. Because TCP-Reno aggressively halve its window size on packet loss, when connection experiences high packet error rate, its window size will fluctuate uncertainly, which hurts the fairness ratio. Therefore, under high packet error rate, TCP-Manet and TCP-Simple are more stable than TCP-Reno.

3.5.4 Backward Compatibility

One of the basic, practical requirements of a new protocol is that they won't create a large impact on users operating legacy protocols. To evaluate the backward compatibility, we repeat the previous fairness measurement, but now with the first flow operating TCP-Simple or TCP-Manet and the second flow operating the standard TCP-Reno.

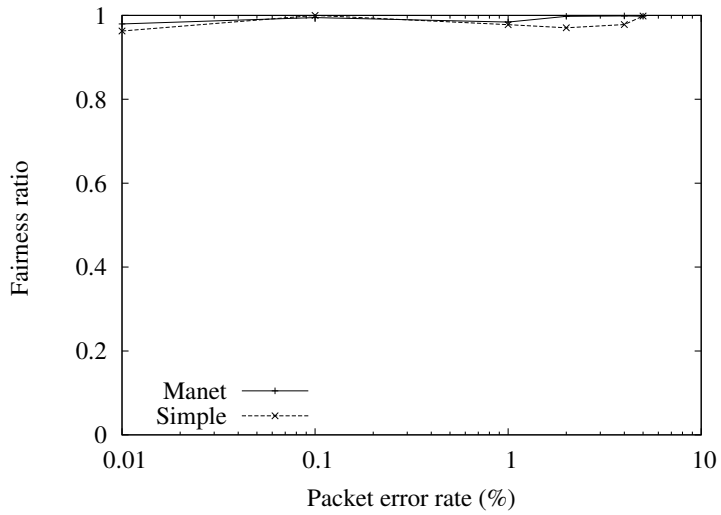
Figure 3.12 shows the fairness ratio of TCP-Simple and TCP-Manet with standard TCP-Reno protocols under symmetric conditions, in which both the TCP connections suffer the same packet loss rate. The fairness ratios in both cases are in range from 95% to 100%.

3.5.5 Effectiveness of Selfish Nodes' Detection

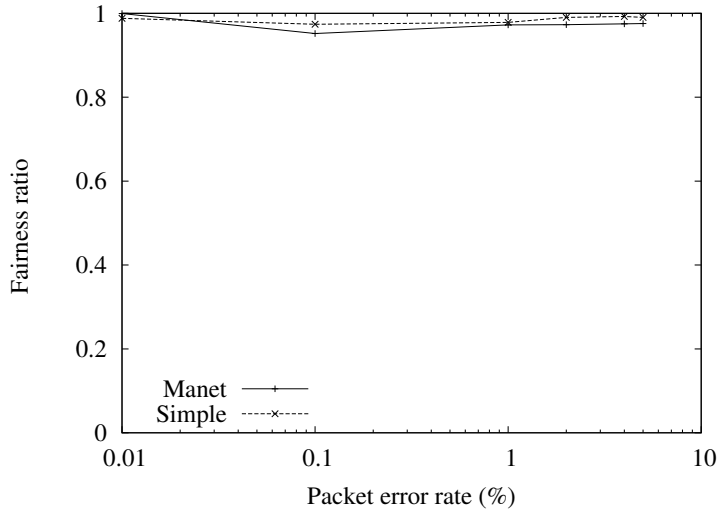
We conducted experiments over a random network topology described in section 3.5.1. The ns-2 simulator was modified to enable particular node(s) to be configured as selfish nodes. The configuration also takes in a time parameter that specifies the time from which that node starts behaving as a selfish node. Beginning from that time, the nodes drops all the packets (non-control packets) that are received at that node till the end of the simulation. The number of selfish nodes varies from 5% to 40%. The placement of the selfish nodes ensures that they will be located along active paths in the network. Each point in the graph is an average of 10 experiments.

Only Selfish Behavior in the Network

Figure 3.13 shows the detection effectiveness of the TCP-Manet. With the increase of selfish nodes in the network, both the sensitivity and specificity decrease. This implies the decrease of the detection effectiveness of TCP-Manet. This is because, when the number of selfish nodes in the network is small, the probability



(a)



(b)

Figure 3.12: Backword compatibility of TCP-Simple and TCP-Manet under symmetrical condition. (a) Bandwidth = 2Mbps (b) Bandwidth = 20Mbps

that only one selfish node exists in the connection is high. TCP-Manet can find the misbehavior nodes. However, with the increase of the number of selfish nodes in the network, there may be multiple selfish nodes in the networks. The TCP sender will have difficult to re-establish connection and detect and isolate the following selfish nodes.

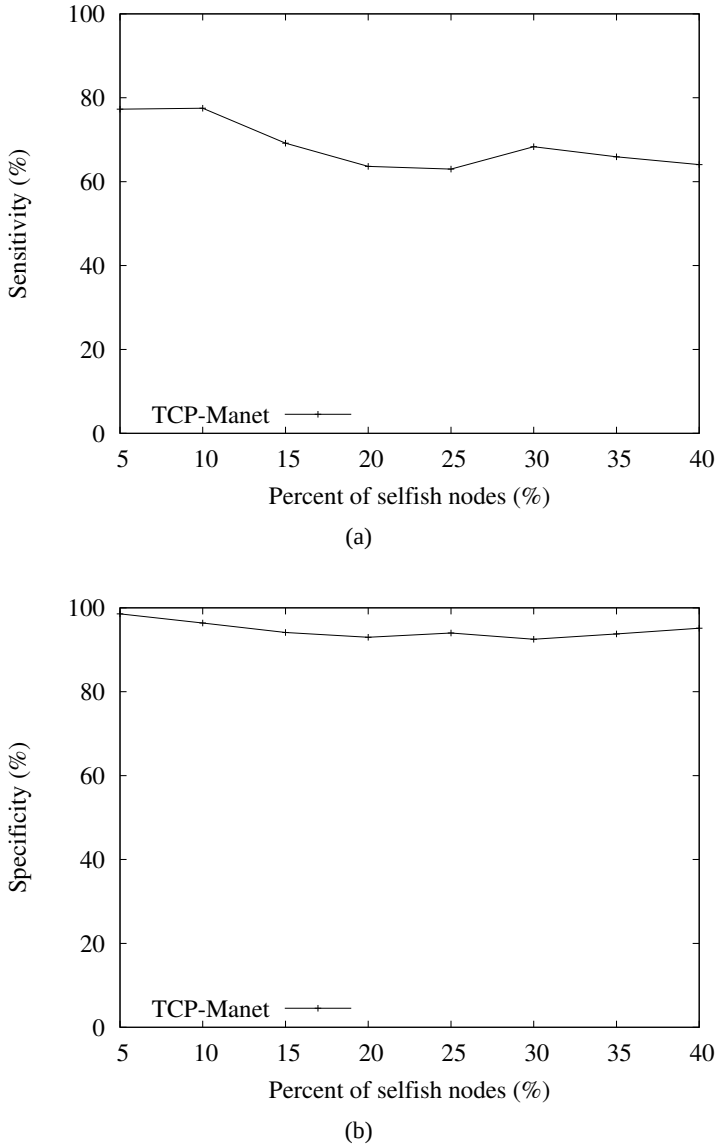


Figure 3.13: Sensitivity and specificity of TCP-Manet varying the number of selfish nodes in the network.

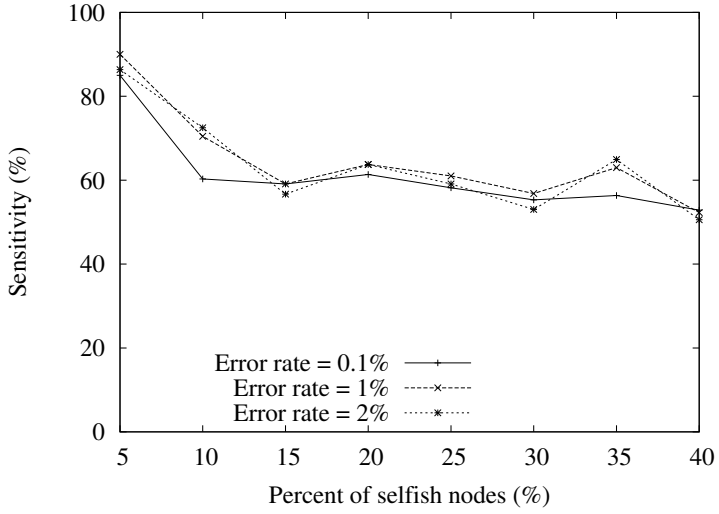
Selfish Behavior with Random Loss

Figure 3.14 shows the detection effectiveness of the selfish nodes of the TCP-Manet with the existence of the random packet loss. 10% of nodes are randomly selected to drop the packet it receives with a specified packet loss rate. As described earlier, with the increase of selfish nodes in the network, both the sensitivity and specificity decrease. Besides, the packet error rate does not have much impact on the detection effectiveness. This is because there are only 10% randomly selected nodes drop the packets. Therefore, the probability that these nodes are in the TCP connection is low.

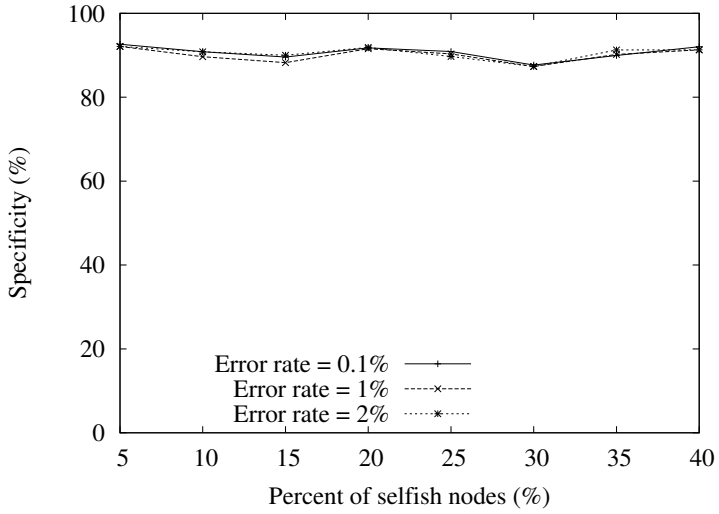
Figure 3.15 shows the detection effectiveness of the selfish nodes of the TCP-Manet with the existence of random packet loss. 20% of nodes are randomly selected to drop the packet it received with a specified packet loss rate. Similarly, with the increase of selfish nodes in the network, both the sensitivity and specificity decrease. Packet error rate does not have much impact on the detection effectiveness. However, compared with figure 3.14 with the increasing broken links in the network, the detection effectiveness will decrease. This is because the probe messages may be lost while transmission. Therefore, after a timeout, the sender will determine a wrong selfish node. This could be improved by retransmit the probe message several times, and after some timeouts the sender determines which nodes is the selfish node.

