IMPROVING SYSTEM PERFORMANCE FOR WIRELESS NETWORKS

Except where reference is made to the work of others, the work described in this dissertation is my own or was done in collaboration with my advisory committee. This dissertation does not include proprietary or classified information.

Fangyang Shen

Certificate of Approval:

Kai Chang
Professor
Computer Science and Software Engineering
Auburn University

Min-Te Sun, Chair
Assistant Professor
Computer Science and Software Engineering

Yu Wang
Assistant Professor
Computer Science and Software Engineering

Shiwen Mao
Assistant Professor
Electrical and Computer Engineering

Xiao Qin
Assistant Professor
Computer Science and Software Engineering

George T. Flowers
Interim Dean
Graduate School

IMPROVING SYSTEM PERFORMANCE FOR WIRELESS NETWORKS

Fangyang Shen

A Dissertation

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Doctor of Philosophy

Auburn, Alabama
August 9, 2008

IMPROVING SYSTEM PERFORMANCE FOR WIRELESS NETWORKS

Fangyang Shen

_____

Signature of Author

_____

Date of Graduation

VITA

Fangyang Shen was born in Guangdong Province, People's Republic of China, on August 16,1975. After completing high school at Qiandong High School, he attended Guangdong University of Technology and graduated with a Bachelor of Engineering in 2001 and Master of Engineering in 2004. Then he joined Auburn University in August 2004 and obtained his Ph.D. in August 2008.

DISSERTATION ABSTRACT

IMPROVING SYSTEM PERFORMANCE FOR WIRELESS NETWORKS

Fangyang Shen

Doctor of Philosophy, August 9, 2008
(M.S., Guangdong University of Technology, 2004)
(B.S., Guangdong University of Technology, 2001)

76 Typed Pages

Directed by Min-Te Sun

In this dissertation, we solve two problems in wireless networks: Transmission Control Protocol (TCP) enhancement and cluster node sleep scheduling.

When TCP is deployed in a wireless mesh network, it encounters several challenges: transmission errors, packet reordering due to multipath routing, interference among multihop wireless links, and congestion in both wired and wireless networks. Most of the existing TCP enhancements do not properly treat all four challenges. In this dissertation, we propose a new TCP enhancement method, called Congestion Coherence, which can effectively decouple congestion signals from different types of packet recovery. This allows all four aforementioned challenges to be handled properly in wireless mesh networks. Simulation results show that this method improves TCP performance significantly.

In cluster-based sensor networks, part of the sensor nodes can be switched into a sleep state in order to conserve energy, if their neighbors can provide the same or almost the same sensing coverage. However, as the number of nodes in the sleep state increases, coverage for the cluster is degraded. It is crucial to maintain high coverage of clusters in order to preserve performance. In this dissertation, we propose a coverage-aware sleep scheduling (CS) algorithm to improve the

coverage of each cluster. Compared with two previous schemes: the randomized scheduling (RS) scheme and the distance-based scheduling (DS) scheme, the CS algorithm maintains higher coverage, while guaranteeing the same lifetime for the cluster. The CS algorithm thus improves the overall performance of the cluster-based sensor networks.

ACKNOWLEDGMENTS

I would like to take this opportunity to thank many people who guided and helped me during this process.

First, I would like to thank Dr. Min-Te (Peter) Sun for his excellent and patient guidance throughout my graduate student career. I would also like to express my gratitude to Dr. Kai Chang for his help and support, to Dr. Yu Wang for her encouragement, to Dr. Shiwen Mao for being a great resource, and to Dr. Xiao Qin for opening my eyes to the exciting research area of operating systems.

Second, I wish to thank Dr. Tin-Yau Tam who served as my outside reader and Dr. Chunlei Liu who helped me with my first published paper.

Finally, I would like to thank all the other people who helped me through my graduate study at Auburn University.

Style manual or journal used Journal of Approximation Theory (together with the style known as "aums"). Bibliograpy follows van Leunen's *A Handbook for Scholars*.

Computer software used The document preparation package TEX (specifically LATEX) together with the departmental style-file `auphd.sty`.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Wireless networks, especially wireless mesh networks (WMNs) and wireless sensor networks, have attracted a lot of attention in the past few years due to their vast applications. There are many research issues in wireless mesh networks [3, 16], for example, connected dominating set problem, network coding, topology control, and congestion control. Some important issues are addressed in wireless sensor networks [52], like security, scheduling, and routing. Among them, Transmission Control Protocol (TCP) enhancement [4] in wireless mesh networks and the cluster node sleep scheduling problem [5] in wireless sensor networks are of great importance. There are two reasons for this. First, TCP is an important protocol in computer networks, the efficiency of TCP will affect the performance of the networks. With the help of a good TCP protocol, the network can improve the performance significantly. Hence, it is necessary to investigate how to improve the performance of TCP. Second, the network lifetime and coverage problems are of crucial significance in wireless sensor networks. In general, prolonging the network lifetime and increasing the network coverage are two conflicting goals. If we can achieve a good balance between lifetime and coverage of wireless sensor networks, then we can achieve better performance of the network systems with less cost. In the next section, we will introduce the background information of these two problems.

1.1 TCP Enhancement in Wireless Mesh Networks

In recent years, WMNs have emerged as a quick and inexpensive way to build access networks from IEEE 802.11 hardware. While traditional wireless local area networks (WLANs) use only a

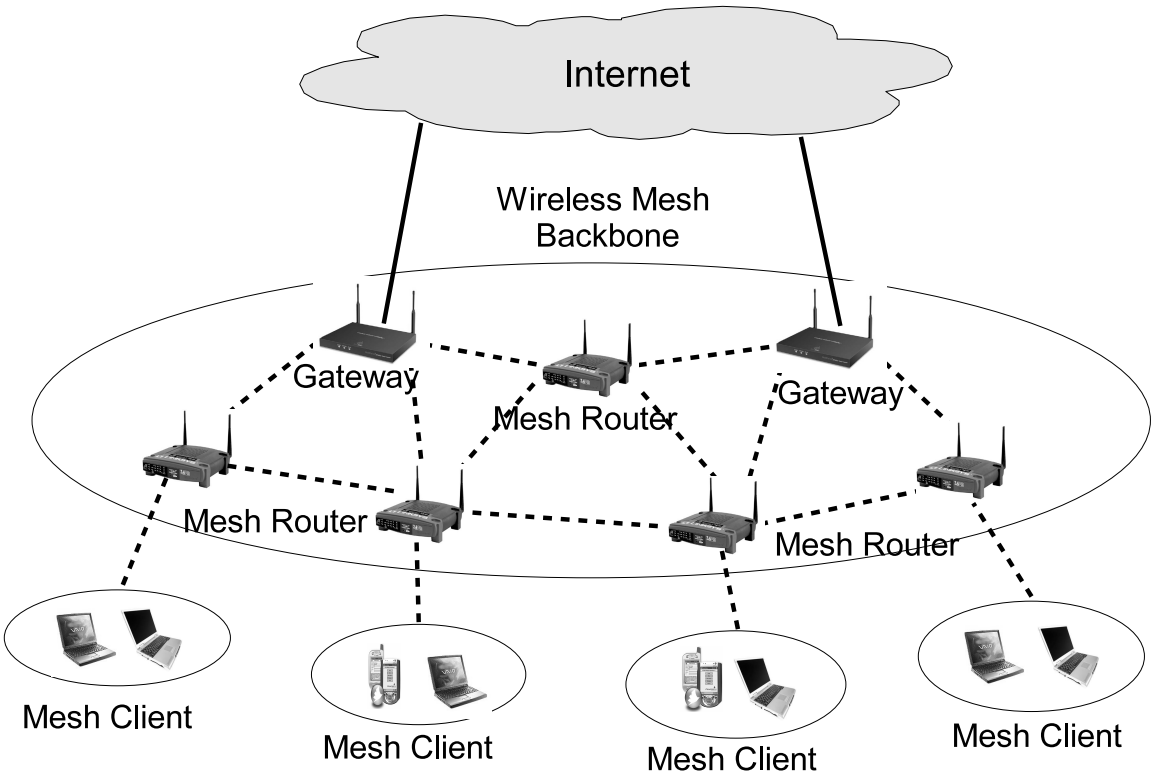

Figure 1.1: An Example of Wireless Mesh Networks

single radio between access points and mobile clients, WMNs exploit multiple radios for transmissions between mesh routers and mesh clients. By coordinating the transmissions on multiple radios and channels, WMNs can avoid interference among neighboring hops and thus provide broadband and expedient connections to the Internet [13, 16, 43]. An example of WMNs [12] is depicted in Figure 1.1 .

To take full advantage of WMNs, many research issues, such as backbone construction, cross-layer design, multi-channel medium access control (MAC), and fault tolerance [16], need to be addressed. In this dissertation, we explore the Transmission Control Protocol (TCP) enhancements for wireless mesh networks.

While the idea of using multiple radios seems plausible, significant effort is needed to coordinate the physical layer, the MAC layer, and the transport layer. TCP is the most widely used transport protocol, but it was designed mainly for wired networks, where transmission errors are rare and the majority of packet losses are caused by congestion. It assumes that all packets of a connection follow the same path from the source to the destination. Under such assumptions, TCP uses packet losses as an indication of network congestion and triggers congestion control mechanisms upon retransmission timeout and three duplicate acknowledgments.

When TCP is deployed on WMNs, the above assumptions are no longer valid. The following four challenges need to be overcome before TCP can be used effectively in WMNs:

- **Transmission Errors:** Unlike a wired media, wireless transmission errors are frequent due to environmental interference and user mobility. Corrupted packets are discarded, and therefore, packet losses can no longer be used as indication of network congestion.

- **Multihop:** WMNs are also multihop wireless networks. A mesh client connects to the Internet by going through several mesh routers that are connected together by wireless links.

This is different from the last-hop access network model, where the only wireless link is the last hop, because transmission on one hop can be interfered by neighboring wireless hops. Hidden terminal and exposed terminal effects dramatically increase the packet loss rate.

- **Multipath:** Since multiple radios are used by the mesh routers, packets are more likely to follow different paths and channels and also reach the destination out-of-order. This could trigger spurious segment retransmissions, keep the congestion window unnecessarily small, break the self-clocking from acknowledgments, and skew the estimated round trip time and retransmission timeout. This will dramatically undermine the foundation of TCP congestion control.

- **Congestion:** In a WMN, wireless media is shared by mesh routers and mesh clients. The available bandwidth of a node is affected by not only the underlying wireless technology but also by the on-going transmissions in the nearby area. Even though the bandwidth of the current wireless technology may be abundant, it is possible to experience congestion inside the mesh network. TCP needs to consider both congestion in wired networks and congestion in WMNs.

Due to these challenges, current TCP algorithms suffer severe performance degradation in WMNs. The performance degradation mainly comes from the following three damages:

- **Spurious end-to-end retransmission** happens when a corrupted packet in wireless transmission is being locally retransmitted, but duplicate acknowledgments trigger the retransmission from the TCP source. It can also happen when packets take different routes and arrive at the destination out of order. The corresponding duplicate acknowledgments are regarded as an

indication of packet loss in the TCP Fast Retransmit algorithm. Spurious end-to-end retransmissions waste precious wireless bandwidth and reduce the network throughput.

- **Unnecessary slowdown** is another side effect of duplicate acknowledgments. When a packet is retransmitted upon three duplicate acknowledgments, TCP assumes congestion has happened in the network and reduces the congestion window by half to slow down the data transmission rate. In the examples of wireless transmission error and out-of-order arrival in multipath routing, this slowdown is unnecessary and could significantly decrease TCP throughput.

- **Timeout** occurs when severe congestion happens in the network and the transmission has to be restarted. There are two common cases of timeout in TCP. First, when a packet is sent but not acknowledged within an extended period of time, TCP uses timeout to restart the transmission of the presumed lost packet [33]. Second, in some versions of TCP, when multiple packets are lost in one window, the sender cannot recover from the losses and a timeout will occur. Timeout damages TCP performance the most because the transmission is at a full stop for an extended period of time, and it takes several round trip times to bring the transmission from a stop back to normal speed. Multihop WMNs are particularly susceptible to timeout because wireless transmission errors coupled with congestion losses increase the chance of multiple losses in one window.

There has been a significant amount of research on enhancing TCP for wireless networks and for different network scenarios. However, none of the previous research solves the previously mentioned four challenges in WMNs. In this dissertation, we present a new approach to address these challenges, based on the idea of replacing the implicit inference of the network congestion status with explicit congestion feedback. On giving priority to local link layer retransmissions, and on

using sequential coherence of neighboring packets to decouple congestion signals from different types of packet recovery, we use the Explicit Congestion Notification (ECN) to deliver the congestion feedback based on using the instantaneous queue length. Simulation results show that our algorithm effectively decouples congestion signals from different types of packet recovery and thus significantly improves the performance of TCP.

1.2 Coverage-aware Sleep Scheduling for Cluster-based Sensor Networks

Wireless sensor networks are network systems composed of small and inexpensive devices, deployed in a region to provide monitoring or communication capabilities for commercial or military applications. Typical applications include asset tracking and habitat monitoring [52], among many others.

The lifetime of a sensor network is measured by the time at which the network can provide a reasonable detection ratio, or before most of the nodes have exhausted their battery power. Due to the vulnerable nature of individual sensors, wireless sensor networks are made up of many sensors deployed at a high density in order to increase the lifetime of the network systems. Since each sensor is battery powered, energy conservation is important to prolong network lifetime. In most of the sensor network systems (e.g., IEEE 802.15.4 ZigBee [53]), each sensor node operates in one of two states: active state and sleep state. In the active state, a node is either actively transmitting or receiving data, or resting in an idle state. In the sleep state, a node does not take part in most activities, therefore it consumes much less energy. In certain scenarios of high density networks with energy constrained sensors, it is possible to selectively turn off some nodes rather than have all nodes active all the time. Sensor nodes can switch to the sleep state if they are in a situation where their neighbors can provide the same or similar sensing coverage. Density control, which controls

the density of active sensors at a desired level for a given area and controls sensor deployment, can ensure that a sufficient number of the sensor nodes remain active to maintain a high coverage level for the area where the sensor network is deployed. It is thus possible to achieve a balance between high coverage and a longer lifetime.

Many sensor systems are formed by a number of clusters [54]. In each cluster, a cluster head is elected to schedule the activities in the cluster, aggregate the sensing data, and communicate with neighboring clusters. In general, the sensing range for each node is smaller than its communication range. In this dissertation, we develop a sleep scheduling algorithm to maximize the coverage of a cluster-based sensor network, while at the same time putting a fixed percentage of the sensor nodes into the sleep state in order to improve the lifetime of the network. Specifically, we propose the coverage-aware sleep scheduling (CS) algorithm. The CS algorithm is based on the consideration of the overlap of sensor nodes' sensing areas. Nodes whose sensing coverage largely overlaps those of their neighbors are assigned a higher probability of being in a sleep state in each cycle. Sensor nodes that have less overlap are assigned a lower probability of being in a sleep state. Simulation results show that the CS algorithm maintains a higher coverage than schemes based on either randomized scheduling (RS) or distance-based scheduling (DS) algorithms [55], while at the same time maintaining almost the same lifetime of the cluster. The CS algorithm thus achieves a much better performance in terms of coverage and lifetime for cluster-based sensor networks. Simulation results show that the CS algorithm is very adaptable and can be applied to sensor networks with heterogeneous nodes (i.e., nodes with different sensing ranges and different initial energy), which was not considered in any previous studies.