3.6 Summary

TCP-Manet (an enhancement of TCP protocol) is designed to improve performance of TCP protocol in wireless ad-hoc networks. It modifies the traditional TCP congestion avoidance algorithm to react to the random packet loss in the connection. Instead of blindly half the congestion window size on packet loss, TCP-Manet measures the trend of the power of the TCP connection to determine if the packet loss is due to wireless random loss, and then triggers its congestion control and avoidance algorithm. Beside, to detect the selfish behavior in the TCP connection, TCP sender sends probe messages to find the selfish node in the connection, therefore ask for a new connection that can isolate the selfish nodes. The simulation and analytical results show that TCP-Manet has better throughput performance over the traditional TCP protocols. It is also more robust than traditional TCP protocol with the existence of selfish behavior in the ad-hoc network.

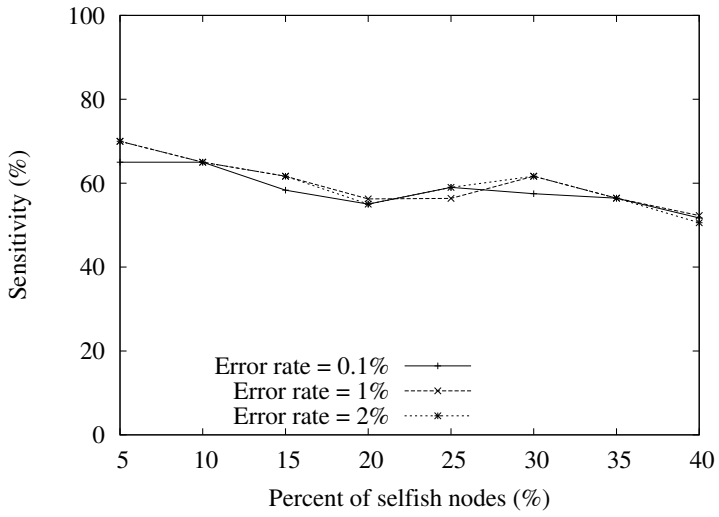


(a)

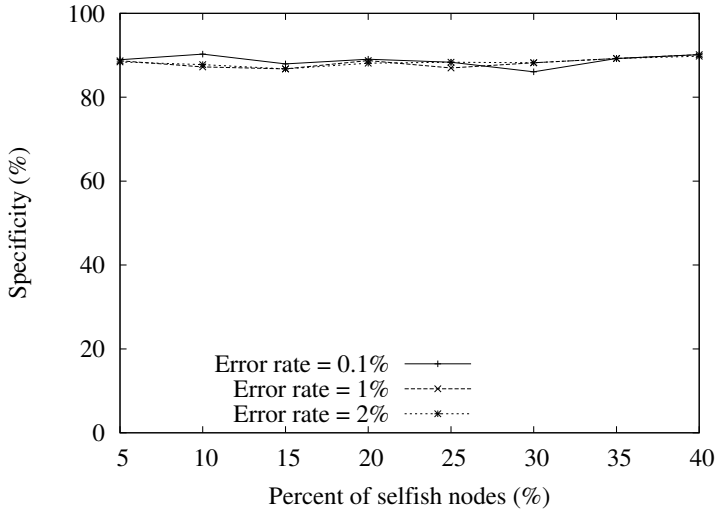


(b)

Figure 3.14: Sensitivity and specificity of TCP-Manet varying number of selfish nodes in the network with 10% of nodes randomly dropping packets.



(a)



(b)

Figure 3.15: Sensitivity and specificity of TCP-Manet varying number of selfish nodes in the network with 20% of nodes randomly dropping packets.

CHAPTER FOUR

DATA DISSEMINATION PROTOCOL (SPIN-G) IN WIRELESS SENSOR NETWORK

Advances in embedded system technologies made it possible to deploy a large number of sensor nodes into a sensor network. These sensor nodes are typically equipped with an embedded processor, one or more sensors, memory, and a low-power radio communication facility. Usually, sensor networks are used for sensing the environmental events, gathering data, and disseminating them throughout the sensing area via the wireless channel. Several critical requirements that influence design and implementation of data dissemination protocols exist:

- *Resource Limitations.* Low-cost and low-power sensor nodes are characterized by limited computation, memory storage, communication, and battery power capabilities. Once sensor nodes are deployed, it is difficult to replace or recharge their battery. Hence, designing power-conserving data dissemination protocol to extend their limited lifetime is a key consideration in sensor networks.

- *Scalability.* The sensor network usually consists of hundreds or even thousands of sensor nodes in the field. Designing a scalable communication protocol is also an important challenge.

- *Timeliness.* Data from sensor nodes are typically time-sensitive. The data in the sensor networks should be received in a timely manner.

SPIN-G is motivated by SPIN protocol (Sensor Protocol for Information via Negotiation) [8] [9]. SPIN aims at minimizing the redundant data transmission by using meta-data negotiation before initiating the real data operation. Hence, it saves energy over classical flooding. Besides this, SPIN-G improves the performance in terms of energy efficiency by deploying the following:

- Gossiping with data aggregation to further reduce the network overhead introduced by meta-data negotiation.

- Make sensor nodes request data from most energetic neighboring node to balance the energy consumption throughout the network.

- Utilize a sleep/active cycle to save the energy of poor nodes and protect them from depleting their energy, thereby improving network lifetime.

4.1 SPIN-G Protocol

This section describes an all-to-all data dissemination protocol SPIN-G using gossip. We will first give a brief overview of SPIN and then introduce the mechanisms of SPIN-G.

4.1.1 SPIN (Sensor Protocols for Information via Negotiation)

SPIN (Sensor Protocol for Information via Negotiation) [10] [11] is an adaptive protocol handling all-to-all information dissemination in wireless sensor networks. It uses negotiation and resource-adaptation to overcome the implosion, overlap, and resource blindness problems of the conventional protocols, such as flooding or gossiping.

In negotiation, sensor nodes use high-level descriptors meta-data to describe or name the data. It works in three stages (ADV-REQ-DATA) to eliminate unnecessary data transfers in the network. A sensor node, in disseminating data, first sends an advertisement (ADV message) describing the new data available with the meta-data to its neighbors. The neighboring nodes, if have not received this data, request such data (with a REQ message). Upon such a request, the sensor node responds with data (DATA message). The overhead of meta-data, ADV and REQ exchange is compensated by the reduction in the duplicate data reception. Since meta-data exchange is based on flooding mechanism, the redundant meta-data messages still exist.

SPIN is resource-adaptive in the sense that each sensor nodes can poll its system resources to find out how much energy is available to them, and determine its activity in terms of energy. However, SPIN sensor nodes are insensitive to the resource capabilities of their neighboring nodes. That is, sensor nodes may keep asking for DATA from some particular neighbor without considering how much energy is left at that node. Some nodes may drain energy faster than others, which lead to network partition, thereby reducing network lifetime.

4.1.2 SPIN-G

The purpose of SPIN-G is to overcome the two deficiencies of SPIN described: (i) reduce the overhead of meta-data negotiation, (ii) achieve balanced energy consumption distribution across the network and extend its lifetime. To achieve the first goal, we employ randomized “gossip” and combine it with data aggregation.

To attain the second goal, we developed a data-requisition strategy to make the sensor nodes choose the advertising neighbor with the most energy to ask for data.

Meta-data Negotiation using Gossiping

SPIN-G is also a 3 stage handshake protocol like SPIN. The protocol starts when a node obtains new data and advertises the data by sending an ADV message to **one** of its randomly selected neighbors. The neighboring node, upon receiving the ADV, checks to see whether it has already received or requested the advertised data. If not, instead of responding right away, sensor node waits for a predefined fixed interval. During this waiting period of time, if the sensor node receives multiple ADVs for the same data from its neighbors, it will use a data requisition strategy (section 4.1.2) to select the most energetic of its advertising neighbors, then responds with a REQ message. The node then responds with a REQ to obtain the DATA message.

Data Aggregation and Retransmission

Employing gossip in negotiation will reduce the data dissemination rate. To alleviate this problem, we use data aggregation and retransmission scheme. When sensor node has new data for advertising, it will aggregate new data with the data it already has and send advertisements of the aggregated data to one of its selected neighbor. Because the sensor node randomly chooses a neighbor for advertising, it has high probability that the sensor nodes will choose a different neighbor from the previous advertisement. Hence, unlike traditional gossip, there are more copies of data flows in the network, speeding up the dissemination rate. In addition, SPIN-G has an ADV retransmission scheme. We predetermine a fixed timeout value, for each node if there is no data arrived for this interval, the node will re-advertise the data it has to a randomly selected neighbor.

Data aggregation and retransmission has the advantage of compensating for link failures. If an ADV of the data is lost in a transmission, the lost data will be aggregated in a new ADV and sent to some other neighbor at a later time.

Resource Adaptation

In SPIN protocol, sensor nodes can only poll their own energy level and then determine whether or not to participate in data dissemination based on their energy level. However, these sensor nodes are resource blind about their neighboring nodes: they may request data from impoverished neighbors further depleting their

resources.

Instead, in SPIN-G, sensor nodes not only know their own energy levels, but also keep track of the energy levels of their neighbors. With this knowledge, a sensor node can request data from a neighbor with highest energy and hence be able to balance the energy dissipation across the network. To accomplish this goal, each sensor node will periodically broadcast its energy level to its neighbors. This could be done in two ways: to broadcast a specific EGY message to the neighbors, or to piggyback the energy level in its advertisement (ADV).

When the sensor node receives multiple ADVs for the same data from its neighbors, it will select its advertising neighbor with the most energy to ask for the DATA. This kind of data requisition from a selective neighbor can lead to an improvement in the life expectancy of the network, and exploit the energy savings possible with fewer transmitted messages.

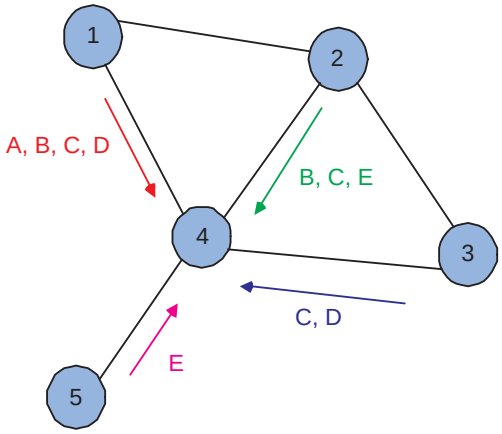
In our data requisition strategy, the node chooses the neighbor with highest energy level, and asks for DATA with REQ message. Each node maintains a table that keeps the data and energy information of all its neighbors. For example, 4.1 (a) shows the topology of a 5-node network, and Table 4.1 is an example of the neighbor table at node 4. For meta-data A, B, C, and D, since the neighbor node 1 has highest energy of 40J, node 4 will send REQ asking for DATA of A, B, C, and D from node 1. For meta-data E, the neighbor node 5 has the highest energy of 30J and node 4 will send REQ for E to node 5. Figure 4.1 (b) illustrates this requisitioning strategy.

Table 4.1: Data and energy information for node 4

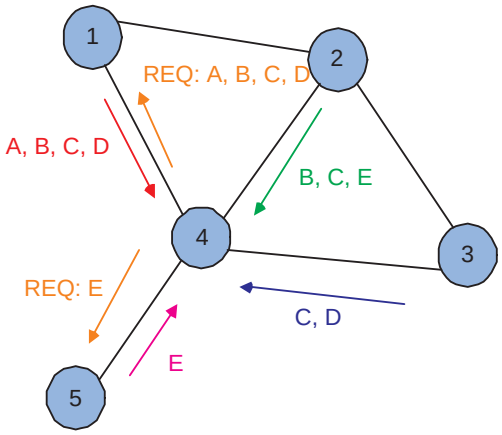
Neighbor	Meta-data	Energy level
1	A, B, C, D	40J
2	B, C, E	20J
3	C, D	30J
5	E	30J

Sleeping-active Cycle for Battery-poor Nodes

The most important way to save energy in a sensor network is to power-down (put to sleep) the sensor node when necessary. To save battery poor nodes in the network and extend the lifetime of the network, we



(a)



(b)

Figure 4.1: Data requisition strategy. (a) ADVs in a 5-node network, (b) REQs in a 5-node network

introduce sleeping-active mode for sensor node that has low battery power. Each sensor has three states: active, sleep and dead. In active state, the nodes is completely functional and can transmit/receive data, in sleep state, the sensor's transceiver is powered off and can not take part in the network activity, and in dead state, sensor node has depleted its battery power. If sensor has abundant energy, it keeps in active mode and takes part in the data dissemination process. However, when its power level falls below a threshold (25% of its initial energy level), the sensor node enters in sleep cycle, in which the node sleeps and wakes up periodically. When the node wakes up, it can join in the dissemination process as usual. The purpose of using sleep cycle for battery poor node is to reduce their participation in data dissemination, protecting them from depleting their energy, and thus extending the network lifetime.

4.2 Performance Evaluation

This section presents the performance analysis of SPIN-G protocol in terms of protocol convergence time and energy consumption, and compares them to SPIN and traditional gossiping. The network and protocols were modeled using NS2 simulator [48].

4.2.1 System Model

The following assumptions are made in the system model, consistent with modeling in literature [10].

- The nodes are deployed in rectangle area.
- All nodes are homogeneous and battery-powered.
- Each node has a limited transmission range r.
- Each node only sends packets to the nodes that are in its transmission range.
- The network is a broadcast network, where only one single, unreliable, broadcast channel is available for all communication.
- The packets can be lost due to collisions or buffer overflow.

Table 4.2 summarizes the system parameters used in the simulation: These parameters are same as the ones reported for SPIN [11] for direct and easy comparison.

Table 4.2: Operational parameter variables

Parameters	Values
Number of nodes (n)	$10 - 70$
Topology (Γ)	Random, regular
DATA packet size (DATA) (S_d)	500 bytes
Meta-data size (ADV, REQ)(S_m)	16 bytes
Network loss	Yes
Simulation area	$40 * 40m^2$
Transmission range (r)	10 m
Initial Energy (J)	$10 - 100$ J
MAC protocol	802.11
Bandwidth (B)	2 Mbps
Transmit Power(E_T)	$5.0023mw$
Receive Power(E_R	$5mw$
Idle Power	$0.0W$
Number of data (d)	$1data/node$ without overlapping
Propagation delay(T_{prop})	$0s$
Processing delay(T_p)	$0.01s$
ADVs timeout (T_{out})	$0.18s$

4.2.2 Performance Metrics

For the evaluation of protocols the following three metrics have been chosen.

1. **Energy Consumption $\mathcal{E}_a$**

   The energy consumption measures the difference between the initial level of energy and the final level of energy that is left for all the nodes in the network. Let,

   $\mathcal{E}_i$ = the initial energy level of a node.

   $\mathcal{E}_f$ = the final energy level of a node.

   n = the number of nodes in the network.

$$\mathcal{E}_a = \sum_{k=1}^{n} (\mathcal{E}_{ik} - \mathcal{E}_{fk}) \tag{4.1}$$

   This metric is important because the energy level that a network uses is inversely proportional to the network's lifetime. Lower the energy consumption the longer is the network's lifespan [47].

2. **Total Data Received by All the Nodes in the Network $\mathcal{D}_p$**

   Let

   $\mathcal{D}$ = total data should be disseminated in the network.

   d_k= the data received by a sensor node k.

   n = number of nodes in the network.

$$\mathcal{D}_p = \frac{\sum_{k=1}^{n} d_k}{\mathcal{D} * n} \tag{4.2}$$

3. **Network Partition Time $\mathcal{T}_p$**

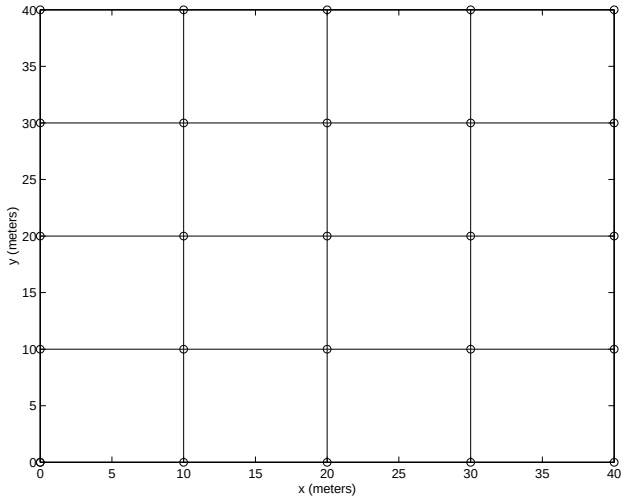
   Network partition time is the duration from the network starts operation till the time network is partitioned due to node failures as their energy is depleted. It is proportional to the energy consumption and the distribution of energy consumption. The lower the energy consumption, the longer the network partition time. The more balanced the energy consumption across the network, the longer the network partition time.

4.2.3 Vary Network Topology

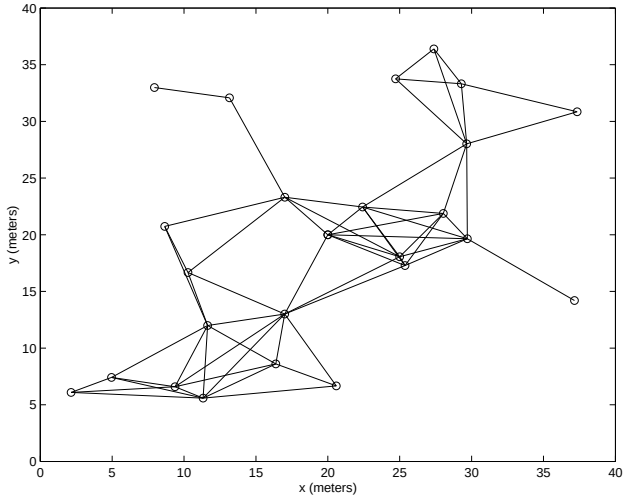
We studied the effect of network topology on SPIN, SPIN-G and Gossip. We consider two sensor network deployment strategies: regular and random [83]. We first study regular deployment strategy, where sensors are distributed as a mesh of 25 nodes in the network, as shown in Figure 4.2(a). We then study random deployment, where sensors are placed at random on a two-dimensional area with the additional constraint that the network be connected. This type of random graph is appropriate for modeling a number of applications such as battle-field, surveillance, etc. We created a 25-node randomly generated network, which is a connected network with 60 edges and an average degree of 4.8., as shown in Figure 4.2(b). For each data point shown, we conducted 10 experiments with different seeds for the simulation and used the average.

Figure 4.3(a) shows the percentage of the total data received ($\mathcal{D}_p$) by all sensor nodes in a mesh network over time for SPIN, SPIN-G, and gossip protocol. SPIN converges fastest at about 1.72 seconds; SPIN-G at 7.11 seconds; and gossip converges slowest at about 24.44 seconds. SPIN converges fastest, as expected, because it is based on flooding for meta-data negotiation and converge in $O(d)$ rounds, where d is the diameter of the network. Traditional gossip protocol converges much slower than SPIN protocol, because optimistic dissemination rate of traditional gossip is at most 1 node/round as there is only one copy of data flowing in the network at any given time. Similar to the traditional gossip protocol, SPIN-G only forwards meta-data to one randomly selected neighbor. Hence, the number of meta-data advertisements received at any node should decrease - thus controlling the implosion problem in meta-data negotiation at the expense of convergence time. SPIN-G converges slightly slower than SPIN, but much faster than gossip protocol. That is, we did not see much increase in convergence time like gossip protocol. This is because we utilize data aggregation and retransmission scheme in SPIN-G, where a sensor sends advertisement of new data aggregated with other data that it already has. In addition, this aggregation is not only helpful for network to cope with link losses, but also speeds up convergence with less energy consumption. This scheme makes our algorithm "gossiping" problem instead of "random walk". As we described in section 4.3, the cover time decreases to $O(logn)$ from possible $O(n^3)$.

Figure 4.3(b) shows the energy consumption of ($\mathcal{E}_a$) of these three protocols in mesh network. SPIN consumes the highest energy, about 0.046J; Gossip consumes the lowest energy, about 0.028J; and SPIN-G consumes 0.037J energy. SPIN-G spends 24% less energy than SPIN and 24% more energy than gossip.

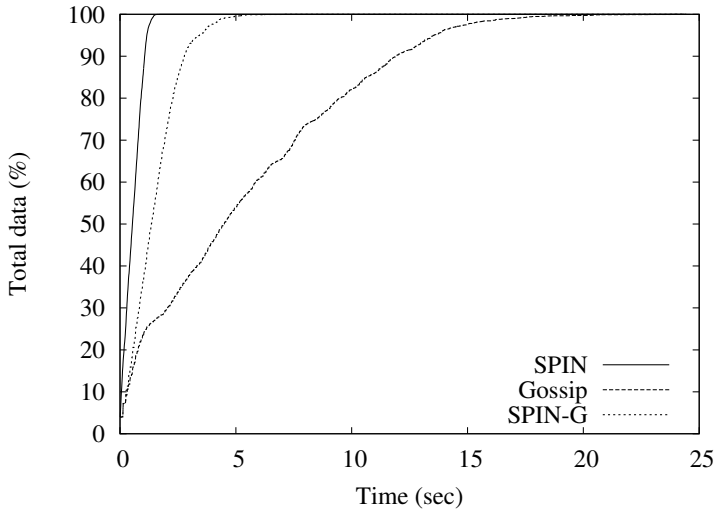


(a)

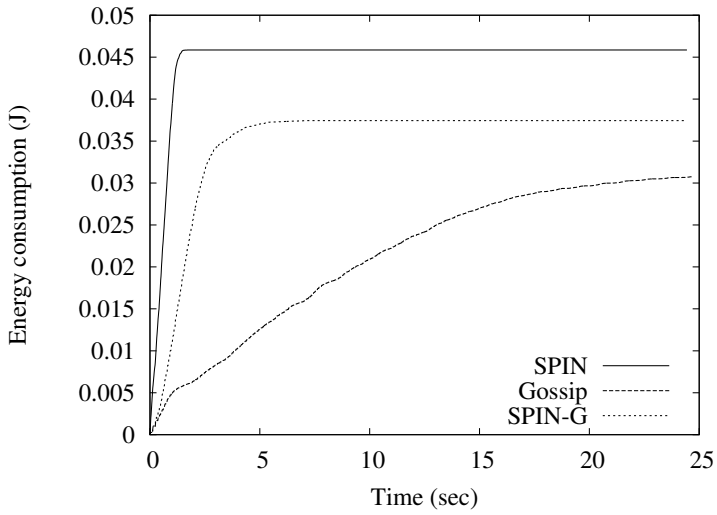


(b)

Figure 4.2: Experimental network topologies. (a) 25-node mesh network, (b) 25-node random network.