The remaining chapters are organized as follows: Chapter 2 presents TCP Enhancement for WMNs; Chapter 3 proposes coverage-aware sleep scheduling for cluster-based sensor networks; Chapter 4 summarizes the dissertation and points out implications for future research.

CHAPTER 2

TCP ENHANCEMENT FOR WIRELESS MESH NETWORKS

2.1 General Issues on Wireless Mesh Networks

WMNs route packets between nodes that are capable of self-organization and maintenance. This provides many alternate paths between the source and the destination and results in quick re-configuration and continuous connections when an existing path fails. As a result, this self-healing ability makes WMNs both robust and reliable.

Mesh networking standard 802.11s [1] is the unapproved Institute of Electrical and Electronics Engineers (IEEE) 802.11 standard for Extended Service Set (ESS) mesh networking. The IEEE 802.11s standard is a collection of access points (APs) interconnected with wireless links that enable automatic topology learning and dynamic path configuration. It specifies an extension to the IEEE 802.11MAC standard to solve the inter-operability problem by defining an architecture and protocol.

The choice of radio technology for WMNs is crucial. In a traditional wireless network, where wireless devices connect to a single access point, each device has to share limited bandwidth. In WMNs, devices with an adaptive radio technology will only connect with other devices within a range. The advantage of WMNs is that more devices within a certain range increases the available bandwidth.

WMNs are generally built on top of home networks, which are typically WLANs. A WLAN provides high data rate connections in a local area to the Internet. Most WLANs operate in unlicensed bands that are free of charge. The IEEE, European Telecommunications Standards Institute (ETSI), and HomeRF Working Group (HomeRF WG) have all been involved in developing standards for WLANs, but the ones that dominate the market are from IEEE. Currently there are four

IEEE specifications for WLAN: 802.11b, 802.11a, 802.11g, and 802.11n. The WLANs that are based on these specifications are the building blocks for wireless mesh networks.

- IEEE 802.11b (also referred to as Wi-Fi) [2, 6, 7, 8] - IEEE 802.11b supports transmission rates of up to 11 Mbps and uses the unlicensed 2.4 GHz Industrial, Scientific, and Medical (ISM) spectrum. There are two immediate consequences of using an unlicensed band. First, the transmissions in a IEEE 802.11b network are prone to interference from other devices utilizing the same spectrum, such as microwave ovens and cordless phones. Second, the transmission power of a IEEE 802.11b device has to conform with certain regulations so that it will not be harmful to other devices using the same unlicensed band. To deal with these issues, spread spectrum is primarily used.

- IEEE 802.11a [2, 6, 7, 9] - IEEE 802.11a supports high transmission rates of up to 54 Mbps by using an orthogonal frequency division multiplexing (OFDM) that operates in the 5 GHz band. Due to the choice of the 5 GHz spectrum, IEEE 802.11a devices lead to far less radio frequency (RF) interference. With high data rates and relatively less interference, IEEE 802.11a is especially well suited to supporting multimedia applications.

- IEEE 802.11g [2, 6, 7, 10] is an attempt to benefit from the positive aspects of earlier standards. It supports bandwidths of up to 54 Mbps and uses the 2.4 GHz ISM spectrum. This is compatible with 802.11b, meaning that 802.11g access points (AP) will also work with 802.11b wireless network adapters and vice versa.

- IEEE 802.11n [2, 6, 7] - IEEE 802.11n builds upon previous 802.11 standards by incorporating multiple-input multiple-output (MIMO) technology. It offers especially high transmission rates of 100 Mbps to 200 Mbps.

To improve the performance of a WMN, especially to improve the per node throughput, the access point is usually assumed to be equipped with multiple wireless interfaces using either the same or different WLAN technologies. One of the important research issues is how to take advantage of the availability of multiple wireless interfaces at each access point to maximize the communication throughput [11].

2.2 Survey of Existing Approaches on TCP Improvement Techniques

2.2.1 TCP Congestion Control Mechanisms

The Transmission Control Protocol is the standard for reliable data transport. It was mainly designed for running on a reliable link layer and assumes most packet losses are due to congestion. TCP achieves reliability by requiring the sender to retransmit the lost packets. Starting with no information about the network's capacity, the sender uses a variable, the congestion window (cwnd), to control the transmission rate. This variable denotes the number of packets that can be sent without acknowledgment. In the *slow start* phase, cwnd doubles in every round trip time. When a lost packet is detected, the sender assumes the network capacity is surpassed, so it reduces the window size to alleviate the congestion and enters the *congestion avoidance* phase.

TCP uses two mechanisms to detect congestion. The first is timeout. When a packet is sent out, the sender starts a retransmission timer. If the packet's acknowledgment is not received before the timer expires, the packet is presumed to be lost due to congestion in the network. The second mechanism is fast retransmit [17]. In this method, the receiver sends a duplicate ACK immediately upon the reception of each out-of-order packet. The sender interprets the reception of three consecutive duplicate ACKs as a congestion packet loss. Fast retransmit can avoid the long wait for the retransmission timer to expire after packet losses.

The two congestion detection methods are based on the assumption that the underlying link layer is reliable. This assumption, while reasonable for wired links, has disastrous results for wireless links. When a packet is lost due to a wireless error, TCP interprets the loss as an indication of congestion and reduces its congestion window. The window reduction causes starvation in the low-bandwidth wireless link. Therefore, standard TCP performs poorly on wireless links. An experiment on the WaveLAN shows that a 1.55% frame error rate causes a 22% reduction in throughput [49].

2.2.2 Wireless TCP Enhancements

Packet losses due to wireless transmission errors have a different impact on TCP than those due to congestion. Packets dropped because of buffer overflow in a congestion episode must be retransmitted from the source and should trigger congestion control actions. On the other hand, wireless packet losses are the results of environment interference, user mobility, or local link quality. In the event of wireless packet loss, the sender of the wireless link usually has a local copy of the lost packets and can retransmit locally. No congestion control actions should be triggered. Many enhancements have been proposed to enhance TCP over wireless links. They either hide, isolate or differentiate wireless transmission errors from congestion losses and then trigger the appropriate congestion control mechanisms.

The local link layer retransmission methods use Forward Error Correction and Automatic Retransmission Request to hide the lossy characteristics of the wireless link and to build a reliable link layer so that upper layers are less affected by wireless transmission errors. Example methods are mentioned in [50, 22, 30, 36, 39, 37]. In reality, retransmission mechanisms in multiple layers may respond to the same loss event and cause undesirable interactions. Although some studies

show that a reliable link layer through retransmissions can achieve good TCP performance, they also pointed out that these enhancements are designed for the characteristics of specific TCP connections and transmission error conditions. When error conditions and connection characteristics change, undesirable interactions and performance degradation may happen.

The split connection methods divide the entire transmission path into two parts i.e., the wired part and the wireless part and run TCP separately on these two parts. I-TCP [18] is a representative of this category. These enhancements violate TCP's end-to-end semantic and may result in unrecoverable data loss. Thus they are a deviation from the original purpose of TCP as a reliable transport protocol.

The source-end TCP enhancements modify TCP code at the source to estimate the available bandwidth, experienced delay or other congestion signals and change the congestion control accordingly. Examples include Wireless TCP [44], TCP Santa Cruz [40], TCP Peach [14], TCP Peach+ [15], TCP Westwood [29], TCP Jersey [47], Multiple Acknowledgments [24], and Control Connection [21]. Their major problem is the location of modifications. Instead of making changes local to the wireless link and host, they require changing computers that the wireless host communicates with. In a typical scenario where a wireless user accesses the Internet, these enhancements would require virtually all the computers on the Internet to change their TCP code.

The wireless-end TCP enhancements modify the behavior of base stations or mobile hosts in cooperation with the TCP algorithm at the source. Examples are the Snoop protocol [19, 20] and the Delayed Duplicate Acknowledgments (DDA) [46]. Since the changes are local to the wireless hosts and links, these enhancements are more practical than the source-end TCP enhancements.

A problem common to many wireless TCP enhancements is assuming that the last-hop access network model is used, where mobile users use wireless link to connect to the wired Internet via a

base station. This assumption is no longer true in wireless mesh networks, so these enhancements cannot be adopted directly.

2.2.3 Multihop Enhancements

Multihop wireless networks present a special problem because many wireless TCP enhancements (such as the Snoop protocol [19]) assume the last-hop model and use this special structure to determine the cause of packet losses and to recover from such losses. Such enhancement cannot apply to wireless mesh networks. Certain wireless TCP enhancements (such as DDA [46]) do not assume the number and location of the wireless links, and therefore can apply to WMNs.

2.2.4 Multipath Enhancements

An assumption of the TCP algorithm is that all packets of a connection follow the same path to travel from the source to the destination. To comply with this assumption, overall throughput is sacrificed. When multiple paths are available, routing packets through multiple paths is a tempting idea to utilize the available bandwidth of these paths. However, because these paths may have different propagation and queuing delays, TCP packets may arrive at the destination out of order. Such out-of-order arrivals can trigger spurious end-to-end retransmission from the fast retransmit mechanism, and hence cause poor performance.

Examples in the literature to enhance TCP in a multipath environment include [51, 37, 23, 38, 26]. These methods can process the ordering information of segments and ACKs received to recover from spurious fast retransmits, postpone the congestion control responses, or adjust the dupthresh value dynamically to avoid spurious fast retransmit [34]. The problem of these methods is that they may not differentiate the causes of out-of-order arrivals in the WMN setting. In the case of

congestion losses, end-to-end retransmission, and congestion control actions may be delayed, thus sacrificing the overall performance.

From our review of related work, we find that existing TCP enhancements can improve TCP performance in certain scenarios, but none of them can address all the four challenges of WMNs.

2.3 Background Information on TCP Enhancement

2.3.1 Three Causes of Out-of-Order Arrivals

In a WMN, packets can be dropped during a congestion episode or in transmission errors. Packets can also travel through different paths. From a TCP receiver's point of view, these anomalies become out-of-order packet arrivals.

When a new packet with an out-of-order TCP sequence number arrives at the destination, the following three events are the most likely causes:

1. **Congestion Loss:** The packet was dropped because of congestion at a router along its transmission path. The packet must be retransmitted from the TCP source.

2. **Wireless Corruption:** The packet itself was corrupted during transmission on a wireless link and arrives at the destination after being retransmitted by the local link layer. In most wireless networks, the wireless links are enhanced by local link layer retransmissions. The sender of a wireless link caches a packet until it is successfully transmitted and acknowledged by the receiver of the link.

3. **Multipath Delay:** The packet itself follows a path with a longer end-to-end delay and took more time than the subsequent packets that traveled through another path.

All three events can cause out-of-order packet arrivals at the destination and create holes in the packet sequence number space, but their consequences are different. A hole caused by a wireless loss will be filled when its retransmission arrives. A packet that traveled through a longer path will arrive after a short delay. However, a hole caused by a congestion loss will not be filled without an end-to-end retransmission from the source. If the receiver knows that the hole is a wireless loss or is caused by the multipath delay, it should wait. Timeout at the source is a way to trigger the end-to-end retransmission, but timeout is usually associated with a prolonged period of idling. A better way to trigger the end-to-end retransmission is through the fast-retransmit. If the receiver knows the hole is a congestion loss, it should send the duplicate acknowledgments right away to trigger the fast retransmit and to reduce the congestion window at the source.

Therefore, the performance of TCP in WMNs can be improved if we can find a scheme to distinguish congestion losses from the wireless loss and multipath delay.

2.3.2 A Dedicated Congestion Indication

As analyzed above, TCP on WMNs faces four challenges: congestion, transmission errors, multipath and multihop wireless links. The performance degradation is caused by not knowing the network congestion status. In the absence of the explicit knowledge, the TCP source guesses packet losses from timeouts and duplicate acknowledgments, and uses them as an indication of network congestion.

In a WMN, using retransmission timeout as an indication of congestion is no longer valid, because the packet that caused the timeout may have experienced a transmission error and is being retransmitted by a local wireless link. Additionally, out-of-order packets arriving at the TCP destination will no longer mean a packet loss. The missing packet may have taken a longer path and will

arrive later than packets that traveled through a shorter one. Therefore, three consecutive duplicate acknowledgments should not trigger an end-to-end fast retransmit at the source and a reduction of the congestion window.

An explicit notification of the congestion status in the network will solve the ambiguity caused by the guesswork using timeout and three consecutive duplicate acknowledgments. The Explicit Congestion Notification (ECN) [31] was an effort to deliver the explicit congestion status of the network to the sender of a TCP connection. ECN uses two bits in the IP header and two bits in the TCP header for ECN capability negotiation and feedback delivery. When its queue length exceeds a threshold, a router marks the packet as *congestion experienced*. At the destination, the Congestion Experience bit is copied to the *ECN-Echo* bit in the TCP acknowledgment and sent back to the sender with the ACK. Upon receiving the *ECN-Echo*, the sender reduces its congestion window to alleviate the congestion.

As a congestion control mechanism, ECN was proved to be more effective than using packet losses to signal the congestion status of the network. It avoids unnecessary packet drops and retransmissions. In RFC 2309 [27], it was recommended to be widely deployed as a router mechanism on the Internet, and was specified in RFC 2481 [41] in 1999 and RFC 3168 [42] in 2001. It is expected to be widely supported soon.

Although some researchers [25] claim that ECN is not useful in a wireless environment, their results are based on average queue length. In our simulations, we find that the instantaneous queue length together with the mark front feature [35] delivers faster congestion feedback to the TCP senders and therefore achieves better performance.

Our proposed method assumes that ECN is enabled in the entire network.

2.3.3 Local Link Layer Retransmission

In our proposed enhancement, local retransmissions are performed in the link layer. All packets transmitted on the wireless link are locally acknowledged before being deleted from the sender's buffer. Packets negatively acknowledged or not acknowledged after a short timer times out will be retransmitted.

Retransmissions of failed packets have a higher priority than new packets. This is important to reduce the delay of the retransmitted packets, and to minimize the chance of triggering end-to-end retransmissions from the source. One way of implementing the higher priority is to use the "insert from front" strategy. When a packet is detected to be lost, the link layer inserts the failed packet into the front of the transmission queue and transmits it when the medium is available. The maximum number of retransmissions for a failed packet is configurable. The link layer can either retransmit persistently or stop after the maximum number of retransmissions is reached.

2.3.4 Congestion Coherence

ECN can deliver the congestion status of the network as long as the data packet and the acknowledgment packet are not dropped. However, when a packet is dropped by a congested router, the ECN congestion signal carried by that packet is also lost. The scheme we use to differentiate congestion losses from wireless corruption and multipath delay is based on the ECN signals in the packets before and after the lost packets. This is derived from the observation that congestion neither happens nor disappears suddenly. Before congestion becomes so severe that a packet has to be dropped, some packets must be marked as "Congestion Experienced" by ECN. Similarly, after a packet is dropped, congestion does not disappear immediately. The queue size falls gradually and some packets are marked. Figure 2.1 depicts a likely queue length change scenario at a congested

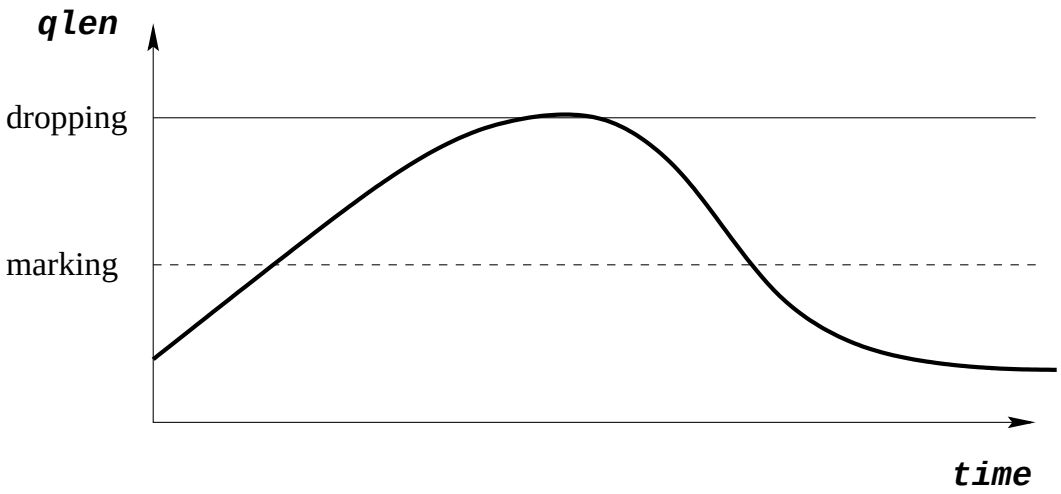

Figure 2.1: Smooth Queue Length Change

router. Between the time that no packet is marked and the time that a packet is dropped, some packets must be marked. An abrupt change as depicted in Figure 2.2 is very unlikely.

As a result, congestion losses are normally preceded and followed by marked packets, see Figure 2.3. We call this phenomenon **congestion coherence** of ECN marking.

The neighborhood of a lost packet is defined as the *coherence context*. There are different ways to define the coherence context, but simulations show that defining the coherence context of packet n as packets $\{n-1, n+1, n+2\}$ yields effective results (Figure 2.14). A packet loss will be considered as a congestion loss if any packet in its coherence context is marked by ECN. In this case, the mesh client responds with duplicate acknowledgments to trigger an end-to-end retransmission and window reduction at the source as specified in RFC 2481 [41].

In contrast to the congestion loss situation, occurrence of transmission errors and multipath delay are usually not related to congestion. If any packet in the coherence context is marked by ECN, congestion control actions are needed to reduce the TCP transmission rate. The chance of having a

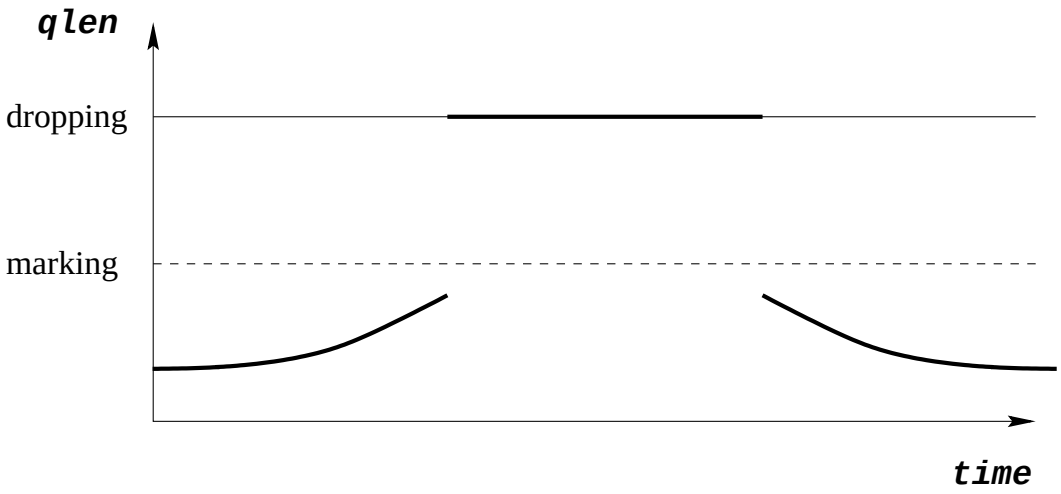

Figure 2.2: Abrupt Queue Length Change Is Unlikely to Happen

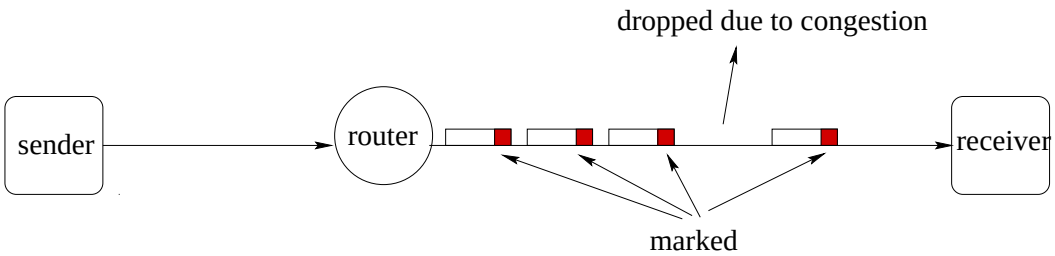

Figure 2.3: Congestion Coherence

congestion loss without a marked packet in the coherence context is very small. So when the coherence context contains no marked packet, the mesh client should hold the duplicate acknowledgments until the retransmitted packet is received.

The above idea is derived in the case where the mesh client is the receiver of the TCP connection. This idea can be applied to the wireless sender case. When the wireless sender receives duplicate acknowledgments, it checks whether the coherence context contains an *ECN-Echo*. If it does, then the duplicate acknowledgments are most likely caused by a congestion loss, so the sender invokes the congestion control. Otherwise, the duplicate acknowledgments are most likely caused by a wireless loss or multipath delay. The sender should ignore duplicate acknowledgments until the local retransmission succeeds.

2.3.5 Compatibility with Legacy TCP

The performance degradation of TCP on WMNs is a problem caused by local network characteristics, such as environmental interference and client mobility. While the mesh clients are eager to enhance TCP, it is unrealistic to expect all or most servers on wired networks to implement the wireless enhancement. Wireless device manufacturers and wireless service providers are likely the implementers of wireless enhancements of TCP. Therefore, the needed changes should be made in the WMNs themselves.

All the needed changes of the proposed method are on the mesh client, so compatibility with legacy TCP is ensured.

Even though this enhancement needs ECN support in all routers in the wired network, it is still a local enhancement because ECN is already a standard that enhances existing wired networks.

The proposed enhancement can apply to encrypted traffic. Intermediate nodes in the path can mark packets as Congestion Experienced through the IP header. They do not need to see the TCP header. The two ECN bits in the TCP header are needed only at the source and destination. Therefore, this method is compatible with encrypted traffic such as that in IPSec.

2.4 Proposed TCP Algorithm

Based on the scheme described above, our proposed TCP enhancement is named *Congestion Coherence* (CC). The algorithms for receiving and for sending are described in Figures 2.4 and 2.5. The modifications to the existing TCP algorithm are made at the wireless end of a TCP connection. No change is needed for the other end of the TCP connection or in intermediate routers. In the case where a mesh client is both a sender and a receiver, the modifications to the receiving and sending algorithms are made at the same end of a TCP connection.

- The TCP sink follows the existing algorithm for sending new acknowledgments, first and second duplicate acknowledgments.
- When the third duplicate acknowledgment is to be sent, the TCP sink checks whether the coherence context is marked. If yes, then the acknowledgment is sent right away. Otherwise, it is deferred for w ms, and a timer is started.
- If the expected packet arrives during the w ms, a new acknowledgment is generated and the timer is cleared.
- If the timer expires, all deferred duplicate acknowledgments are released.

Figure 2.4: Congestion Coherence Receiving Algorithm

- The TCP sender follows the existing algorithm for sending packets and updating the congestion window upon receiving new acknowledgments, the first, and the second duplicate acknowledgments.
- When the third duplicate acknowledgment arrives, the sender checks whether any acknowledgment in the coherence context is an *ECN-Echo*. If yes, then the packet corresponding to the duplicate acknowledgments is sent right away and the congestion window is reduced to half, if a reduction has not been done in the previous RTT. Otherwise, the sender ignores the duplicate acknowledgment and a timer of w ms is started.
- If a new acknowledgment arrives during the w ms, the timer is cleared and new packets are sent as if the duplicate acknowledgments did not happen.
- If the timer expires, the packet corresponding to the duplicate acknowledgments is sent and the congestion window is reduced to half if a reduction has not been done in the previous RTT.

Figure 2.5: Congestion Coherence Sending Algorithm

2.5 Simulation Results

In order to evaluate the performance of the proposed Congestion Coherence(CC) enhancement, we performed a set of simulations with the *ns2* simulator (version 2.30) [45], and collected extensive trace data to analyze CC's performance.

2.5.1 Simulation Model and Settings

Our simulations are performed on the simplified network model as shown in Figure 2.6, where s_1, s_2, s_3 are sources and d_1, d_2, d_3 are TCP destinations. Node r_1 is the gateway of the wireless mesh network. b_1 is a fallback router, r_2 is a mesh router. The wireless links are shown in dashed lines. The numbers beside each link represent its rate and delay. With multipath routing, data and

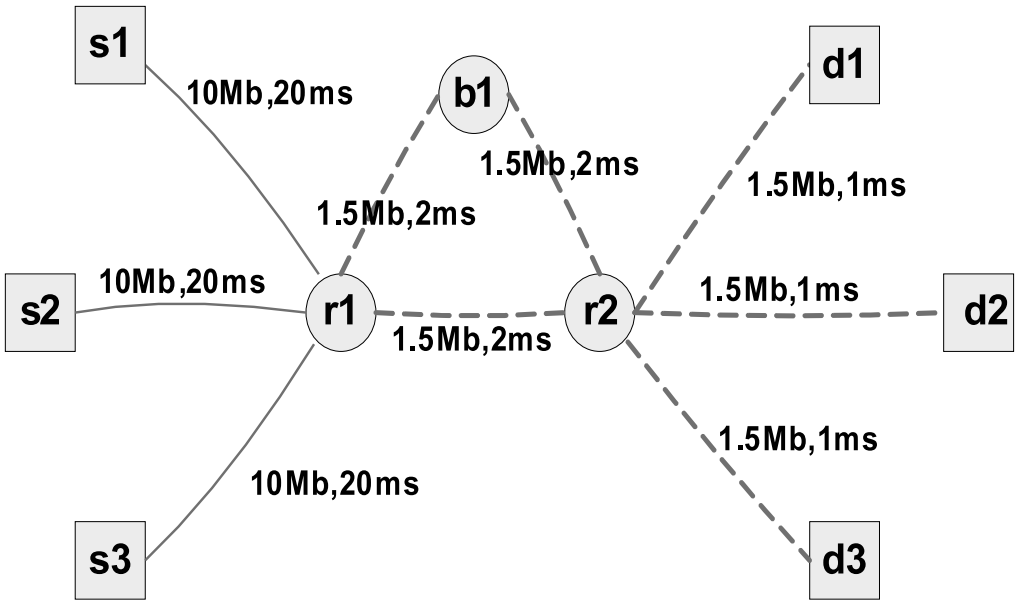

Figure 2.6: A Simulation Model

acknowledgment packets are routed in a round robin fashion at r_1 and r_2, through the direct link and the fallback router b_1.

The experiment traffic is FTP sessions from s_1 to d_1 and from s_2 to d_2 using TCP Reno or Vegas as the transport protocol. A UDP flow from s_3 to d_3 generated from an exponential on-off model is used as the background traffic. The mean burst period and the mean silence period are both 100 ms. The burst data rate is 2,000 kbps. Both TCP and UDP packet sizes are 1,000 bytes, and TCP acknowledgments are 40 bytes long.

Packets transmitted on the wireless link are subject to random transmission errors. The raw packet error rate varies from 0.001 to 0.1. Considering the packet size of 1000 bytes, the bit error rate is roughly 10^{-7} to 10^{-5}. For transmission systems that use Forward Error Correction, this bit error rate should be the residual error rate.

Link layer retransmissions are implemented on the wireless link. Packets sent but not acknowledged at the link layer in 4 ms are resent. Retransmitted packets have a higher priority than new packets, but they are also subject to transmission errors at the same rate. The waiting time w at the receiver is set at 4.2 ms.

To obtain steady state measurements, the length of the simulations is set to 500 seconds.

Our proposed method is compared with TCP Reno, TCP Vegas, TCP Reno with ECN, and DDA. The parameters (α, β) in Vegas are defaulted to $(1, 3)$ as in [28]. Note that some enhancement methods reviewed in the previous section only apply to certain scenarios. For example, Snoop only applies to the last-hop access network model, and the multipath reordering enhancements do not consider wireless networks. Therefore, we have not included them in our comparison.