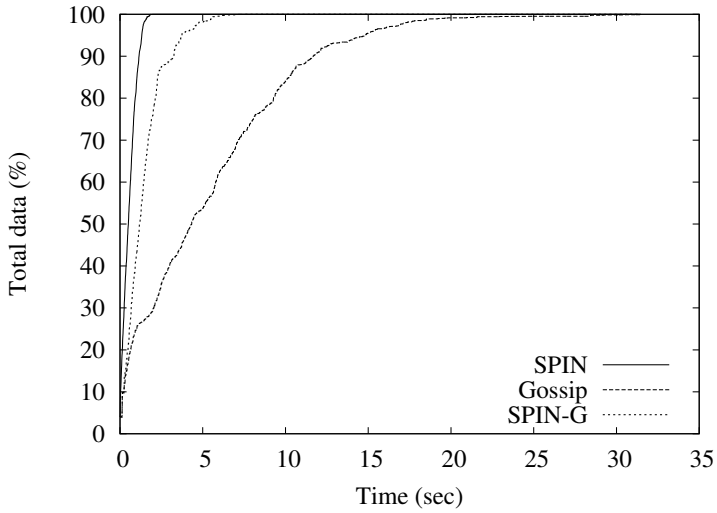


(a)

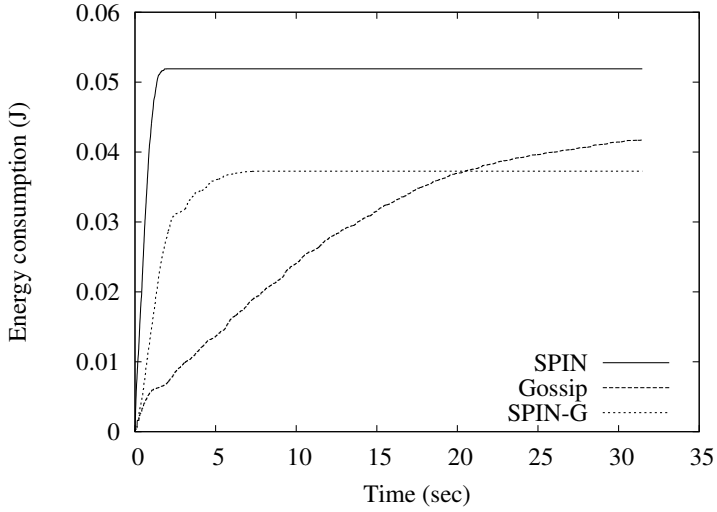


(b)

Figure 4.3: Performance of SPIN, SPIN-G, and gossip in a 25-node mesh network: (a) percent of the total data received ($\mathcal{D}_p$) by the network over time, (b) energy consumption ($\mathcal{E}_a$) by the network over time.



(a)



(b)

Figure 4.4: Performance of SPIN, SPIN-G, and gossip in a 25-node random network. (a) percent of the total data received ($\mathcal{D}_p$) by the network over time, (b) energy consumption ($\mathcal{E}_a$) by the network over time.

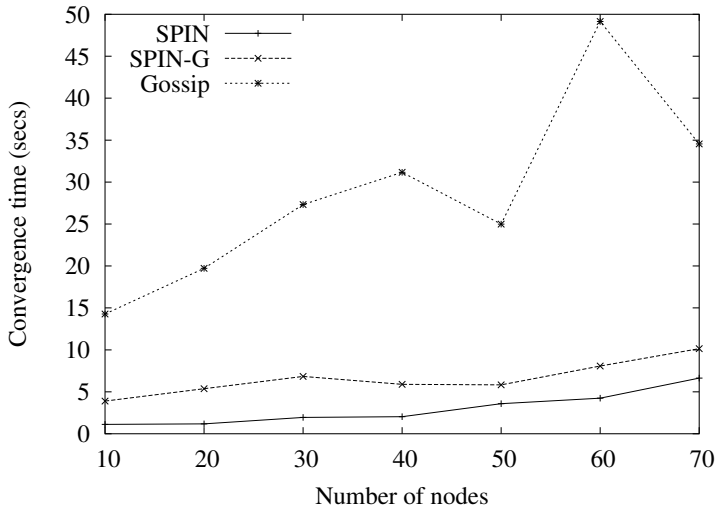
Although SPIN reduces the redundant data communication, it generates overhead of meta-data flooding. Gossip seems to consume least energy in a regular network, because the probability that the copy of data traverses throughout the regular and symmetric network is high. SPIN-G has a good balance of energy consumption and convergence time. It consumes energy close to gossip, with convergence time close to SPIN.

Figure 4.4(a)(b) shows the percent of the total data received ($\mathcal{D}_p$) and the energy consumption ($\mathcal{E}_a$) of all three protocols by all the sensor nodes in a 25-node random network over time. As expected, SPIN converges fastest at about 1.98 seconds with energy consumption of 0.052J; SPIN-G converges at about 7.52 seconds with energy consumption of 0.037J; and gossip converges at about 31.47 seconds with energy consumption of 0.042J. The network diameter of this 25-node random network is 9 compared to 7 of mesh network. Therefore, SPIN converges a little bit slower in random network than in mesh network. However, gossip converges much slower in random network than in mesh network. As described in section 4.3, in gossip there is only one copy of data flowing in the network. This flow may have to revisit some nodes several times to assure that every node in the network receives the data, resulting in convergence time varying to $O(n^3)$. In addition, the increase of the convergence time leads to exacerbated redundant meta-data communications in a random network reflecting in high energy consumption. This explains why SPIN-G consumes less energy than gossip protocol in random network. For SPIN-G, network topology does not impact convergence time and energy consumption much.

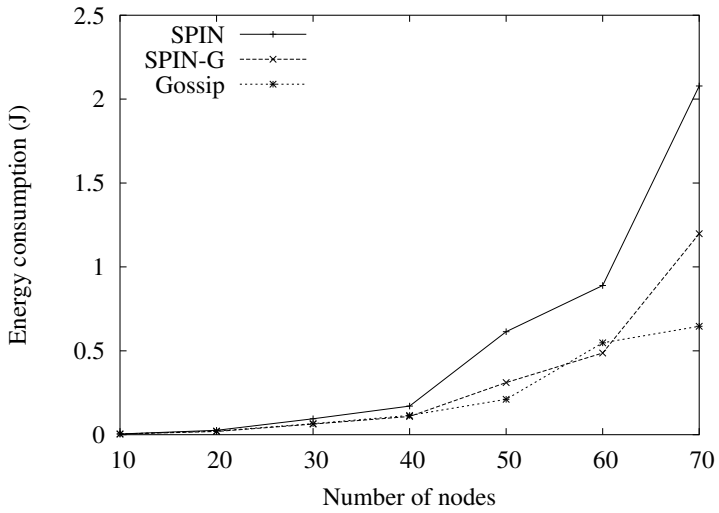
4.2.4 Vary Network Density

In this section, we study the effect of network size and density on SPIN, SPIN-G and gossip protocols. We randomly created 40 by 40 meters network varying number of nodes from 10 to 70.

Figure 4.5 (a) (b) shows the percent of the total data received ($\mathcal{D}_p$) and energy dissipation ($\mathcal{E}_a$) by all the sensor nodes varying number of nodes in the network from 10 to 70. As expected, with the increase of the network density, protocol convergence time increases. With the increase of the density, the average degree Δ increases. The collision among sensor nodes increases (proportion to Δ^2). As the number nodes increase from 10 to 70, the average degree (Δ) increases from 2.6 to 11.5. Since, the convergence times are proportional to Δ^2 for all protocols. We can see, for SPIN and SPIN-G protocols, the convergence time increases quadratically to Δ. However, for gossip protocol, the convergence time is more dependent on



(a)



(b)

Figure 4.5: Performance of SPIN, SPIN-G, and gossip varying number of nodes in a $40m \times 40m$ random network. (a) percent of the total data received ($\mathcal{D}_p$), (b) energy consumption ($\mathcal{E}_a$) over time.

the network topology and the number of nodes. Δ has less impact on convergence time. For the energy consumption, with the increase of the number of nodes, the energy consumption increases. Considering SPIN and SPIN-G, the difference of energy consumption is in overhead of meta-data negotiation. This is because SPIN is based on flooding on meta-data negotiation. With the increase of the network density, the implosion problem will be degenerated, flooding storm becomes inevitable. From equation 3, we can infer the energy consumption for meta-data negotiation is proportional to n^2, However, for SPIN-G, from equation 17, we can infer the energy consumption for meta-data negotiation is proportional to $n^{1.44}$. Therefore, with the increase of the network density, SPIN-G will save more energy than SPIN. From figure 4.5 (b), we can verify that as the number of nodes increased SPIN-G saves about 50% of energy than SPIN in 50-node network, compared to 20% savings in 20-node network. In addition, since gossip relies on the network topology and number of nodes in the network, with the increase of the density, the connectivity of the network also increases, gossip starts consume less energy than SPIN-G. Therefore, SPIN-G is more suitable for large-scale networks.

4.2.5 Impact of Sleeping-Active Cycle

In this section, we study the impact of sleeping mode on the performance of data dissemination protocols in terms of convergence time, energy consumption, and network partition time.

To protect the battery poor nodes in the network, thereby extending the lifetime of the network, we introduce sleeping mode in sensors. If the energy level of the sensor is below the low-level threshold (25%), the sensor node will enter an alternative status. The sensor node sleeps for a fixed interval, wakes up to communicate with other nodes, and then sleeps again, and so on. The purpose of this scheme is to try to protect the energy poor nodes. We realize that the node not only spends energy on transmitting data, but also on receiving data, even in overhearing data. If the battery poor nodes could sleep for a period of time, the other nodes in the network may take responsibility of forwarding data, thereby relieving the burden of battery poor nodes, and extending the network lifetime.