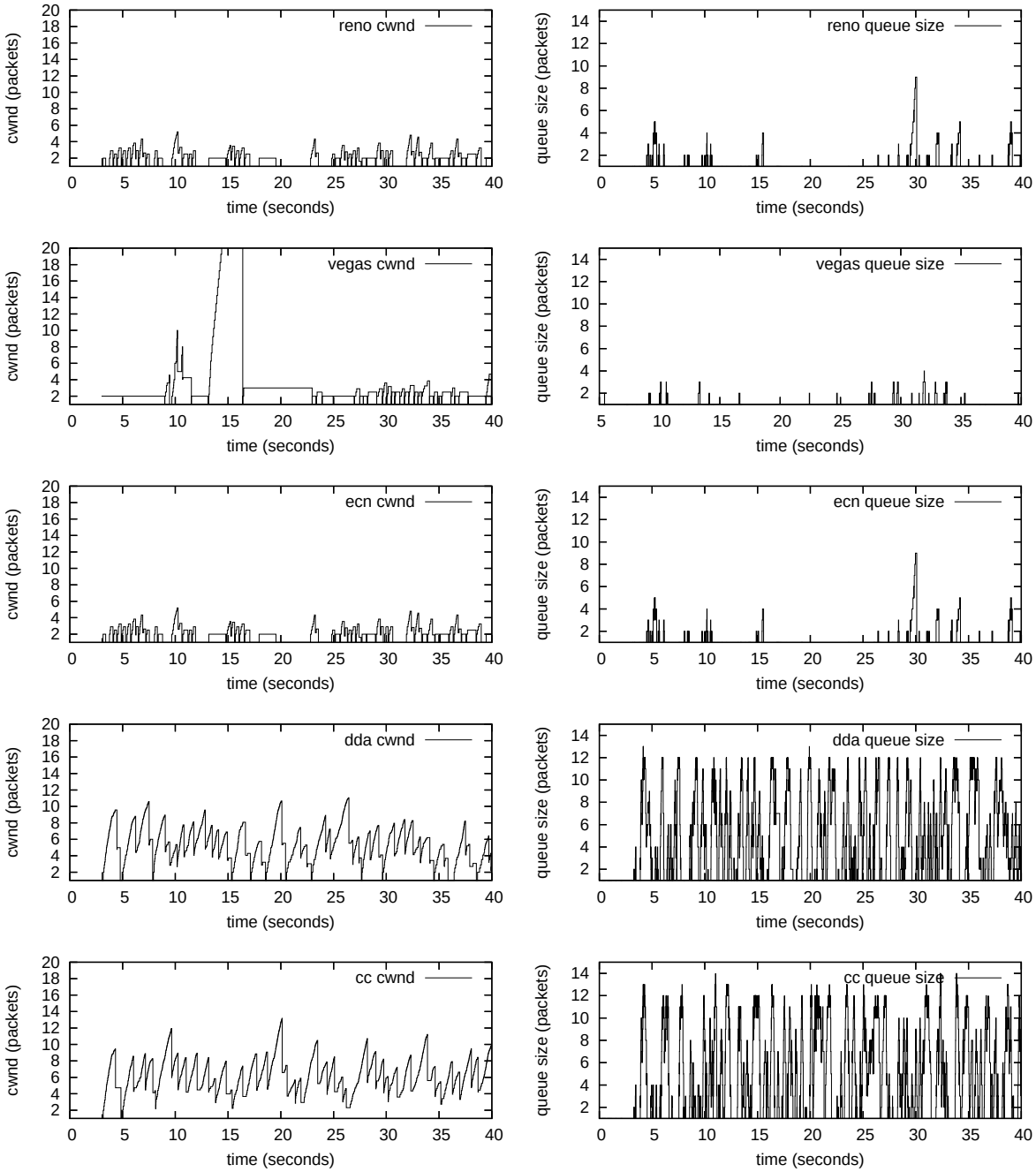

Figure 2.7: Congestion Window and Queue Length for TCP Reno, Vegas, ECN, DDA and CC

2.5.2 Congestion Windows and Queue Length

Our first group of results, shown in Figure 2.7, illustrate the TCP congestion window and queue length of each method. They are collected from 40-second simulation traces. The packet error rate in the simulation is 0.1, corresponding to a bit error rate of roughly 10^{-5}. A calculation shows that the delay and bandwidth that the wireless connection can support is a window size of about 10 packets, but as shown in the figure, the window size of Reno and ECN is reduced frequently. Their corresponding queue size graph shows the queue at the bottleneck link is almost always empty. Therefore, their link efficiency is very low. Vegas's queue sizes are improved. But from the congestion window size, we can see it suffers from long periods of timeout. DDA solves the problem of unnecessary window reductions caused by transmission errors. The window size is significantly increased and the bottleneck link is better utilized. Nevertheless, the spikes in the bottom of DDA cwnd figure indicate that it suffers severe degradation from timeouts. Congestion Coherence is a thorough solution. Unnecessary window reductions and timeouts are avoided. The queue size figure shows that Congestion Coherence has high link efficiency.

2.5.3 Goodput

The major metric to evaluate the methods is *goodput*, which is defined as the number of packets successfully received and acknowledged by the mesh client, excluding the retransmitted ones. The goodput of the five methods under different packet error rates is shown in Figure 2.8. The performance curves confirm that TCP needs enhancements on wireless links. Reno and Vegas perform reasonably well when the packet error rate is very small, but as the packet error rate increases, their performance degrades quickly. ECN is better than Reno and Vegas initially, but deteriorates quickly. DDA does not degrade much with the error rate, but its performance under small error rates is the

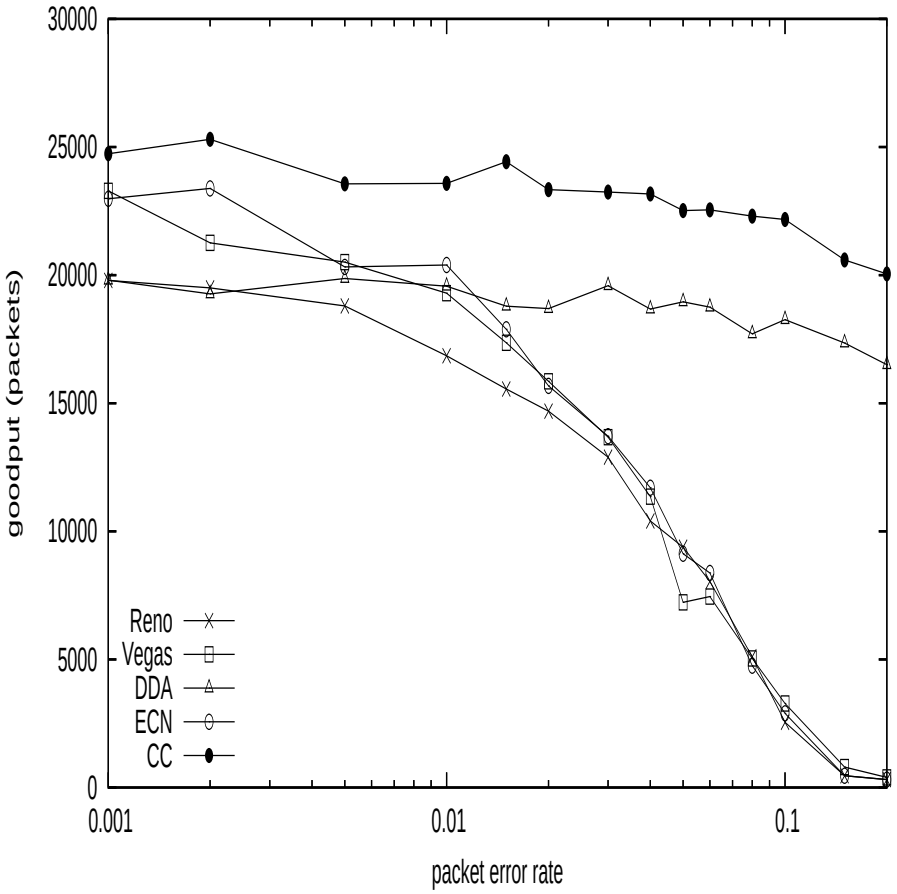

Figure 2.8: Goodput for the Five Methods

lowest. Congestion Coherence performs much better than all other methods regardless of the error rates.

In addition to goodput, we also analyzed the simulation trace and collected other data that helped us understand why some methods work better than others.

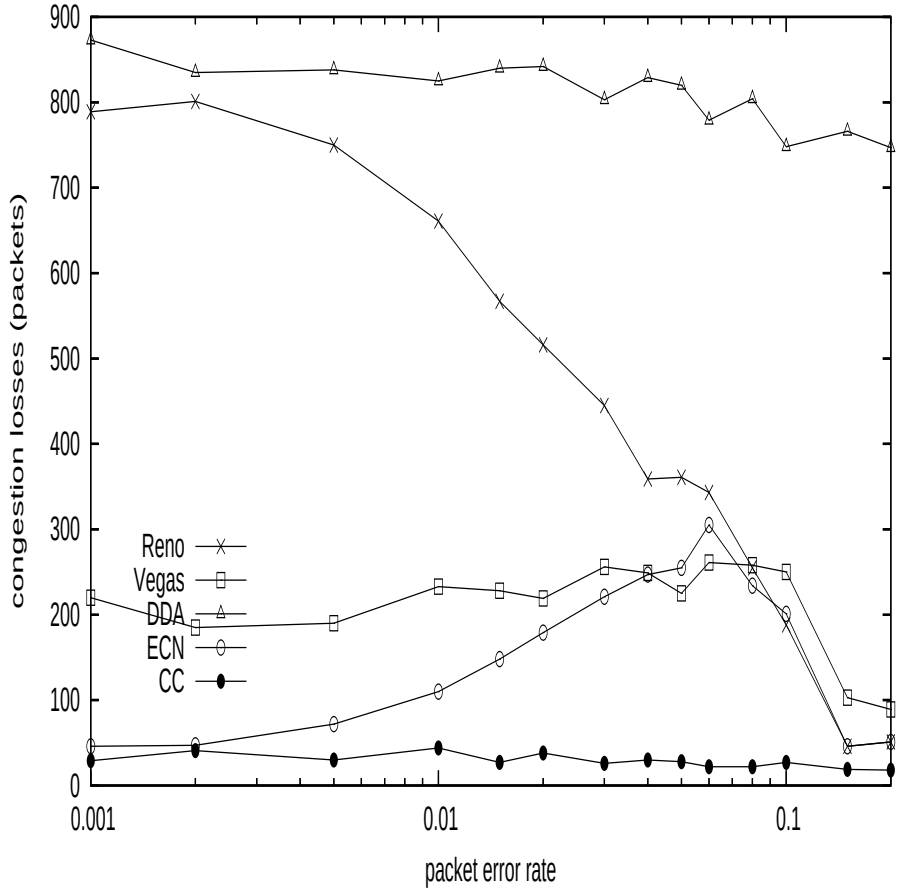

Figure 2.9: Congestion Losses

2.5.4 Congestion Loss

Figure 2.9 show the number of congestion losses. The numbers of congestion losses of Reno, Vegas, and DDA are significantly more than other methods because they use packet losses as a congestion control mechanism. As the packet error rate increases, wireless losses reduce the congestion window so frequently that the window seldom grows to the level that a packet needs to be dropped. This explains the smaller number of congestion losses of Reno in the right half of Figure 2.9. In

contrast, methods using ECN do not suffer from congestion losses on a regular basis. Congestion losses happen only when bursts of background traffic generate so many packets that the buffer of the bottleneck link cannot absorb them. Compared to other methods, the congestion losses of Congestion Coherence are almost unnoticeable. This indicates that Congestion Coherence can quickly and correctly adjust the window size to network congestion. Having fewer congestion losses helps to reduce end-to-end retransmissions and the chance of timeout.

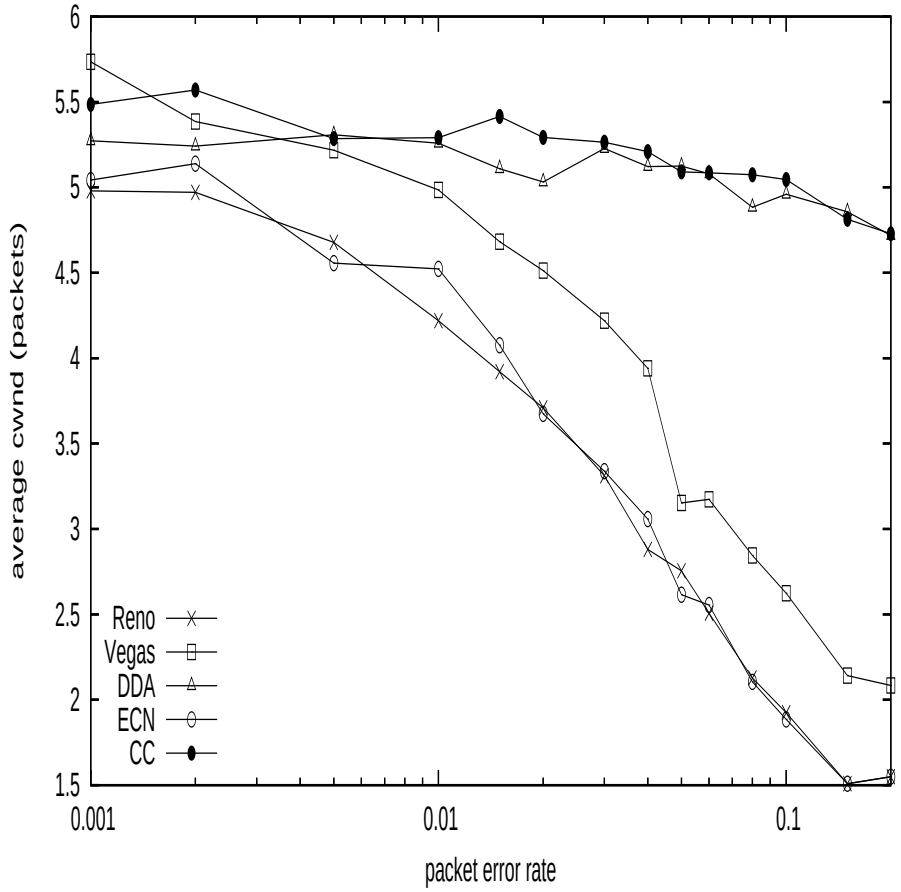

Figure 2.10: Average Congestion Window Size

2.5.5 Average Congestion Window Size

Figure 2.10 shows the average congestion window size. As the packet error rate increases, wireless losses cause unnecessary window reductions in Reno, Vegas and ECN, but the window size of DDA and Congestion Coherence is not affected very much by transmission errors. The slight drop of window size of DDA and Congestion Coherence in the right upper corner is caused by transmission errors in the retransmitted packets. This figure confirms that DDA and Congestion Coherence solve the problem of unnecessary window reductions.

2.5.6 Number of Timeouts

Figure 2.11 shows the number of timeouts that occurred during the simulation period. When the packet error rate is small, Reno, Vegas and DDA have the largest number of timeouts because they use packet losses for congestion control. Their buffer occupancy at the bottleneck link can grow so high that bursts in background traffic can cause continual losses, since two or more losses in a window will cause a timeout. This flaw translates into a large number of timeouts. At the low packet error rate, Vegas, ECN, and Congestion Coherence have very few timeouts because most of their congestion losses are avoided. Background traffic causes occasional losses, but it seldom becomes multiple losses in one window. As the packet error rate increases, the number of timeouts of Reno, Vegas and ECN increases dramatically because a larger number of wireless losses increases the chance of multiple losses in one window. As the congestion window of Reno is reduced frequently by wireless losses (Figure 2.10), and congestion losses become fewer (Figure 2.9), Reno behaves almost identical to ECN. The timeouts of DDA are mainly caused by congestion losses. The numbers are roughly the same for all packet error rates. Congestion Coherence has the smallest

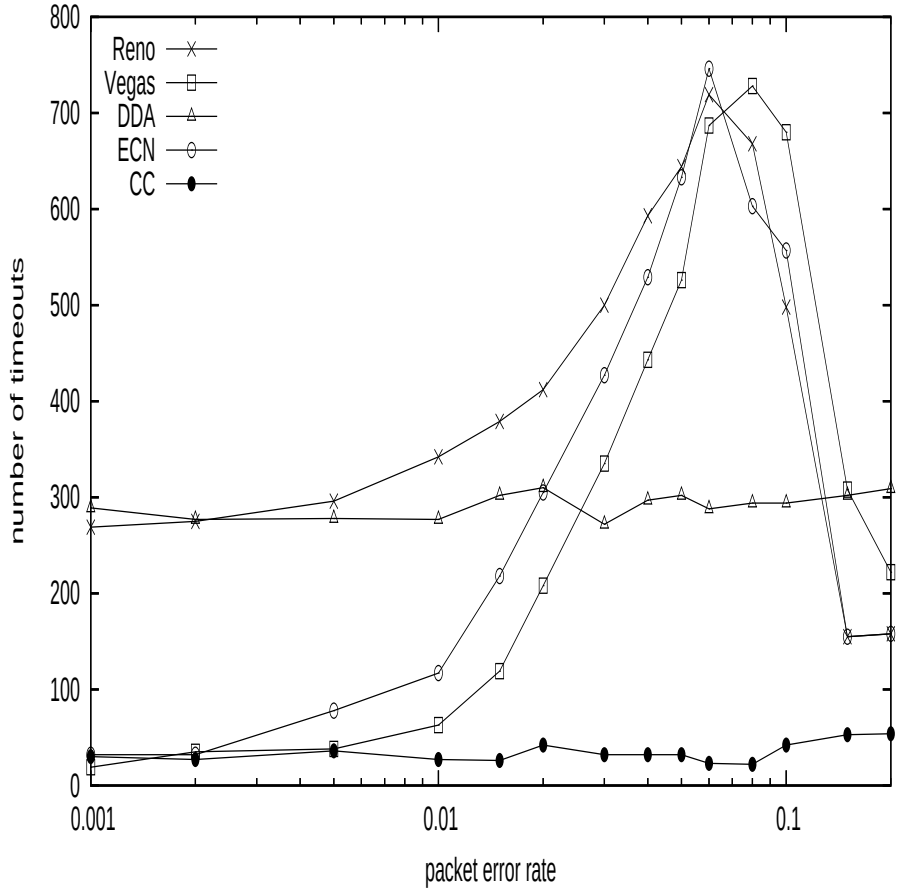

Figure 2.11: Number of Timeouts

number of timeouts of all methods. This figure proves that only our Congestion Coherence avoids the degradation caused by timeouts.