Figure 4.6 shows the convergence time and energy consumption of SPIN and SPIN-G protocols with and without sleeping-active cycle in 25-node random networks. The low-energy threshold is 25%, therefore there are average 5 nodes below threshold triggering sleeping-active cycle. From the figure, we can see using sleeping-active cycle, the convergence time increases and energy consumption reduces as compared

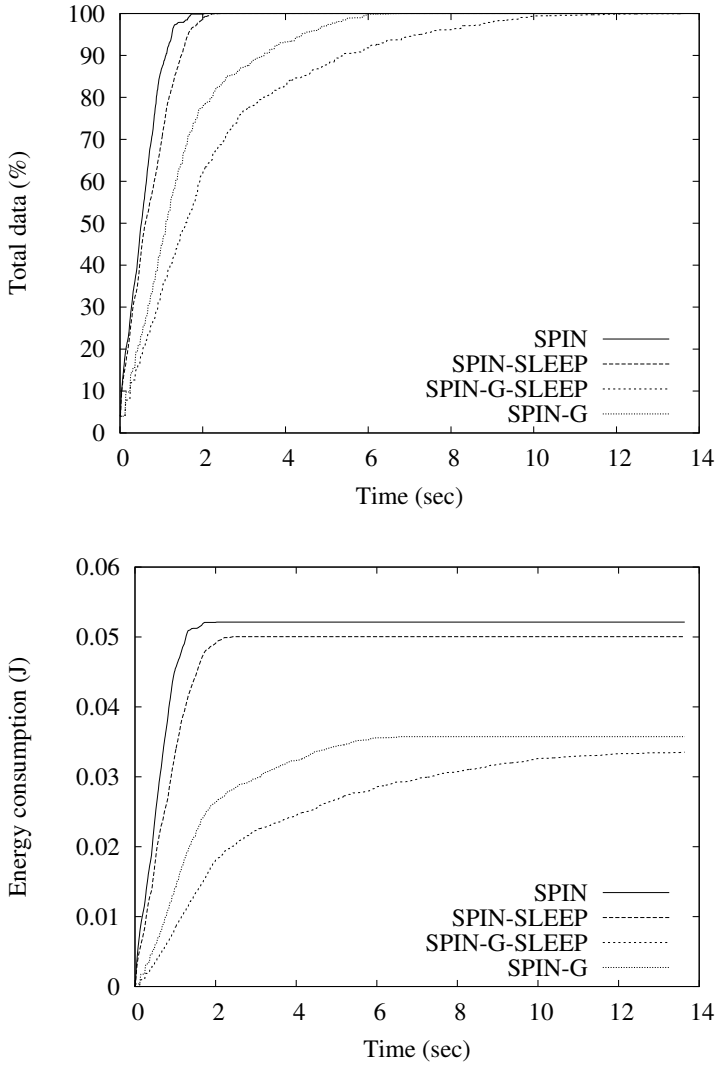


Figure 4.6: Performance of SPIN, SPIN-G, and gossip w/o sleeping/active cycle. (a) percent of the total data received ($\mathcal{D}_p$) by the network over time, (b) energy consumption ($\mathcal{E}_a$) by the network over time.

to without using sleep-active cycle. For SPIN-G protocol, the convergence time increases from 6.74 to 13.64 seconds, energy consumption reduced from 0.0357 to 0.0335 Joules. For SPIN protocol, the convergence time increases from 2.05 to 2.14 seconds, and energy consumption decreases from 0.052 to 0.050 Joules. Sleeping-active cycle has less impact on SPIN protocol than SPIN-G protocol. This is expected, since SPIN use flooding for advertisement. There are multiple copies of ADVs in the network, even some of the nodes entering sleeping mode, other nodes can take part in the data dissemination.

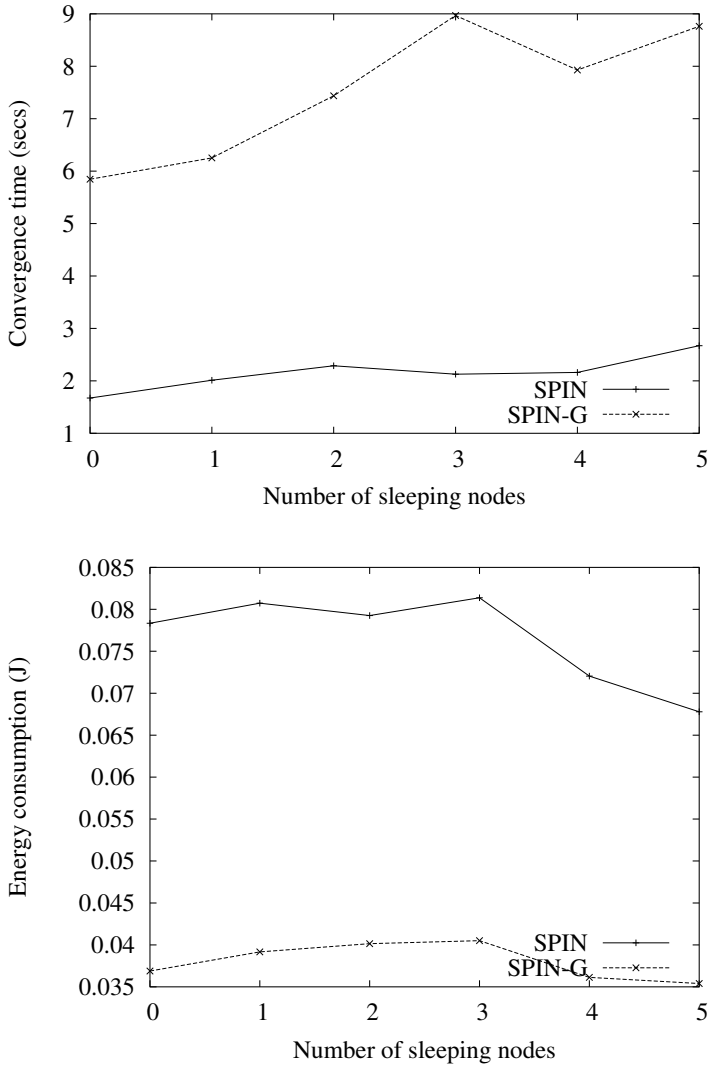


Figure 4.7: Performance of SPIN and SPIN-G varying number of sleeping nodes. (a) percent of the total data received ($\mathcal{D}_p$) by the network over time, (b) energy consumption ($\mathcal{E}_a$) by the network over time.

Figure 4.7 shows the effects of the number of sleeping node on performance. We vary the number of sleeping node from 1 to 5. With the increase of the number of nodes that have low battery power, the convergence time increases. This is expected. The energy consumption increases and then decreases. This is because with the increase of the sleeping node, we save energy for low energy level nodes at the expense of other nodes take part in the data dissemination. As the number of nodes further increase, the total energy consumption decreases.

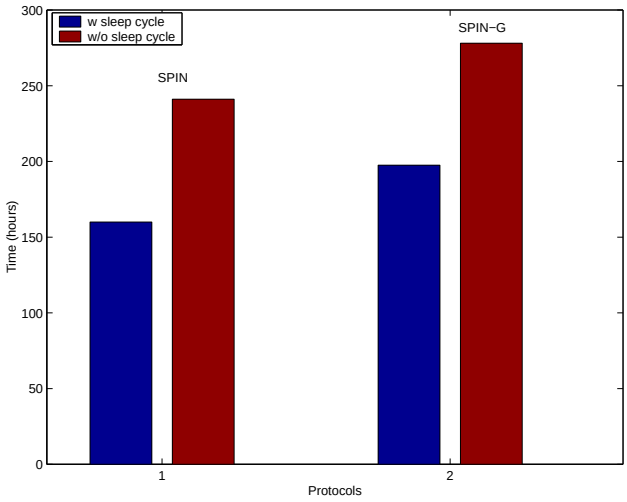


Figure 4.8: Network partition times of SPIN and SPIN-G w/o sleeping mode.

Figure 4.8 shows the results of network partition times of SPIN and SPIN-G protocol with and without trigger sleeping-active cycle for a 25-node random network. Each node has initial energy level varying from 10 Joules to 100 Joules. SPIN has 160 hours of network lifetime without sleeping-active cycle, and 241 hours with sleeping-active cycle. SPIN-G has 197 hours of network lifetime without sleeping-active cycle and 278 hours with sleeping-active cycle. Using sleeping-active cycle, SPIN has 50% of longer network lifetime, and SPIN-G has 40% of longer network lifetime. This is expected. We save same amount of energy in SPIN and SPIN-G. The network is disconnected when there are 2 nodes dead in the network. We do not show much energy savings when there are more battery-poor nodes.

In summary, deploying gossip and data aggregation, SPIN-G can exploit benefits of both SPIN and gossip. That is SPIN-G has energy consumption close to gossip and protocol convergence time close to SPIN. It can save energy compared to SPIN at a little expense of convergence time. With an increase in

network density, SPIN-G is much more energy-efficient. Besides, using sleeping-active cycle in battery poor nodes can further save energy, and extend the network lifetime at the expense of the increased convergence time.

4.3 Analysis

In this section, we provide an analysis for the convergence time and energy consumption of SPIN-G compared with SPIN and gossip protocol in case of a random network. The nodes in the network are placed at random in a rectangular area. Each node is battery-powered and has only a limited range of transmission r. Two nodes are neighbors if they are within the transmission range of each other. This type of random network is useful for modeling a large number of practical situation involving ad-hoc and sensor networks.

The sensor network can be modeled as a connected, undirected graph $G = (V, E)$, where V is a set of nodes, E is the set of pairs (i, j) where $i, j \in E$ if and only if j/i can be reached by node i/j with its transmission range.

Let,

n denotes the number of vertices in the graph G, $n = |V|$.

m denotes the number of edges of graph G, $m = |E|$.

D denotes the diameter of the graph; (The diameter of a graph is the longest of the shortest paths between any two vertices).

N_i denotes the set of vertices reachable by vertex i in G within its transmission range.

$|N_i|$ is the number of neighbors of node i.

Δ denotes average number of neighbors of each node in graph G. $\Delta = \frac{\sum_{i=1}^{n} |N_i|}{n}$

T_{mac} is the delay to access a channel, which is taken to be proportional to δ^2. Let C be the proportionality constant. Then $T_{mac} = C\delta^2$ [9].

Round r denotes one round where the nodes acquire data via three-way handshaking (ADV-REQ-DATA).

$\hat{S_m}$ denotes average packet size of ADV and REQ:

$$\hat{S_m} = S_m \times (n + 1)/2 \tag{4.3}$$

In general,

$$\begin{aligned} Total\ energy\ consumption &= energy\ consumption\ for\ receiving\ packet \\ &\quad + energy\ consumption\ for\ sending\ packet \end{aligned}$$

4.3.1 SPIN

The sensor nodes in SPIN acquire data from their neighbors through negotiation. Because SPIN uses flooding in meta-data advertising, it will converge in D rounds, here D is the diameter of the graph G. Let T_{spin} denotes the time duration of one round r, we get,

$$\begin{aligned} T_{spin} &= 3T_{mac} + \hat{S_m}/B + \hat{S_m}/B + S_d/B + 3T_p \\ &= 3T_{mac} + 2\hat{S_m}/B + S_d/B + 3T_p \end{aligned} \tag{4.4}$$

The convergence time C_{spin},

$$\begin{aligned} C_{spin} &= D \times T_{spin} \\ &= D \times (3C\Delta^2 + 2\hat{S_m}/B + S_d/B + 3T_p) \end{aligned} \tag{4.5}$$

For a 25-node random network with $D = 9$, $\Delta = 4.8$, $S_m = 16 + 52 = 68bytes = 544bits$, $\hat{S_m} = 2080bits$, $S_d = 500 + 52 = 552bytes = 4416bits$, $B = 2Mbps$, $T_p = 0.01s$, $C = 0.002$, we get $C_{spin} = 1.82s$.