2.5.7 End-to-End Retransmission

Figure 2.12 shows the number of end-to-end retransmissions. This number depends on the number of congestion losses, wireless losses and timeouts, as well as the enhancement method used. In general, congestion losses in all methods are retransmitted end-to-end. Wireless losses in Reno,

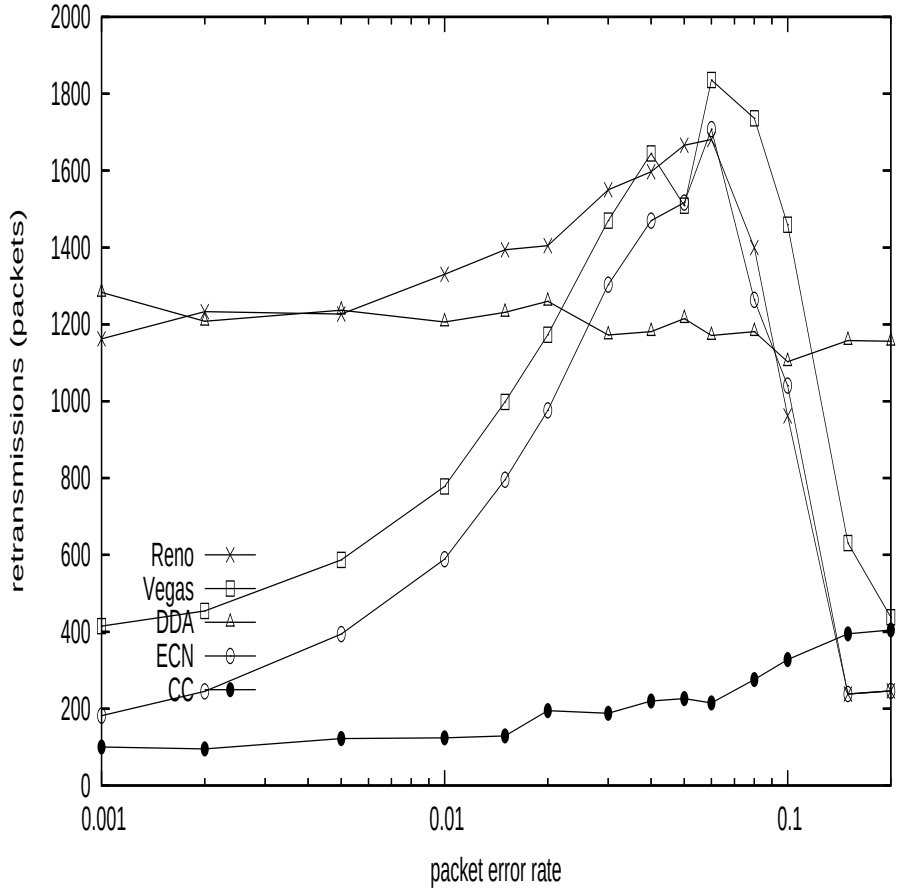

Figure 2.12: End-to-end Retransmissions

Vegas, and ECN are retransmitted end-to-end as well. When a timeout happens, a full window of packets is retransmitted. DDA avoids the majority of end-to-end retransmissions of wireless losses, but still has a large number of retransmissions because of congestion losses and timeouts. ECN reduces congestion losses, but it cannot recover from wireless losses. Congestion Coherence avoids the majority of congestion losses, and waits for the local retransmission for wireless losses, so it has the smallest number of end-to-end retransmissions.

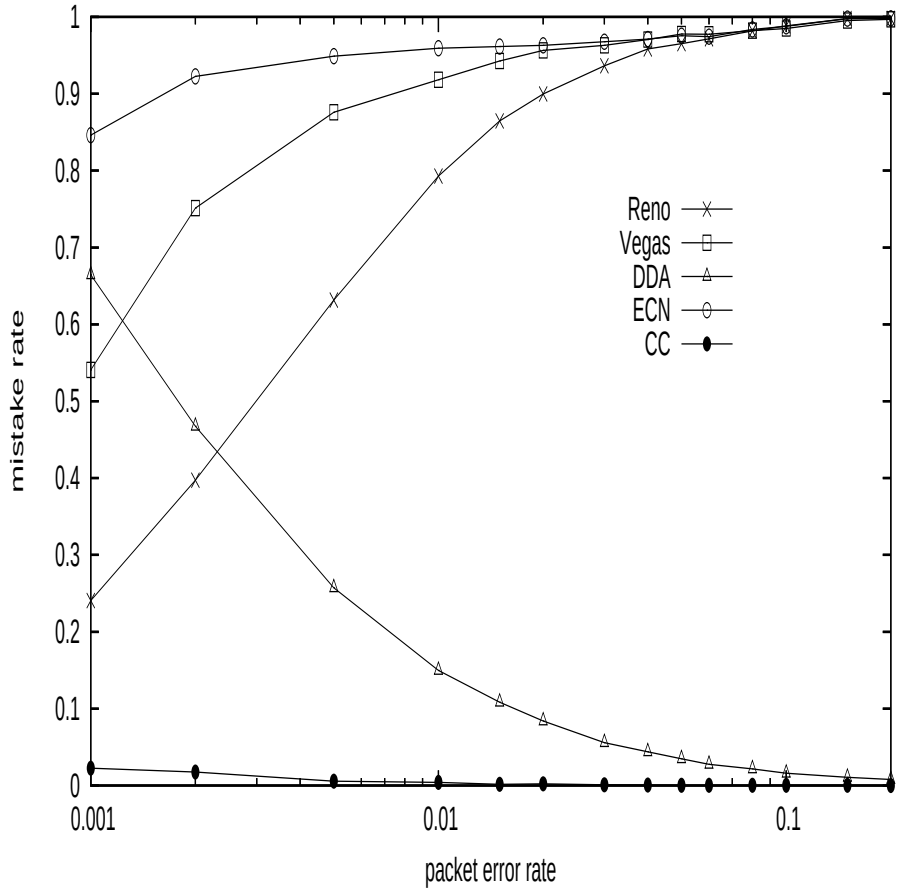

Figure 2.13: Mistake Rate

2.5.8 Mistake Rate

The mistake rate in determining the cause of packet losses is shown in Figure 2.13. Reno, Vegas, and ECN assume all losses are caused by congestion, so their mistake rate is the percentage of wireless losses of all losses. ECN makes almost the same number of mistakes as Vegas and Reno, but it has a much higher mistake rate because of its small number of congestion losses. DDA assumes all packet losses are due to transmission errors, so its mistake rate decreases when

the packet error rate increases. Congestion Coherence takes advantage of congestion coherence and makes the right guess in most cases, but it makes mistakes when highly burst traffic causes sudden packet losses without having neighboring packets marked. In our simulations, Congestion Coherence's mistake rate is almost 0, and it has a minimal impact on performance.

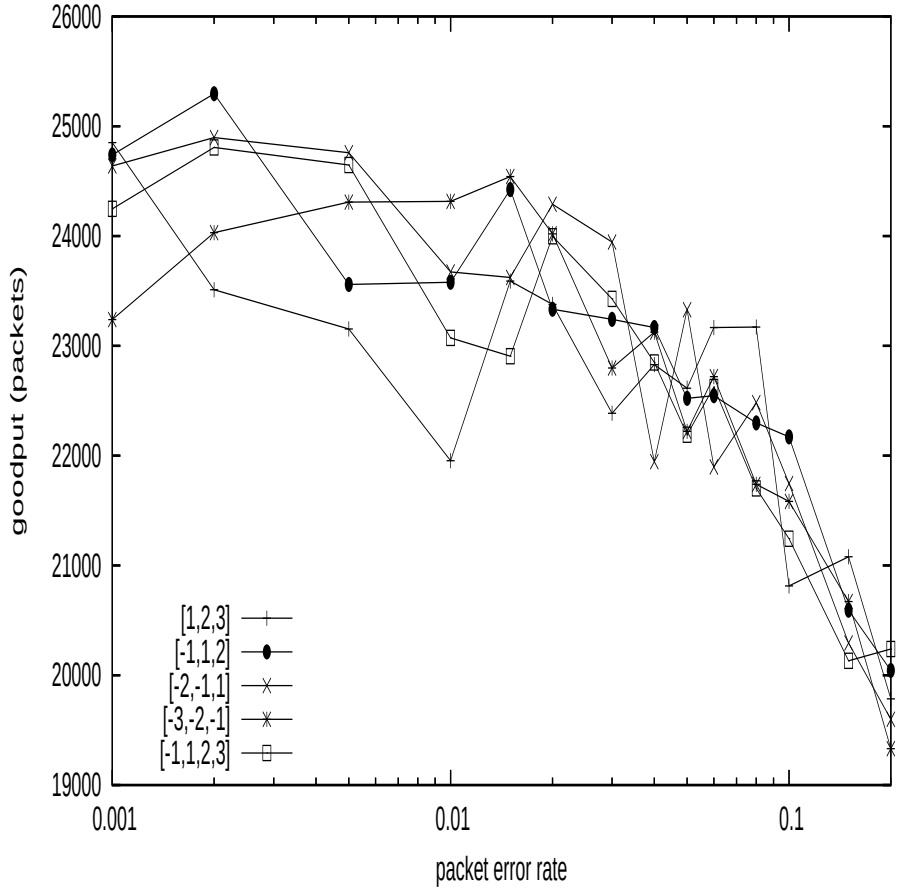

Figure 2.14: Performance of Different Coherence Contexts

2.5.9 Performance of Different Coherence Contexts

Finally in Figure 2.14, we compare the performance of different coherence contexts. Assuming n is the lost packet, the coherence context [-1, 1, 2] consists of packets $\{n-1, n+1, n+2\}$. We can see most of the coherence contexts works well. [-1, 1, 2] yields effective results and has been used in our simulations.

In summary, the simulation results show that Congestion Coherence avoids the majority of congestion losses and is able to distinguish wireless losses from congestion losses. It is the only enhancement that avoids the three degradations of TCP performance over wireless links — end-to-end retransmissions, unnecessary window reductions, and timeouts. Therefore, the Congestion Coherence enhanced TCP outperforms the other existing enhancement methods significantly.

2.6 Summary

In this chapter, we discussed the challenges to TCP in WMNs and their damage to TCP's performance. Built on top of explicit congestion notification, the author proposed a new TCP enhancement for WMNs, namely Congestion Coherence. The proposed enhancement explored the coherence of ECN markings in neighboring packets to decouple the end-to-end congestion signals from the local operations of wireless retransmissions and multipath routing. This allowed different types of packet losses to be properly addressed by the TCP mechanism. Simulations showed that Congestion Coherence dramatically reduced the damage to TCP performance, and therefore, provided a unified solution to the challenges of TCP in WMNs.

CHAPTER 3

COVERAGE-AWARE SLEEP SCHEDULING FOR CLUSTER-BASED SENSOR NETWORKS

3.1 Survey on Scheduling for Cluster-based Sensor Networks

The lifetime of a network is the time span from its initial deployment to the instant when it is deemed nonfunctional [56]. It can be defined as the instant when the first sensor dies, or a certain percentage of sensors die, or if there is an overall loss of coverage. There has been considerable research on network lifetimes of wireless sensor networks [56, 57, 58]. Chen and Zhao [56] proposed a general formula for the lifetime of wireless sensor networks by identifying two key parameters at the physical layer that affected the network lifetime: namely the channel state and the residual energy of sensors. Rai and Mahapatra [57] designed a mathematical model to describe the lifetime of a sensor network when the data generation events within the network were spatially and temporally independent. Based on that model, they introduced an efficient routing strategy to achieve the optimal lifetime. Ritter and Voigt [58] presented a method for experimental lifetime measurement of sensor networks that made it possible to validate lifetime models within a reasonable amount of time. All of these models and methods can accurately measure the lifetime of wireless sensor networks, but they are inadequate to quantify the coverage of wireless sensor networks.

The coverage problem in wireless sensor networks has been the subject of increasing attention in recent years [59, 60]. Meguerdichian et al. [59] proposed an optimal polynomial time algorithm that used graph theory and computational geometry for solving the best and worst case coverage. Ahmed et al. [60] proposed a distributed probabilistic coverage algorithm to evaluate the degree of confidence in detection probability provided by a randomly deployed sensor network, using a uniform circular disc for sensing coverage in the binary detection model. Although most of these

methods attempt to address and solve the coverage problems in different scenarios, few consider the impact on network lifetime. Furthermore, the above approaches can not be applied to the scenarios with density control in wireless sensor networks.

There have been some studies that focus on large scale wireless sensor networks [61, 62, 63]. Zhang and Hou [61] investigated the relationship between coverage and connectivity in large sensor networks. Liu and Towsley [62] studied the issues affecting the coverage and detectability of a 2-dimensional finite-width strip sensor network probability algorithm, which was used to find a path between two random locations without being detected. Then the paper went on to characterize the asymptotic behaviors of the coverage and detectability of large scale sensor networks. Ye et al. [63] studied the issue of sink mobility in large-scale sensor networks and proposed a two-tier data dissemination approach that provided scalable and efficient data delivery to multiple mobile sinks. These studies integrate large wireless sensor networks, but none of the techniques address the combination of coverage and lifetime problems in wireless sensor networks.

Deng et al. [55] studied the node sleep scheduling problem in the context of cluster-based sensor networks. In this paper, the traditional sleep scheduling scheme, randomized sleep scheduling(RS) scheme, is described and used to schedule the node sleeping problem. In the RS scheme, all the nodes are assigned with a random probability to sleep in each cycle. Then the author proposed the linear distance based scheduling (DS) scheme to put the sensor nodes to sleep in each cycle. The DS scheme selects sensor nodes to sleep with higher probability, if they are farther away from the cluster head. In contrast, the overall performance of the DS scheme is better than that of RS. Although DS offers a longer lifetime than RS, its coverage is not high. In general, high coverage is vital for wireless sensor clusters. Even though the lifetime is also significant, coverage of a cluster cannot be sacrificed solely in order to achieve a longer lifetime.

Table 3.1: Terms

R: the maximum transmission range of a cluster head;
P: the set of nodes in the cluster;
β_s: the percentage of nodes to sleep in each cycle;
β_d: the percentage of sensor nodes that run out of energy;
T(β_d): the time when β_d percent of sensors run out of energy;
p(k): the probability of a sensor node k to be put to sleep;
N: the number of test points in the cluster;
D(i, j): the distance between two points i and j;
r: the transmission range of each sensor node;
$I_k(i)$**:** the indicator function between an active node k and a point i;
O(i): the overlap of all nodes at point i;
Z(k): the coverage degree of node k;
C(k): the summation of the overlap degree of all points in Z(k);
θ**:** the parameter to adjust the value of p(k) of node k;
$E_{active}(x)$**:** the average energy consumption per second of each active node;
λ**:** the average packet transmission rate per second for each sensor node sending data to cluster head;
k_1**:** the constant on energy consumption due to transmission of each packet;
k_2**:** the idel/receive energy consumption per second;
x_{min}**:** the minimum transmission range for the minimum allowable transmission energy;
γ**:** the path loss exponent;

3.2 Coverage-aware Sleep Scheduling Algorithm

This section introduces the CS algorithm. The related terms are defined in Table 3.1. The CS algorithm is designed to solve the following problem: how can a cluster head schedule nodes in the cluster to sleep, so that the cluster can still achieve high coverage and maintain the longest possible lifetime? The problem is formulated by utilizing the following scenario. In a static cluster, the maximum transmission range of the cluster head is **R** and the set of sensor nodes in the cluster is **P**. Let β_s represent the percentage of sensor nodes in the sleep state in each cycle. The lifetime of a cluster T(β_d) is defined as the time when β_d percentage of sensor nodes run out of energy. In

order to maximize the sensing coverage and maintain a reasonable lifetime for the whole cluster, how does the cluster head select exactly β_s percentage of sensor nodes to sleep in each cycle?

The underlying concept governing the CS algorithm is that the greater the overlap of a sensor node k's sensing coverage with its neighbors, the higher the probability $p(k)$ that a sensor node can be placed in sleep state without affecting the overall network performance. In each cycle, the probability that each node should be in a sleep state can be calculated based on its overlap with its neighbors. Then β_d percentage of nodes is set to the sleep state.

Here, we apply the CS algorithm with the following assumptions. First, the cluster maintains a static structure. Each sensor node will stay in the same cluster for the duration of its lifetime. Second, the cluster head is located in the center of the cluster. Third, a cluster head knows the location and sensing range of each node in its cluster. Last, a two-dimensional Poisson Process is used to control the distribution of the nodes in a cluster.

The cluster spans a circle area of πR^2 and each sensor node covers a small area of πr^2 if their sensing range is r. To estimate the coverage of the cluster, we use N test points in the cluster circle area. In general, the higher density of the points in a cluster, the more accuracy to estimate the coverage of a cluster. If the distance of two points i and j is denoted as $D(i, j)$, the indicator function $I_k(i)$ represents whether an active node k covers a point i, which is defined in Equation 3.1.

$$I_k(i) = \begin{cases} 1, & \text{if } D(i,k) \leq r \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Then the overlap degree of a test point i, $O(i)$, is the number of sensor nodes that cover point i, which can be obtained from Equation 3.2.

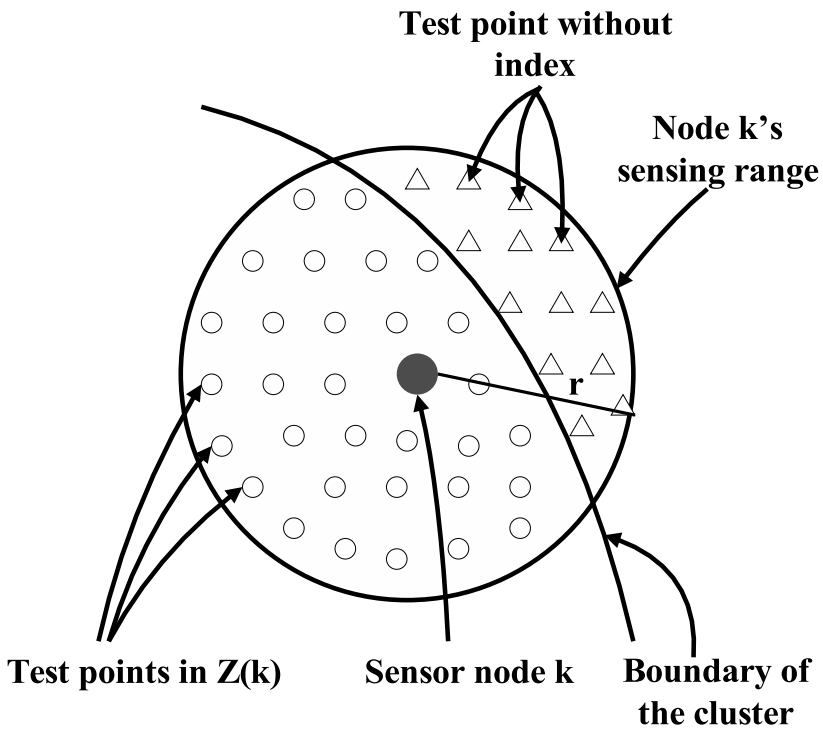

Figure 3.1: Test Points in The Sensing Range of Node k

$$O(i) = \sum_{k \in \mathbf{P}} I_k(i) \tag{3.2}$$

Note that some test points may be outside of a cluster (i.e, away from the cluster head for more than the maximum transmission range of the cluster head), but are covered by sensor nodes in the cluster. Therefore, before we estimate the degree of overlap for a sensing range by these test points, we need to accurately estimate the actual points within the sensing range of each sensor which are actually inside the cluster.

A cluster head indexes each test point inside its cluster with a unique integer ranging from 1 to N. The sensing range of sensor node k covers a number of test points. Let us denote those points

situated inside the cluster (i.e., indexed by the cluster head of node k) as $Z(k)$; the coverage degree of node k, $C(k)$, is defined as the summation of the overlap degree of all points in $Z(k)$. Figure 3.1 illustrates the test points in $Z(k)$ by small circles, which have been indexed by the cluster head of node k.

For each sensor node k inside a cluster, there exists a corresponding C(k). For all sensor nodes in a cluster with cluster head h, let min(C(k)) be $\min_{D(k,h)\leq R} C(k)$. And max(C(k)) be $\max_{D(k,h)\leq R} C(k)$. The probability of node k to be in the sleep state, $p(k)$, can be computed from C(k) using Equation 3.3.

$$p(k) = \frac{\theta \cdot C(k)}{MAX(C(k)) - MIN(C(k))} \quad 1 \leq k \leq P; \tag{3.3}$$

In Equation 3.3, the value of θ is a constant, which will be set appropriately so that the number of nodes in sleep reaches the expected level of β_s percent. In other words, θ is set so that Equation 3.4 is satisfied.

$$\sum_{k \in \mathbf{P}} p(k) = \beta_s \cdot \|\mathbf{P}\| \tag{3.4}$$

As we mentioned before, the lifetime of a cluster is defined as T(β_d), the time when β_d portion of sensors runs out of energy. To calculate the lifetime, we need to know the average energy consumption of each sensor node. According to [55], assuming the cluster head is h, the average energy consumption per second of an active node x for sensing and data communications can be obtained from equation 3.5.

$$E_{active}(x) = \lambda * k_1 * [max(x_{min}, D(x,h))]^{\gamma} + k_2 \tag{3.5}$$

It is important to note that this approach does not guarantee that a sensor node will be assigned to the sleep state even if its sensing range is fully covered by its neighbors. However, the CS algorithm addresses the following two key issues: First, the more often a sensor's sensing range is covered by its neighbors, the higher probability the sensor will be assigned to sleep. Second, the greater the percentage of a sensor's sensing range that is covered by its neighbors, the higher probability that that sensor will be assigned to sleep. This strategy is not directly related to the distance between a sensor node and its cluster head. A major advantage of this approach is that it can be applied to sensor nodes with different sensing ranges, which is not addressed by either the RS or the DS scheme. The CS algorithm also maintains a higher level of coverage than either of the other two approaches.

The time complexity of the RS scheme is $O(1)$ and that of DS is $(\|\mathbf{P}\|)$. On the other hand, the time complexity of the CS algorithm is in proportion to the product of N and $\|\mathbf{P}\|$. To achieve better coverage estimation accuracy, the value of N is normally set to be large, so the run time of the CS algorithm is longer than that of RS and DS. However, there is no need for a cluster head to coordinate with other clusters when conducting the sleep scheduling in its cluster using the CS algorithm as all the information needed for the computation can be obtained locally.

3.3 Simulation Results

3.3.1 Simulation Settings

To determine the impact of the CS algorithm, we simulated the coverage and lifetime for cluster-based sensor networks of the CS algorithm using Matlab and compared the results with those obtained by the RS and DS algorithms. To assess the coverage of a cluster, the three algorithms were tested under multiple situations, including assigning different percentages of nodes to the sleep state,

maintaining nodes with different sensing ranges, and changing the number of nodes in the cluster. Simulation results in terms of lifetimes were shown for different situations, including the use of sensor nodes with different initial energies and sensor nodes with different sensing ranges. The default values used are as follows: $R = 100$ meters, minimum transmission range for each sensor = 5 meters, $\gamma = 2$, average total number of nodes is set to be 500, $k_1 = 0.000001J$, $k_2 = 0.1$ J/sec, $\lambda = 100$ frame/sec. For DS scheme, α is set to be 1.0.

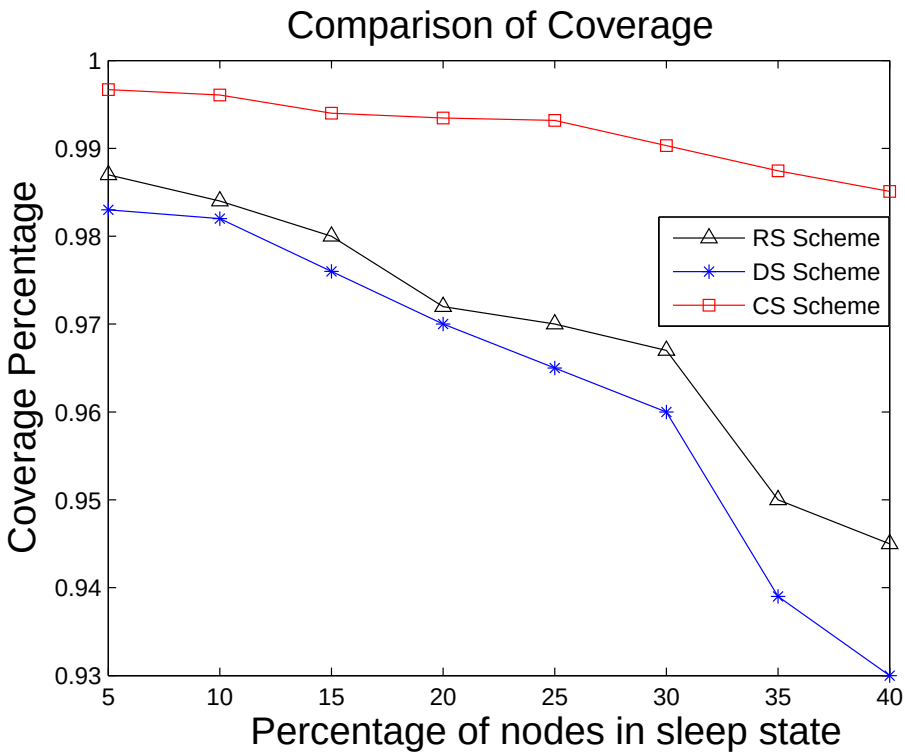

Figure 3.2: Coverage for RS, DS, and CS Algorithms

3.3.2 Impact of β_s on Coverage

The relationship between the percentage of nodes in a cluster assigned to sleep β_s and the coverage of the cluster for the different algorithms are shown in Figure 3.2. The results plotted in Figure 3.2 are obtained by using the default values. The percentage of β_s is set as 5, 10, 15, 20, 25, 30, 35, and 40. It is evident that the coverage of a cluster achieved by all three algorithms degrades as the percentage of sensor nodes in the sleep state increases, which occurs because the sensing ranges of some sleeping nodes are not adequately covered by their neighbors. Once these nodes go into sleep, their sensing range will not be covered anymore, which reduces the total coverage for the cluster. Figure 3.2 reveals that the CS algorithm not only achieves the best performance in terms of coverage, but also has a lower tendency of decreased coverage with increasing numbers of nodes in sleep state. This is because the sensing area of the CS algorithm takes into account overlapping coverage with neighbors, and so performance suffers less in terms of coverage. Based on the evidence presented in this figure, we can conclude that CS algorithm is more effective at achieving good coverage for the cluster than either of the other two algorithms.

3.3.3 Impact of Number of Nodes on Coverage

Figure 3.3 and 3.4 show the correlation between the number of nodes and the coverage of a cluster. The simulation settings of these two figures again use the default values, except that Figure 3.3 uses $\beta_s = 0.10$ and Figure 3.4 sets $\beta_s = 0.30$. The values of R are set as 300, 350, 400, and 450. From these figures, we can see that the coverage of the cluster gradually increases with an increase in the number of nodes for all three algorithms. One important observation is that the coverage achieved by the DS algorithm decreases sharply as the number of nodes in the cluster decreases. This is because the DS scheme shuts down nodes which are far away from the cluster

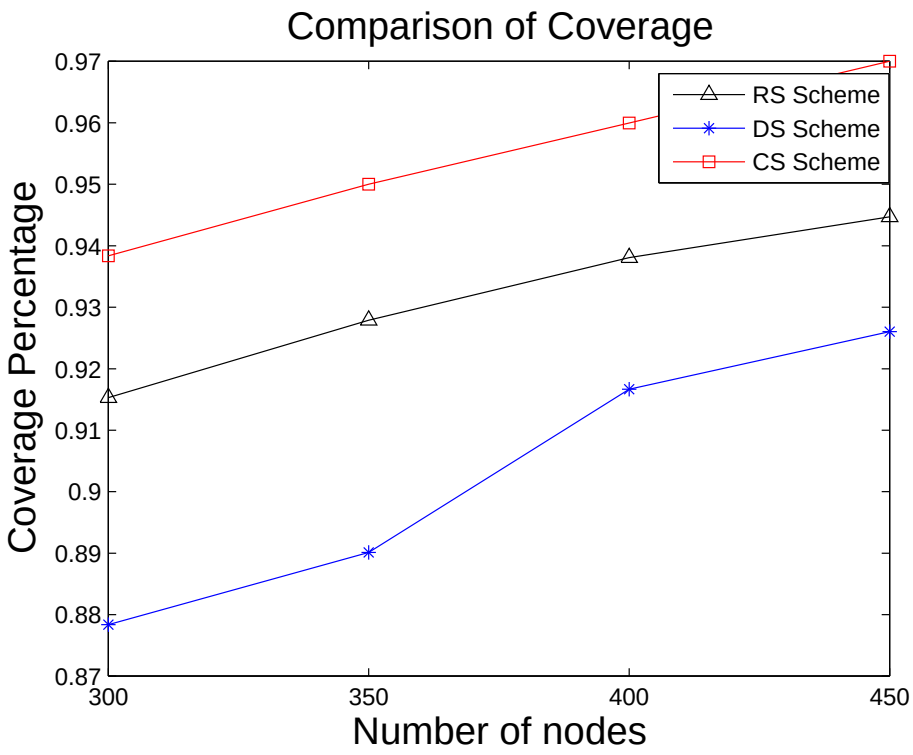

Figure 3.3: Coverage for RS, DS, and CS Algorithms, $\beta_s = 0.10$

head with a higher probability, and the sensing ranges for these nodes are less likely to be covered by their neighbors. The RS scheme performs better because it selects nodes to put into the sleep state with a random probability between 0 and 1, which results in a uniform distribution of sleeping nodes in the cluster. On the whole, the CS algorithm achieves the best performance because the CS algorithm puts those nodes with the highest overlap with their neighbors to sleep. Hence, even as the number of nodes steadily decreases, the coverage achieved does not degrade quickly, as the sensing areas of the sleeping nodes are covered by their active neighbors. This also means that the CS algorithm adapts better to changing numbers of nodes in a cluster and can achieve better coverage using fewer nodes than either of the other two algorithms.