Suppose there are l percent of battery-poor nodes in the network, the data will be lost with probability of $p = 1 - 1/2^{Dl}$. This lost data will be retransmitted if the node receives a new data or waits until a timeout T_{out}. In the presence of sleeping nodes, the convergence time will be:

$$\begin{aligned}\hat{C}_{spin} &= \sum_{i=0}^{\infty}(i \times T_{out} + C_{spin}) \times p^i \times (1-p) \\ &= \frac{T_{out} \times p}{(1-p)} + C_{spin} \end{aligned} \tag{4.6}$$

For energy consumption, because we use broadcast wireless network, for each data, there are n transmissions and Δn receptions of ADV message; $n-1$ transmissions and $\Delta(n-1)$ receptions of REQ message; and max($n/\Delta, D$) transmissions and $max(n/\Delta, D)\Delta$ receptions of DATA message. We assume each transmission informs Δ node. Therefore, the total energy consumption E_{spin} is,

$$\begin{aligned}E_{spin} &= n(E_{adv} + E_{req} + E_{data}) \\ &= n((E_T + E_R \times \Delta) \times n \times \hat{S_m} \\ &\quad +(E_T + E_R \times \Delta) \times (n-1) \times \hat{S_m} \\ &\quad +(E_T + E_R \times \Delta) \times (max(n/\Delta, D)) \times S_d) \end{aligned} \tag{4.7}$$

$$\begin{aligned}E_{spin} &= n((2n-1)\hat{S_m} + max(\frac{n}{\Delta}, D)S_d) \\ &\quad (E_T + E_R\Delta) \end{aligned} \tag{4.8}$$

For above 25-node random networks, suppose transmission power $P_t = 2.3uw$ for 10 meters range, receiver power $P_{xcvr} = 5mw$, we have $E_T = 5.0023mw/(2 \times 10^6 bps) = 2.50115 * 10^{-9} J/bit$, $E_R = 2.5 * 10^{-9} J/bit$.

So the energy consumption for distributing one data is 1.99×10^{-3}J, the total energy consumption of disseminating one data for all 25 nodes is $1.99 \times 10^{-3} \times 25 = 0.05J$.

4.3.2 Gossip

To analyze convergence time and energy consumption, we formalize it as a random work problem. Suppose there is a graph G, A random walk on G is a sequence of discrete steps: The process starts at a vertex v_0. We first randomly choose a neighbor of v_0 , say $v_1 \in N_0$, and walk along the edge incident on v_0 and to vertex v_1. At the second step, we proceed to a randomly chosen neighbor of v_1, and so on. Here, "randomly chosen neighbor" means a neighbor chosen uniformly at random; the chose at each step is independent of all previous choices [84].

"Random Walk" can be induced to a Markov chain M_g, the states of M_g are the vertices of G, and for any two vertices $u, v \in V$ [84].

$$p_{uv} = \begin{cases} 1/|N_u| \text{ , } if(u,v) \in E \\ 0 \quad \text{ , } \quad otherwise \end{cases}$$

Let $C_u(G)$ denote the expected number of steps taken by a random walk that starts at u and ends upon visiting every vertex in G at least once. The cover time of G, denote $C(G)$, is defined by $C(G) = max_{u \in E}(C_u(G))$. The best dissemination rate of gossip protocol for any specific graph is $1node/round$, $C_{gossip} \geq n$. Known theorem for cover time of random walk is $C(G) \leq 2m(n-1)$. We get, $n \leq C(G) \leq 2m(n-1) = \Delta n(n-1)$ [84].

$$C_{gossip} = C(G) * T_{gossip} \tag{4.9}$$

$$T_{gossip} = T_{mac} + S_d/B + T_p \tag{4.10}$$

For broadcast networks we get:

$$max(n/\Delta, n)(T_{mac} + S_d/B + T_p) \leq C_{gossip} \leq \Delta n(n-1)(T_{mac} + S_d/B + T_p) \tag{4.11}$$

With above parameters of 25-node network, we get

$$1.45s \leq C_{gossip} \leq 167.87s \tag{4.12}$$

The convergence time of gossip somewhat depends on the network topology, the fastest case usually correspond to dense, highly connected graphs, for example, the complete graph, d-regular graphs with $d > |N|/1$, and hypercube. With the decrease of the connectivity and existence of bottleneck node in the network, the cover time of gossip will increase [85], [86].

In gossip, each step of random walk needs one transmission and one reception. Therefore, we get E_{gossip},

$$E_{gossip} = nC(G)(E_T + E_R\Delta)S_d \tag{4.13}$$

$$max(n/\Delta, D)n(E_T + E_R\Delta)S_d \leq E_{gossip} \leq \Delta n^2(n-1)(E_T + E_R\Delta)S_d \tag{4.14}$$

For 25-node random network, we get,

$$0.013J \quad \leq \quad E_{gossip} \leq 4.6J \tag{4.15}$$

For each of 25 nodes in network disseminating one data, we get energy consumption is between 0.038J and 4.5J

From the result, we can see that if gossip can converge in a short period time, the energy consumption will be low. However, with the increase of the convergence time, gossip protocol would end up with high energy consumption than SPIN protocol.

4.3.3 SPIN-G

The important feature of SPIN-G is using randomized algorithm of randomly choosing one neighbor for advertising and data aggregation scheme. Therefore, SPIN-G can be formalized as a "gossiping" problem. Gossiping refers to the information dissemination problem that each node in a graph G knows a unique piece of information and must transmit to every other node in G [87]. While information is spreading in G, each vertex can only communicate with a subset of other vertices that are neighboring to it. Gossiping and random walk are different problems. In gossiping, after receiving a data, both sender and receiver can spread the data through a randomly chosen neighbor. However, in random walk, there is only one copy of data flowing in the network. After receiving a data, only receiver can spread the data to one of its neighbors.

The minimum total time required $S(G)$ for distributing data to all the nodes in any graph with n nodes is $log_\rho n$ where the logarithm is in the base of the golden ratio $\rho = (1+\sqrt{5})/2$, so $log_\rho n = 1.44 log_2 n$ [88].

$$S(G) = max(1.44 log_2 n, D) \tag{4.16}$$

$$C_{spin-g} = S(G) * T_{spin-g} \tag{4.17}$$

$$\begin{aligned} T_{spin-g} &= 3T_{mac} + 2\hat{S_m}/B + S_d/B + 3T_p + T_{out} \\ &= 3C\Delta^2 + 2\hat{S_m}/B + S_d/B + 3T_p + T_{out} \end{aligned} \tag{4.18}$$

$$C_{spin-g} = max(1.44 log_2 n, D) * (3C\Delta^2 + 2\hat{S_m}/B + S_d/B + 3T_p + T_{out}) \tag{4.19}$$

For a 25-node random network,

$$C_{spin-g} = 3.17s \tag{4.20}$$

Since the all nodes get informed at time $S(G)$ with high probability, the convergence time of C_{spin-g} close to this lower bound [89].

SPIN-G needs $1 + 2 + 4 + \ldots + 1.44log_2n = 2^{1.44log_2n+1} - 1$ transmissions of ADVs before all nodes get informed. Therefore, for ADV, there are $2^{1.44log_2n+1} - 1$ transmissions and receptions; for REQ, there are $n - 1$ transmissions and receptions for each data; and $max(n/\Delta, d)$ transmissions and receptions of DATA messages.

$$\begin{aligned} E_{spin-g} &= (2^{1.44log_2n+1} - 1)(E_T + \Delta E_R)\hat{S_m} \\ &\quad +n(n-1)(E_T + \Delta E_R)\hat{S_m} \\ &\quad +max(n/\Delta, D)n(E_T + \Delta E_R)S_d \\ &= ((2^{1.44log_2n+1} + n^2 - n - 1)\hat{S_m} \\ &\quad +max(n/\Delta, D)nS_d)(E_T + \Delta E_R) \end{aligned} \tag{4.21}$$

For 25-node network, we get

$$\begin{aligned} E_{spin-g} &= ((2^{1.44log_225+1} + 25^2 - 25 - 1) \times 2080 \\ &\quad +25 \times 9 \times 4416) \\ &\quad \times (2.50115 + 2.5 \times 4.8) \times 10^{-9} \\ &= 0.038J \end{aligned} \tag{4.22}$$

Since SPIN-G employs data requisition scheme that each node waits for a fix interval for multiple ADVs

and ask for data from neighboring node with lowest energy level. Each node has more possibility to assembly multiple REQ and DATA into one packet. Therefore, SPIN-G should consume less energy than this calculated result.

In summary, compared with SPIN, SPIN-G converges slower than SPIN with lower energy consumption. The performance of all three protocols depends on the network topology. SPIN relies on the network diameters, SPIN-G and gossip relies on many metrics of the network, number of nodes and edges in the network, connectivity of the network, etc.

4.4 Summary

In this chapter, we presented a data dissemination protocol using gossip (SPIN-G) that exploring several energy saving schemes. It is motivated by SPIN protocol, which uses meta-data negotiation to reduce the redundant data communication. SPIN-G keeps the negotiation feature of SPIN. However, instead of using flooding for meta-data negotiation, SPIN-G employed randomized gossiping, where the sensor node randomly selects one neighbor for advertising, to further reduce the overhead of negotiation. Moreover, SPIN-G also has a data requisition strategy, where a node chooses its neighbor with the most energy for requesting data. This makes the node adapt its behavior based on the energy levels of its neighbors. This resource awareness provides more balanced energy consumption across the network, leading to better network lifetime. To extend the connectivity of the network, a sleeping-active cycle is deployed to battery poor nodes to further extend the system lifetime. Our performance study shows the tradeoff between network convergence time and energy consumption. Although convergence time of SPIN-G is slightly slower than SPIN, SPIN-G can save about 20% energy than SPIN. With the increase of the network density, SPIN-G is much more energy efficient (i.e. 40% energy saving for 50-node network). Besides, using sleeping-active cycle, we can gain 40% longer system lifetime with the proposed protocol.

CHAPTER FIVE

CONCLUSIONS

The use of wireless channel is growing at an amazing speed. With the advances in wireless communication techniques, we can envision that some day we can achieve the goal of ”anytime and anywhere” communication among and between users and devices. However, bandwidth and energy limitation in wireless ad-hoc and sensor networks can affect the ease of the communication among them. In addition, time-varying and dynamic condition of the system may influence the performance of the protocol. Therefore, it is important to design new communication protocols that can operate efficiently in such a challenging environment.

In this chapter we conclude this dissertation by summarizing the research discussed in the previous chapters, followed by a section on direction for future research.

5.1 Conclusions

In this dissertation, two communication protocols are proposed and discussed: (i) TCP-Manet (TCP Enhancement) for wireless ad-hoc networks, and (ii) SPIN-G (Energy-Aware Data Dissemination Protocol) for wireless sensor networks.

5.1.1 TCP-Manet

TCP-Manet adds new mechanisms to the traditional TCP to determine the nature of the packet losses - congestion error, wireless link error, or network misbehavior etc. using cross layer design. Then TCP-Manet can trigger different kinds of recovery strategies to achieve better throughput over the wireless ad-hoc networks.

Once TCP-Manet sender detects a packet loss by three duplicate acknowledgments, it checks the trend of “power”. If it is in an increasing trend, this error is considered as a wireless link error, the sender retransmits the packet without reducing the congestion window size. Otherwise TCP-Manet sender enters fast retransmission using new congestion control and avoidance algorithm. If timeouts occur, the sender retransmits the packet while holding the congestion window size unchanged. If the sender gets new acknowledgement, which means it is a congestion error, TCP sender will set congestion window size to 1. Otherwise, after four timeouts, TCP sender sends probe messages to the receiver to identify if there is a selfish node in the

connection.

Theoretical analysis is provided for TCP-Manet. Simulation is carried out. Throughput, fairness, backward compatibility, and detection effectiveness are carefully studied and compared with the theoretical analysis. The theoretical analysis and simulation results demonstrate that TCP-Manet has better performance than traditional TCP over the wireless ad-hoc networks.