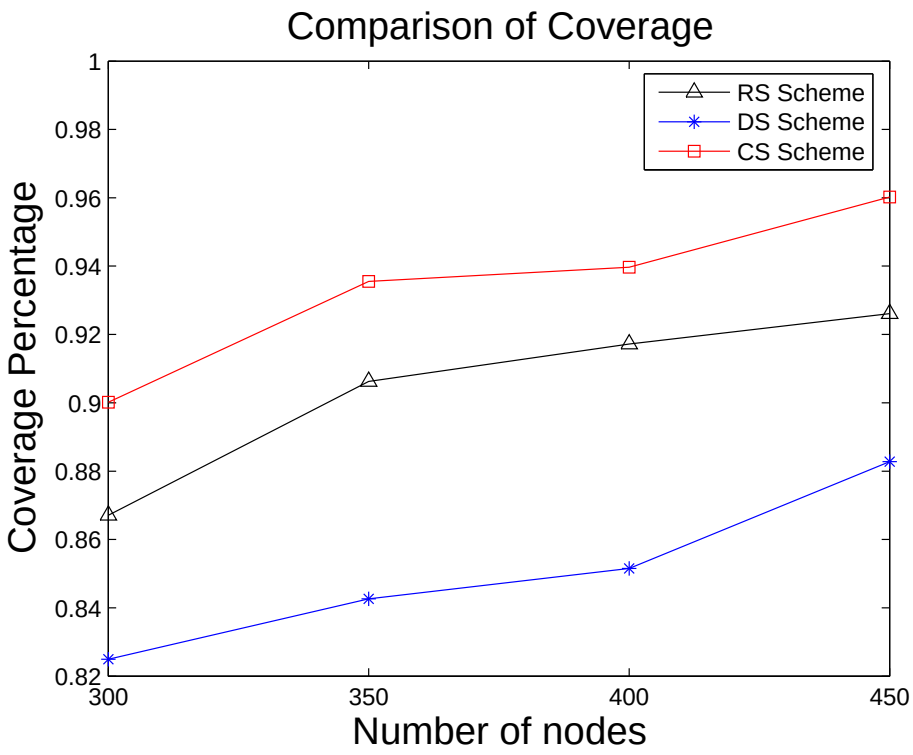

Figure 3.4: Coverage for RS, DS, and CS Algorithms, $\beta_s = 0.30$

3.3.4 Impact of Sensing Range on Coverage

In order to evaluate the performance of different protocols for different sensing ranges, simulations were conducted using different sensing ranges for the sensor nodes, ranging from 6 to 14 meters. In each of the experiments, all the sensors had the same sensing range, for example, 6 meters or 8 meters. Other settings used were the default values. The results are presented in Figures 3.5 and 3.6. Figure 3.5 shows the coverage of the three algorithms for $\beta_s = 0.10$ and Figure 3.6 shows the coverage of the three algorithms for $\beta_s = 0.30$. Figure 3.5 reveals that the performance of all three algorithms is poor when the sensing ranges for all the sensors are below 10 meters. This is because shutting down some portions of the sensor nodes will reduce the coverage when the sensing range

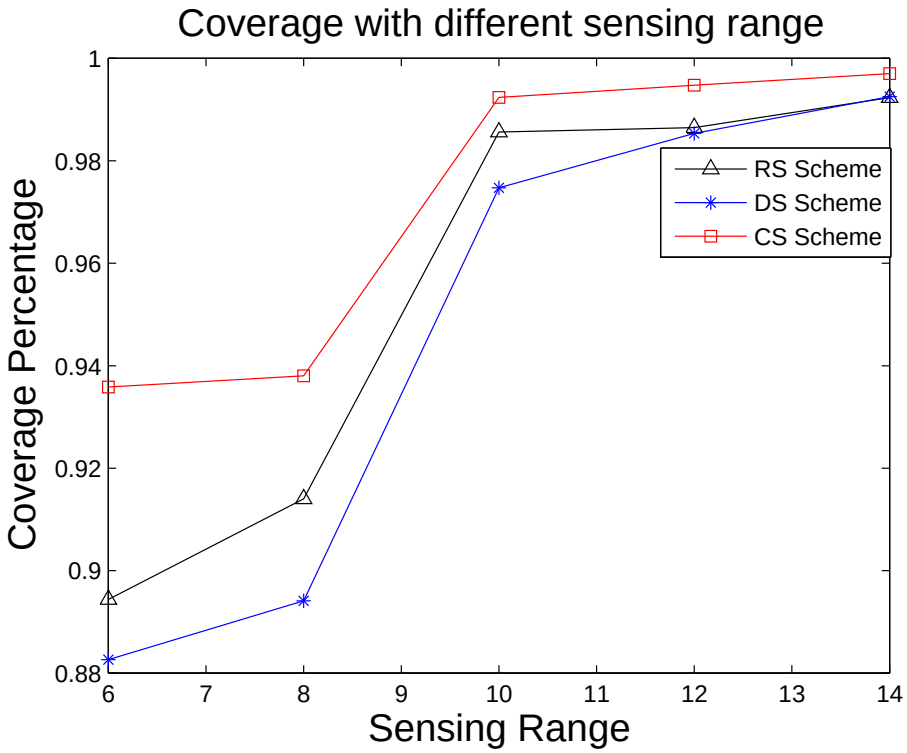

Figure 3.5: Impact of Sensing Range, $\beta_s = 0.10$

for all the sensors is small. When sensing ranges are greater than 10 meters, the three protocols achieve almost the same performance, because nodes have relatively large overlaps, so turning off a small number of nodes will not significantly degrade the performance of sensor networks. Figure 3.6 reveals a similar situation, with the only difference being that the total coverage for all the protocols is reduced proportionally when a higher percentage of nodes go to sleep. Based on the data presented in Figures 3.5 and 3.6, we can conclude that a small sensing range is not good for any of the sleep scheduling protocols. A large sensing range also contributes little to the cluster's overall coverage, so the optimum balance can be achieved when the sensing range is around 10 meters.

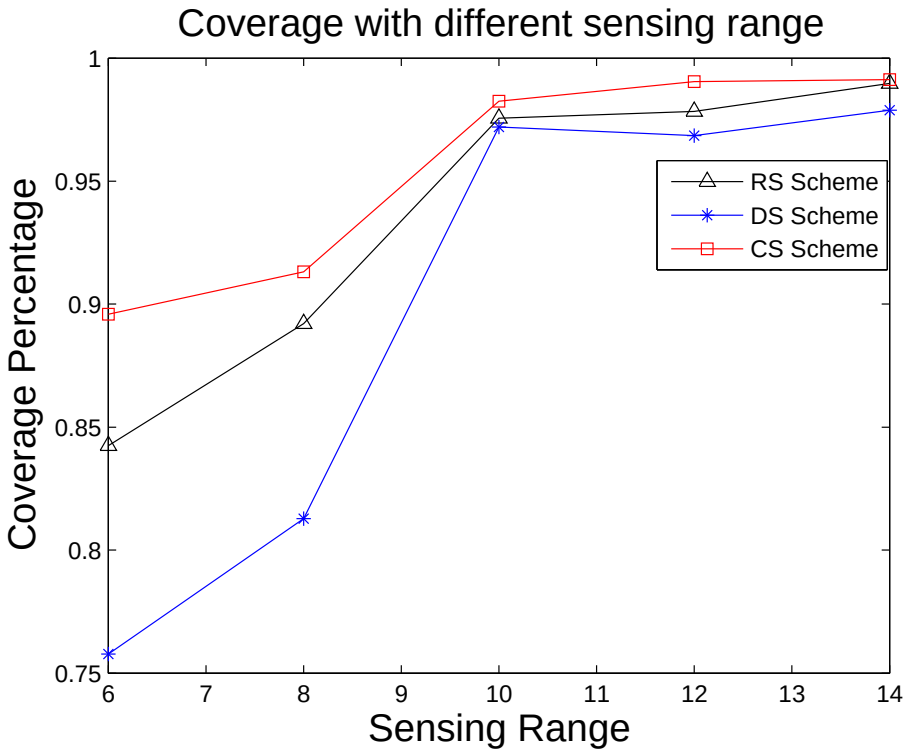

Figure 3.6: Impact of Sensing Range, $\beta_s = 0.30$

3.3.5 Impact of Random Sensing Range on Coverage

To further test the ability of the CS algorithm to achieve a high coverage for the cluster, we conducted experiments on sensor nodes with randomly assigned sensing ranges of 6 meters to 14 meters. The difference from the experiment described in the previous section is that here all the sensors are assigned a random sensing range, while previously all the sensors used the same sensing range in each experiment. The simulation settings were once again the default settings, and the performance of the three algorithms was compared by setting β_s to values in the range of $5\% \sim 40\%$. The simulation results are presented in Figure 3.7. As the figure shows, the coverage of all the protocols decreases with an increase in the number of sleeping nodes. When less than 25% of nodes

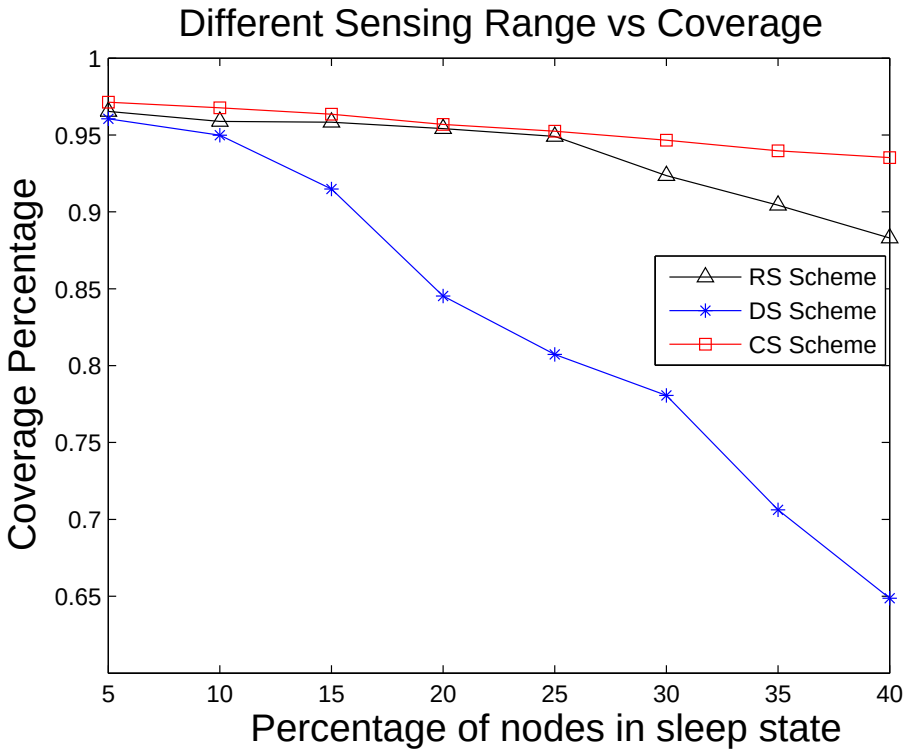

Figure 3.7: Impact of Random Sensing Range

are in the sleep state, the RS and CS algorithms exhibit almost the same performance. When the portion of nodes in sleep state exceeds 25%, the CS algorithm performs slightly better than the RS algorithm. From the same graph, we can see that when β_s is larger than 15%, the coverage of the DS algorithm becomes rapidly worse, dropping sharply with a gradually increasing percentage of nodes going to sleep. These results indicate that both the RS scheme and the CS algorithm are suitable for sensor networks with heterogeneous sensor nodes, with the CS algorithm achieving a slightly higher coverage. The DS scheme performs poorly in situations where sensors have different sensing ranges. Overall, the CS algorithm is best suited for networks made up of heterogeneous sensors with different sensing ranges.