The key contributions are following:

- Improved Throughput. TCP-Manet includes detection functionalities that can determine the reasons of packet loss, and then trigger corresponding recovery strategies.
- Fairness and Friendliness. TCP-Manet remains fairness and friendliness of traditional TCP protocol.
- Sender Side Only Modification. Like original TCP design, TCP-Manet uses end-to-end mechanism, which makes TCP-Manet compatible to other TCP-variant and easy to deploy.
- Cross Layer Design. By interacting with lower level protocol, TCP can access more network information to help it react to the variance of the network conditions.

5.1.2 SPIN-G

Disseminating data among sensors is a fundamental operation in energy-constrained wireless sensor networks. SPIN-G is a gossip-based adaptive protocol that introduces many energy saving mechanisms to extend the network lifetime of the sensor networks: It uses meta-data to name the data and uses negotiation to eliminate redundant transmissions of duplicate data in the network; It adapts gossiping with data dissemination protocol to reduce the overhead of meta-data communications; The sensor nodes in SPIN choose the most energetic neighbor for requesting data therefore balance network consumption distributions throughout the network; Moreover, the battery poor node in the network will fall into sleep cycle to further save their energy, therefore keep the connectivity of the network.

The performance of this system is analyzed by simulation and theory. Two performance metrics are studied - protocol convergence time and energy consumption throughout the network, while varying the network nodes in the network. In addition, the performance comparison with SPIN and gossip is made. The results show that SPIN-G save more energy ands has longer lifetime than SPIN.

The key contributions are following:

- Simple Design. SPIN-G explores randomized algorithm in wireless sensor network protocol design. Randomized algorithm is characterized as simple and robust, which is suitable for resource constrained sensor networks.
- Extended System Lifetime. Sensor nodes are battery operated. SPIN-G deploys several energy-saving mechanisms to reduce the energy consumption during the data dissemination operation.
- Acceptable Protocol Convergence Time. Although gossip introduces non-deterministic behavior that may prolong the data distribution speed, SPIN-G uses data-aggregation mechanism scheme that can make the protocol converge in acceptable time.

5.2 Future Work

Two communication protocols are introduced in the dissertation to address the problem brought today's wireless ad-hoc and sensor network systems. There are still many open problems related to this kind of system.

5.2.1 TCP-Manet

TCP-Manet uses the trend of "power" to help differentiate wireless link error on packet losses. In simulation, we found that power fluctuates frequently. This hurts the measurements of the trend of power. For future work, we could use trend of "moving power" instead of "instantiate power". Besides, due to the dynamic behavior of the ad-hoc networks, it is helpful to identify more metrics that could be used to detect the nature of the error. The more metrics we use, the more precise and robust of the detection mechanism.

The simulation results explore the effectiveness of selfish nodes' detection. We considered the conditions of selfish node only case and selfish node with random packet losses case. In ad-hoc networks, the host can move following some patterns. For example, some of the mobile nodes whose movements are independent of each other; some of the mobile nodes whose movements are dependent each other [90]. The further study of TCP-Manet could include study of how different mobility models affect the effectiveness of selfish nodes' detection.

TCP-Manet is an enhancement based on TCP-Reno. TCP protocols have different variants, such as TCP-Vegas, TCP-SACK, and TCP-NewReno etc. In the future, we will examine if the mechanisms used in TCP-Manet can be also deployed in other TCP flavors, and how they perform.

5.2.2 SPIN-G

Most data dissemination protocols do not consider the reliable data dissemination of the data. This is because the vast majority of sensor network applications do not require reliable data delivery [91]. However, we believe that some days in the future, reliable data distribution will also be a requirement of the design of protocols in sensor networks.

Sensor networks are usually application specific. Due to its distinguishing characteristics, general purpose protocol architecture, such as traditional TCP/IP stack may not appropriate. Currently, most sensor networks deploy application specific protocols that can exploit features of the application to achieve greater performance. However, we believe that as the sensor networks become more and more popular, it is required to architect a tunable network stack to meet the demands of sensor network applications.

Randomized algorithm is characterized as simplicity and scalability. Hence, it will have more potential use in the protocol design in sensor network which are featured as low battery, computation capability. However, using randomized algorithm will affects the reliability and responsiveness of the protocol. Therefore, it requires new design that can compensate this problem.

Putting nodes into sleeping mode is one of the effective ways of saving energy. In the SPIN-G protocol, instead of making node go to sleep when energy is low, we can apply a more aggressive energy saving scheme that let nodes go to sleep randomly. However, as a trade-off of deploying sleep mode in sensor node, system delay will also increase. To alleviate this problem, before going to sleep, sensor nodes may send out a message telling other nodes its pending unavailability. Other nodes may recalculate their neighboring tables to reflect the change.

BIBLIOGRAPHY

[1] R. Jain and K. K. Ramakrishnan, "Congestion Avoidance in Computer Networks with a Connectionless Network Layer: Concepts, Goals, and Methodology," in *Proceedings of the Computer Networking Symposium, IEEE*, pp. 134–143, April 1988.

[2] P. Rysavy, "The Road to a Wireless Future," *Network Computing*, October 2005.

[3] H. Balakrishnan, V. N. 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, no. 6, pp. 756–769, 1997.

[4] A. S. Tanenbaum, *Computer Networks*. fourth edition Prentice Hall PTR, 2003.

[5] A. Chockalingam, M. Zorzi, and R. Rao, "Performance of TCP on Wireless Fading Links with Memory," in *Proceedings of the IEEE ICC'98*, June 1998.

[6] V. Tsaossidis, H. Badr, X. Ge, and K. Pentikousis, "Energy/Throughput Tradeoffs of TCP Error Cotnrol Strategies," in *Proceedings of the 5th IEEE Symposium on Computers and Communications (ISCC)*, 2000.

[7] T. Lakshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE/ACM Transactions on Networking*, pp. 336–350, June 1997.

[8] G. Khanna and Y.-S. Wu, "Data Dissemination Protocol in Sensor Networks to Tolerate and Link Failures." http://dynamo.ecn.purdue.edu/ sbagchi/Research/Papers/spms_dsn03fastabls.pdf.

[9] D. Khanna, S. Bagchi, and Y.-S. Wu, "Fault Tolerant Energy Aware Data Dissemination Protocol in Sensor Networks," in *Proceedings of 2004 international conference on dependable systems and networks (DSN'04)*, (Florence, Italy), pp. 795–804, Jun. 28 - Jul. 1 2004.

[10] J. Kulik, W. heinzelman, and H. Balakrishanan, "Negotiation-Based Protocols for Dissemination Information in Wireless Sensor Networks," *Wireless Networks*, 2002.

[11] J. Kulik, W. Rabiner, and H. Balakrishnan, “Adaptive Protocols for Information Dissemination in Wireless Sensor Networks,” in *Proceedings of 5th annual ACM/IEEE international conference on mobile computing and networking (MOBICOM’99)*, (Seattle, WA, USA), pp. 174–185, 1999.

[12] A. Sinha and A. Chandrakasan, “Dynamic Power Management in Wireless Sensor Networks,” *IEEE Design and Test of Computers Magazine*, vol. 18, pp. 62–74, Mar. - Apr. 2001.

[13] G. Holland and N. Vaidya, “Analysis of TCP Performance over Mobile Ad Hoc Networks,” *PhD Thesis*, August 1999.

[14] J. Liu and S. Singh, “ATCP: TCP for Mobile Ad Hoc Networks,” *IEEE Journal on Selected Areas in Communications, J-SAC*, vol. 19, no. 7, pp. 1300–1315, 2001.

[15] T. Goff, J. Moronski, and D. Phatak, “Freeze-TCP: A True End-to-End Enhancement Mechanism for Mobile Environments,” in *Proceedings of the INFOCOM*, 2000.

[16] K. Ratnam and I. Matta, “WTCP: An Efficient Mechanism for Improving TCP Performance over Wireless Links,” in *Proceedings of the Third IEEE Symposium on Computer and Communications (ISCC98)*, June 1998.

[17] G. Xylomenos, G. C. Polyzos, P. Mahonen, and M. Saaranen, “TCP Performance Issues over Wireless Links,” *IEEE Communi. Magazine*, vol. 39, pp. 52–58, April 2001.

[18] X. Zhang, S. Wu, Z. Fu, and T.-L. Wu, “Malicious Packet Dropping: How It might Impact the TCP Performance and How We can Detect It,” in *Proceedings of International Conference on Network Protocols*, pp. 263–272, November 2000.

[19] R. de Oliveria and T. Braun, “TCP in Wireless Mobile Ad-Hoc Networks,” Tech. Rep. IAM-02-003, July 2002.

[20] A. Bakne and B. R. Badrinath, “I-TCP: Indirect TCP for Mobile Hosts,” in *Proc. 15th Int’l Conf. on Distr. Computer Systems, IEEE*, pp. 136–143, 1995.

[21] K. Brown and S. Singh, “M-TCP: TCP for Mobile Cellular Networks,” pp. 19–43, 1997.

[22] B. S. Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan, “Improving Performance of TCP over Wireless Networks,” Tech. Rep. Technical Report 96-014, Texas A&M University, 1996.

[23] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz, “Improving TCP/IP Performance over Wireless Networks,” in *Proceedings of the 1st ACM Int’l Conf. On Mobile Computing and Networking (Mobicom)*, November 1995.

[24] R. Caceres and L. Iftode, “Improving the Performance of Reliable Transport Protocol in Mobile Computing Environment,” *IEEE Journal of Selected Areas in Communications*, vol. 13, June 1995.

[25] C. Barakat, E. Altman, and W. Dabbous, “TCP Performance in a Heterogeneous Networks: A survey,” *IEEE Communi. Magazine*, vol. 38, pp. 40–46, January 2000.

[26] N. Ghani and S. Dixit, “TCP/IP Enhancements for Satellite Networks,” *IEEE Commun. Magazine*, vol. 37, pp. 64–72, 1999.

[27] G. Huston, “TCP in a Wireless World,” *IEEE Internet Computing*, vol. 5, pp. 82–84, March-April 2001.

[28] V. Tsaoussidis and I. Matta, “Open Issues on TCP for Mobile Computing,” *Wireless Communications and Mobile Computing - Special issue on Reliable Transport Protocols for Mobile Computing*, February 2002.

[29] K. Ramakrishnan and S. Floyd, “A Proposal to Add Explicit Congestion Notification (ECN) to IP. RFC 2481,” January 1999.

[30] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, “A Feedback-based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks,” *Personal Communications, IEEE [see also IEEE Wireless Communications]*, vol. 8, Feb 2001.

[31] T. Dyer and R. Boppana, “A Comparison of TCP Performance over Three Routing Protocols for Mobile Ad Hoc Networks,” in *Proceedings of the 2nd ACM Symposium on Mobile Ad Hoc Networking and Computing*, pp. 55–66, October 2001.

[32] F. Wang and Y. Zhang, "Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response," in *proceedings of MOBIHOC'02*, pp. 217–225, June 2002.

[33] S. Biaz and N. Vaidya, "Discriminating Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver," *IEEE Symposium on Application - Specific Systems and Software Designing and Technology, ASSET'99*, 1999.

[34] S. Cen, P. C. Cosman, and G. M. Voelker, "End-to-end Differentiation of Congestion and Wireless Losses," *IEEE Trans. on Networking*, pp. 703–717, October 2003.

[35] C. Zhang and V. Tsaoussidis, "TCP Real: Improving Real-time Capabilities of TCP over Heterogeneous Networks," in *Proceedings of the 11th IEEE/ACM NOSSDAV 2001*, 2001.

[36] M. Gerla, M. Y. Sanadidi, R. Wang, and A. Zanella, "TCP Westwood: Congestion Window Control using Bandwidth Estimation," in *Proceedings IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 3(25-29), pp. 1698–1702, 2001.

[37] A. Zanella, G. Procissi, M. Gerla, and M. Sanadidi, "TCP Westwood: Analytic Model and Performance Evaluation," in *Proceedings of IEEE Globecom*, pp. 1703–1707, 2001.

[38] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," in *Mobile Computing and Networking*, pp. 287–297, 2001.

[39] S. Baucke, "Using Receiver-based Rate Matching for Congestion Detectin in Rate-based Protocols," in *Wireless Communication and Networking, 2003, WCNC 2003. 2003 IEEE*, vol. 3, pp. 1784–1789, 2003.

[40] C. Parsa and G. Aceves, "Improving TCP Congestion Control over Internets with Heterogeneous Transmission Media," in *IEEE International Conference on Network Protocols (ICNP 99)*, 1999.

[41] H. Sabbineni and K. Chakrabarty, "Location-Aided Flooding: An Energy-Efficient Data Dissemination Protocol for Wireless Sensor Networks," *IEEE Transaction on Computers*, vol. 54, pp. 36–46, January 2005.

[42] J. W. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," in *Proceedings of the 2nd international conference on Embedded networked sensor systems, conference on Embeded Network Sensor Systems*, pp. 81–94, 2004.

[43] S. S. Kulkarni and M. Arumugam, "Infuse: A TDMA Based Data Dissemination Protocol for Sensor Networks," Tech. Rep. Technical Report MSU-CSE-04-46, Michigan State Unifersity, 1990.

[44] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in *Proceedings of the 6th ACM/IEEE international conference on Mobile computing and networking (MOBICOM'00)*, (Boston, MA, USA), pp. 56 – 67, Aug. 2000.

[45] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A Two-Tier Data Dissemination Model for Large-scale Wireless Sensor Networks," in *Proceedings of the 8th ACM/IEEE international conference on Mobile computing and networking (MOBICOM'02)*, (Atlanta, GA, USA), pp. 148 – 159, Sept. 23 - 28 2002.

[46] F. Ye, G. Zhong, S. Lu, and L. Zhang, "GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks," in *Proceedings of IEEE international workshop on information processing in sensor networks (IPSN)*, 2003.

[47] T. Bokareva, N. Bulusu, and S. Jha, "A Performance Comparison of Data Dissemination Protocols for Wireless Sensor Networks," in *Proceedings of IEEE GLOBECOM wireless ad hoc and sensor networks workshop*, (Dallas, TX, USA), November 2004.

[48] "NS-2 Network Simulator." http://www-mash.cs.berkeley.edu/ns/.

[49] C. Avin and G. Ercal, "Bounds on Mixing Time and Partial Cover of Ad-hoc and Sensor Networks," Tech. Rep. Technical Report 040028, UCLA, June 2004.

[50] D. Braginsky and D. Estrin, "Rumor Routing Algorithm for Sensor Networks," in *Proceedings of first international workshop on sensor networks and applications (in conjunction with ACM Mobicom'02)*, pp. 22 – 31, 2002.

[51] C.-F. Chiasserini and M. Garetto, "Modeling the Performance of Wireless Sensor Networks," in *INFOCOM 2004, Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 1, March 2004.

[52] J. V. Greunen, D. Petrović, A. Bonivento, J. Rabaey, K. Ramchandran, and A. Sangiovanni-Vincentelli, "Adaptive Sleep Discipline for Energy Conservation and Robustness in Dense Sensor Networks," in *2004 IEEE International Conference on Communications, (ICC 2004)*, vol. 6, pp. 3657–3662, June 2004.

[53] U. Cetintemel, A. Flinders, and Y. Sun, "Power-Efficient Data Dissemination in Wireless Sensor Networks," in *Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access (MobiDE'03)*, (San Diego, CA, USA), pp. 1–8, Sept. 9 2003.

[54] R. Griswold and S. Medidi, "Malicious Node Detection in Ad Hoc Wireless Networks," in *Proceedings of SPIE AeroSense Conference on Digital Wireless Communications, 5100*, April 2003.

[55] S. R. Medidi, M. Medidi, and S. Gavini, "Detecting Packet-dropping Faults in Mobile Ad-hoc Networks," in *proceedings of IEEE ASILOMAR Conference on Signals, Systems and Computers (ASILOMAR)*, November 2003.

[56] V. Jacobson, "Congestion Avoidance and Control," in *Proc. SIGCOMM88 Conf. ACM*, pp. 314–329, 1988.

[57] M. Jain and C. Dovrolis, "End-to-end Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," in *proceedings of ACM SIGCOMM*, pp. 295–308, 2002.

[58] R. Jain, "A Delay-based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks," *Computer Communication Review*, vol. 19, pp. 56–71, October 1989.

[59] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," in *proceedings ACM SIGCOMM*, pp. 24–35, August 1994.

[60] Z. Wang and J. Crowcroft, "Eliminating Periodic Packet Losses in the 4.3-Tahoe BSD TCP Congestion Control Algorithm," *Computer Communication Review*, vol. 22, pp. 9–16, April 1992.

[61] J. Martin, A. Nilsson, and I. Rhee, “Delay-Based Congestion Avoidance for TCP,” *IEEE/ACM Transaction on Networking*, vol. 11, pp. 356–369, June 2003.

[62] A. Giessler, J. Haanle, A. Konig, and E. Pade, “Free Buffer Allocation - An Investigation by Simulation,” *Computer Networks*, vol. 1, pp. 191–204, July 1978.

[63] L. Kleinrock, “Power and Deterministic in Congestion Control Techniques,” in *proceedings of international conference communication*, pp. 43.1.1–10, June 1979.

[64] M. Hollick, J. Schmitt, C. Seipl, and R. Steinmetz, “On the Effect of Node Misbehavior in Ad Hoc Networks,” in *Proceedings of IEEE International Conference on Communications, ICC’04*, vol. 6, pp. 3759–3763, June 2004.

[65] A. Cardenas, S. Radosavac, and J. S. Baras, “Detection and Prevention of MAC Layer Misbehavior for Ad-Hoc Networks,” in *2004 ACM Workship on Security of Ad-hoc and Sensor Networks (SASN2004)*, pp. 17–22, 2004.

[66] S. Vassilaras, D. Vogiatzis, and G. S. Yovanof, “Misbehavior Detection in Clustered Ad-hoc Networks with Centrol Control,” in *Proceedings of International Conference on Information Technology : Coding and Computing (ITCC’05)*, vol. 2, pp. 687–692, 2005.

[67] J. padhye, V. Firoiu, D. Towsley, and J. Kurose, “Modeling TCP Throughput: A Simple Model and its Empirical Validation,” in *ACM SIGCOMM’98 Conference on applications, technologies, architectures, and protocols for computer communication*, pp. 303–314, 1998.

[68] V. Anantharaman and R. Sivakumar, “A Microscopic Analysis of TCP Performance over Wireless Ad-hoc Networks,” in *proceedings of ACM SIGMETRICS ’02*, June 2002.

[69] A. A. Abouzeid and S. Roy, “Stochastic Modeling of TCP in Networks with Abrupt Delay Variations,” *Wireless Networks*, vol. 9, pp. 509–524, September 2003.

[70] N. Cardwell, S. Savage, and T. Anderson, “Modeling TCP Latency,” in *INFOCOM’00*, pp. 1742–1751, 2000.

[71] C. B. Samios and M. K. Vernon, “Modeling the Throughput of TCP Vegas,” in *Joint International Conference on Measurement and Modeling of Computer Systems, proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 71–81, 2003.

[72] B. Sikdar, S. Kalyanaraman, and K. S. Vastola, “Analytic Model for the Latency and Steady-State Throughput of TCP Tahoe, Reno, and SACK,” *IEEE/ACM Transactions on Networking (TON)*, vol. 11, pp. 959–971, December 2003.

[73] U. of Southern California Information Sciences Institute (USC/ISI), “The Network Simulator: NS-2. Computer Software.”

[74] Y.-T. Li, D. Leith, and R. N. Shorten, “Experimental Evaluation of TCP Protocols for High-Speed networks.”

[75] S. Floyd, “Metrics for the Evluation of Congestion Control Mechnanism - Internet Engineering Task Force Internet-Draft.”

[76] D. X. Wei, “Benchmark for TCP Congestion Control.”

[77] E. Hahne and R. Gallager, “Round Robin Scheduling for Fair Flow Control in Data Communications Networks,” in *IEEE International Conference on Communications*, June 1986.

[78] F. Kelly, A. Maulloo, and D. Tan, “Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability,” *Journal of the operational research society*, vol. 49, pp. 237–252, 1998.

[79] F. Kelly, “Mathematical Modelling of the Internet,” *Mathematics Unlimited - 2001 and Beyond (Editors B. Engquist and W. Schmid)*, pp. 685–702, 2001.

[80] R.Jain, D. M. Chiu, and W. Hawe, “A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems,” tech. rep., 1984.

[81] K. Bharath-Kumar and J. Jeffrey, “a New Approach to Performance-Oriented Flow Control,” *IEEE Transaction on Communications*, vol. COM-29, April 1981.

[82] “Wikipedia, the Free Encyclopedia.”

[83] S. Tilak, A. Murphy, and W. Heinzelman, “Non-uniform Information Dissemination for Sensor Networks,” in *Proceedings of the 11th IEEE International Conference on Network Protocol (ICNP’03)*, (Atlanta, GA, USA), pp. 295 – 304, Nov. 4 - 7 2003.

[84] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

[85] C. Avin and G. Ercal, “Bounds on the Mixing Time and Partial Cover of Ad-Hoc and Sensor Networks,” in *Second European Workshop on Wireless Sensor Networks (EWSN)*, January 31 - February 2 2005.

[86] C. Avin and G. Ercal, “Bounds on the Mixing Time and Partial Cover of Ad-Hoc and Sensor Networks,” Tech. Rep. Technical Report CSD-TR 040028, UCLA, 2004.

[87] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman, “A Survey of Gossiping and Broadcasting in Communication Networks,” *Networks*, vol. 18, pp. 319 – 349, 1988.

[88] D. W. Krumme, G. Cybenko, and K. N. Venkataraman, “Gossiping in Minimal Time,” Tech. Rep. Technical Report 1027, Center of Supercomputing Research and Development, August 1990.

[89] R. M. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, “Randomized Rumor Spreading,” in *IEEE Symposium on Foundations of Computer Science*, pp. 565–574, 2000.

[90] T. Camp, J. Boleng, and V. Davies, “A Survey of Mobility Models for Ad Hoc Network Research,” *Wireless Communication & Mobile Computing (WCMC): A special issue on Mobile Ad Hoc Networking: Research, Trends and Applications* , vol. 2, no. 5, pp. 483–502, 2002.

[91] S. Palchauduri and R. Kumar, “Case for a New Sensor Network Stack Architecture.”