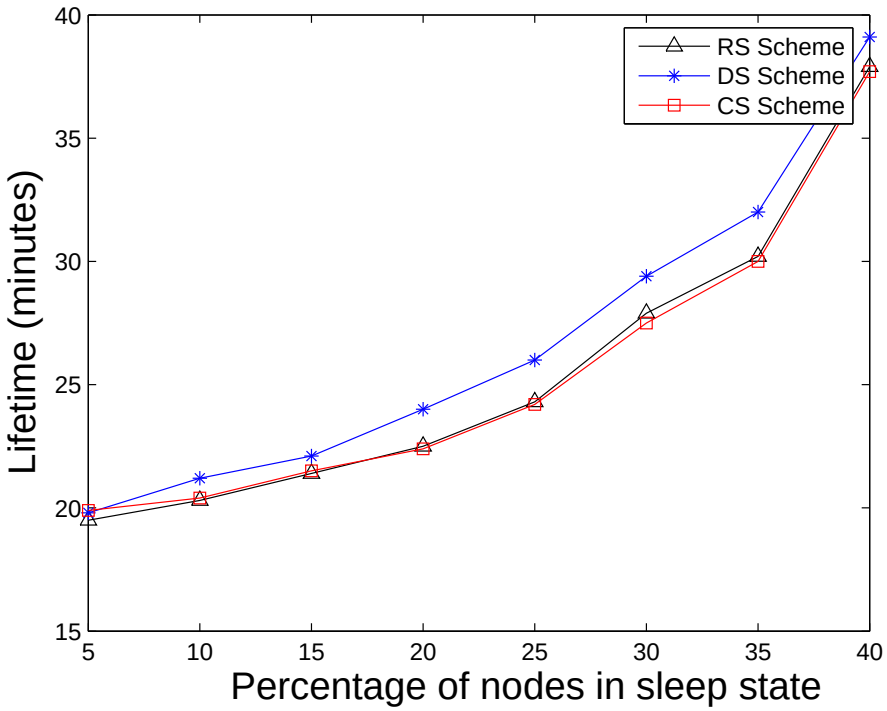

Figure 3.8: Impact of β_s on Lifetime, β_d=0.2

3.3.6 Impact of β_s on Network Lifetime

The impact of β_s on network lifetime for the three algorithms is shown in Figures 3.8 and 3.9. The simulation settings are the default values. The percentage of β_s is set as 5, 10, 15, 20, 25, 30, 35, and 40. Figure 3.8 sets $\beta_d = 0.2$, and Figure 3.9 sets $\beta_d = 0.5$. From Figure 3.8, we can see that overall the DS scheme achieves the best network lifetime under all the settings of β_s. Although the RS scheme and CS scheme achieve almost the same network lifetime, both have about 10 percent shorter lifetime than that of the DS algorithm. This is because the DS scheme preferentially uses nodes that are close to the cluster head, which consume less energy than nodes that are farther away from the cluster head. From Figure 3.9, with the increase of β_d, and when there are less than 25% of

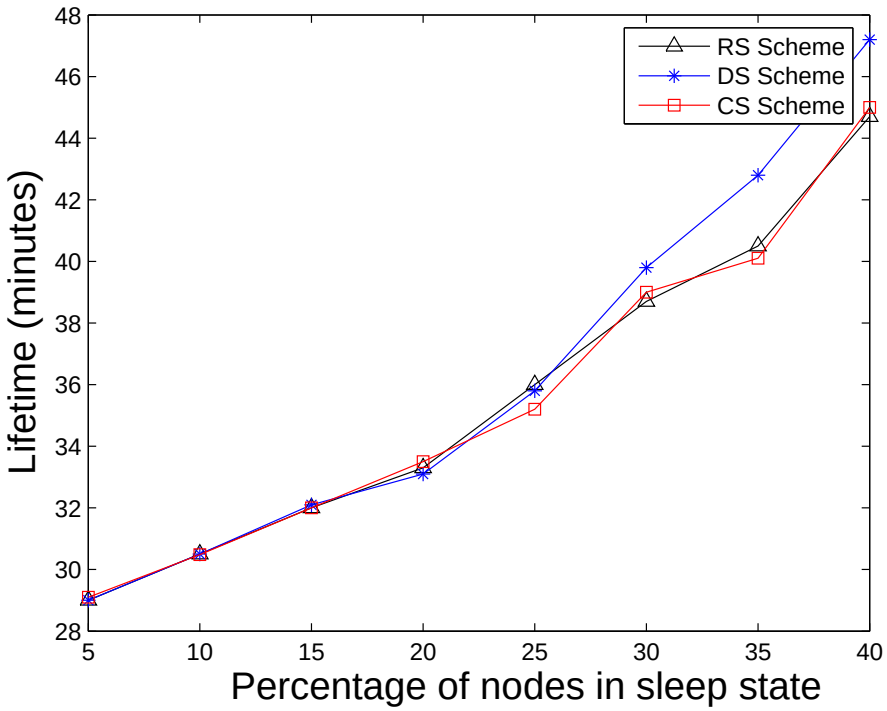

Figure 3.9: Impact of β_s on Lifetime, β_d=0.5

nodes in the sleep state, all three algorithms achieve almost the same lifetime. Above this point, the DS scheme performs slightly better than the other two, which means that the DS scheme is effective only when a high percentage of nodes are sleeping. If fewer than 25% of nodes are sleeping, and when the value of β_d is large, there is no significant difference between the three algorithms in terms of lifetime.

3.3.7 Impact of Different Initial Energy on Network Lifetime

To further compare the effectiveness of the three algorithms, we tested their lifetimes for sensor nodes with different initial energies, once again using the default values for these experiments. Each

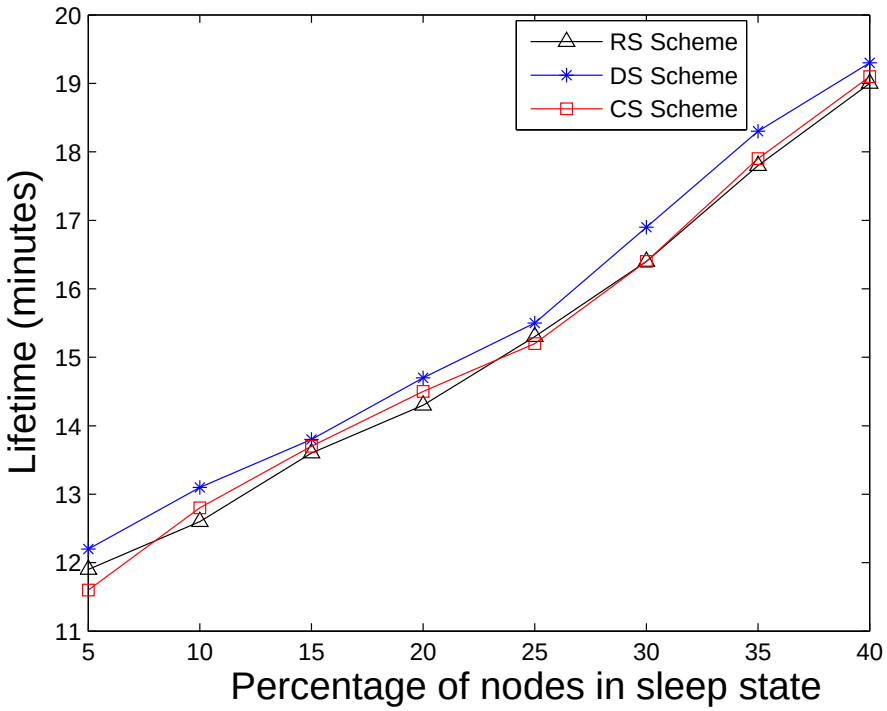

Figure 3.10: Impact of Different Initial Energy, β_d=0.2

node was randomly assigned an initial energy ranging from $500J$ to 1000J. The percentage of β_s is set as 5, 10, 15, 20, 25, 30, 35 and 40. The results are presented in Figures 3.10 and 3.11. Figure 3.10 sets $\beta_d = 0.2$ and Figure 3.11 uses $\beta_d = 0.5$. As the figures show, all three algorithms result in almost the same network lifetimes for all the settings of β_s, no matter whether β_d is large or small. This implies that none of the three algorithms adapt well to situations in which the nodes have different initial energies. These results also indicate that the DS algorithm is particularly ineffective in achieving better network lifetimes when nodes have different initial energies. The possible reason is as follows. There are more nodes with high initial energies which are far away from the cluster head, and the DS scheme puts a disproportionally large number of nodes with high initial energies

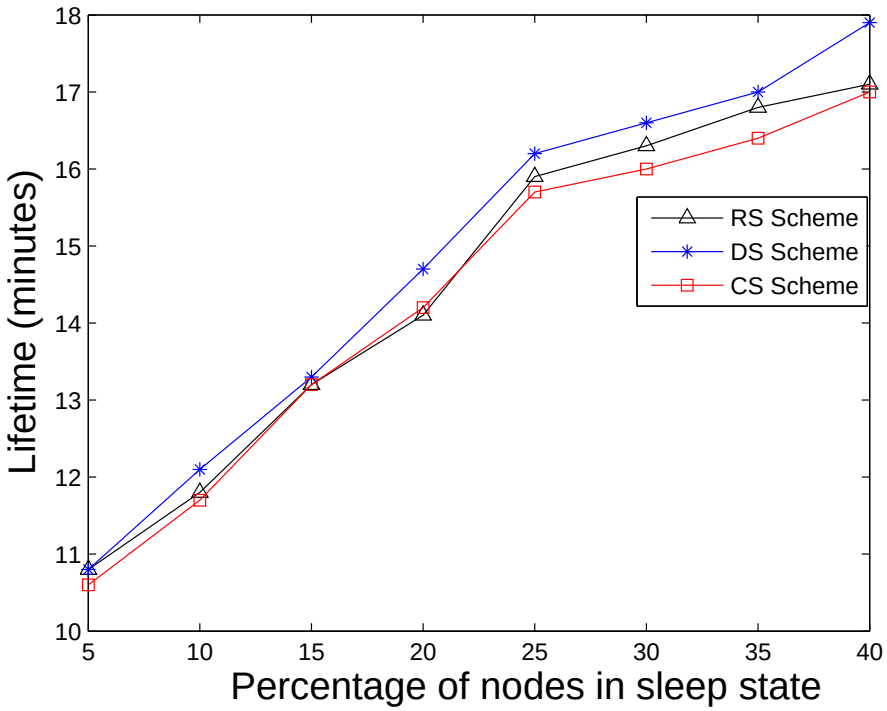

Figure 3.11: Impact of Different Initial Energy, β_d=0.5

to sleep. However, the nodes with lower initial energies remain active during most cycles. Thus they exhaust their batteries quickly. This does not affect either the RS scheme or the CS algorithm, as neither is sensitive to nodes with different initial energies.

3.4 Summary

In this chapter, we discussed the cluster node sleep scheduling problem in Wireless Sensor Networks. Specifically, we developed the CS algorithm to improve the coverage for the whole cluster. The basic idea for the CS algorithm is to schedule the nodes whose sensing coverage have higher overlap with their neighbors to have higher probabilities to sleep in each cycle. It will also

schedule the sensor nodes that have less overlap with their neighbors to be in active states with higher probabilities. In our simulation, we found that the CS algorithm achieved higher coverage than those of the RS and DS schemes, while guaranteeing almost the same lifetime for the cluster. Simulation results also showed that the CS algorithm adapts well to sensor nodes in clusters using different sensing ranges.

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 Conclusion

In this dissertation, we investigate two problems in wireless networks: Transmission Control Protocol(TCP) enhancement and cluster node sleep scheduling.

Built on top of Explicit Congestion Notification (ECN), we propose a new TCP enhancement for wireless mesh networks (WMNs), Congestion Coherence. The proposed enhancement explores the coherence of ECN markings in neighboring packets to decouple the end-to-end congestion signals from the local operations of wireless retransmissions and multipath routing. This allows different types of packet loss to be properly addressed by the TCP mechanism. Simulations showed that Congestion Coherence dramatically reduces the damage to TCP performance, and therefore, provided a unified solution to the challenges of TCP in WMNs.

Network coverage is of crucial importance for the operation of cluster-based sensor systems, therefore, it is necessary to give equal attention to network lifetime and network coverage in cluster-based sensor networks. In this dissertation, we proposed the coverage-aware sleep scheduling (CS) algorithm to improve the coverage of the cluster with no adverse effects on its lifetime. The fundamental concept governing the design of the CS algorithm is to assign the nodes with the highest sensing coverage overlap with their neighbors to be in the sleep state with the highest probabilities in each cycle. At the same time, it schedules the sensor nodes with less overlap to remain active with higher probabilities.

The simulations revealed that CS algorithm maintained higher coverage than either the randomized scheduling and distance-based scheduling schemes, while guaranteeing the same lifetime

for the cluster. Simulation results also showed that CS algorithm adapted well to clusters containing sensor nodes with different sensing ranges and different initial energies.

In summary of the contribution, we enhance the Transmission Control Protocol(TCP) and solve the cluster node sleep scheduling problem in this dissertation.

- With the proposed TCP enhancement protocol, the efficiency of TCP is improved significantly, thus the network performance is improved.
- Our proposed coverage-aware node sleep scheduling algorithm achieves a good trade-off between lifetime and coverage of wireless sensor networks. This helps improve the performance of the cluster-based sensor networks with less cost.

4.2 Future Work

4.2.1 Further Solution of Node Sleep Scheduling Problem

In the future, we plan to apply the coverage-aware sleep scheduling (CS) algorithm to the setting of dynamic cluster formation. In addition, we also want to reduce the computation complexity of the CS algorithm. Finally, we want to develop the coverage-aware sleep scheduling algorithm using a distributed approach.

4.2.2 Connected Dominating Set Problem

Other than the node sleep scheduling problem, we are planning to investigate the connected dominating set (CDS) problem [3]. A virtual backbone is essentially a connected dominating set. It is defined as a subset of nodes in a network, such that each node in the network is either in the set or a neighbor of some node in the set, and the induced graph of the nodes in the set is connected.

CDS has been extensively used for routing and broadcast in mobile ad hoc networks and wireless mesh networks. While existing protocols are successful in constructing virtual backbones with competitive size using localized information in mobile ad hoc networks, they lack the mechanism to recover when a virtual backbone becomes corrupted. Moreover, most of the past research are under the assumption of a single radio and a single channel environments in mobile ad hoc networks, which are not suitable for multi-radio wireless mesh networks. In the future, we want to develop a protocol to construct and maintain virtual backbones without introducing much communication overhead in multi-radio wireless mesh networks.

BIBLIOGRAPHY

[1] http://www.ieee802.org/11/.

[2] http://standards.ieee.org/getieee802/, *IEEE 802 working group.*

[3] K. Sakai, F. Shen, and M.-T. Sun, "Multi-Initiator Connected Dominating Set Construction for Mobile Ad Hoc Networks," *Proc. Int'l Conf. Comm.,* pp. 2431-2436, May 2008.

[4] C. Liu, F. Shen, M.-T. Sun, "A Unified TCP Enhancement for Wireless Mesh Networks," *Proc. 2007 Int'l Conf. Parallel Processing Workshops,* pp. 71-76, Sep. 2007.

[5] F. Shen, M.-T. Sun, "Coverage-aware Sleep Scheduling for High Density Cluster-based Sensor Networks," *Auburn University Computer Science and Software Engineering Technical Report, CSSE08-01,* Jun. 2008.

[6] IEEE LAN MAN Standards Committee, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11-1997," *The Institute of Electrical and Electronics Engineers,*1997.

[7] http://standards.ieee.org, *IEEE standards organization.*

[8] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," *IEEE Std 802.11b-1999/Cor 1-2001 (R2003).*

[9] IEEE LAN MAN Standards Committee, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11a-1999," *The Institute of Electrical and Electronics Engineers,* 1999.

[10] "IEEE 802.11g Standard," standards.ieee.org/getieee802/download/802.11g-2003.pdf.

[11] F. Andre, J.-M. Bonnin, B. Deniaud, K. Guillouard, N. Montavont, T. Noel, and L. Suciu, "Optimized Support of Multiple Wireless Interfaces within an IPv6 End-Terminal," *Smart Object Conference,* 2003.

[12] J. Yang, K. Sakai, B. Kim, H. Okada, and M-T Sun, "Cost-Aware Route Selection in Wireless Mesh Networks," *Int'l Conf. Mobile Ad Hoc and Sensor Networks,* pp. 171-184, 2006.

[13] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou, "A Multi-radio Unification Protocol for IEEE 802.11 Wireless Networks," *Int'l Conf. Broadband Communications,Networks and Systems,* pp. 344-354,2004.

[14] I.F. Akyildiz, G. Morabito and S. Palazzo, "TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks," *IEEE/ACM Trans. Networking,* vol. 9, no. 3, pp. 307-321,2001.

[15] I.F. Akyildiz, X. Zhang and J. Fang, "TCP-Peach+: Enhancement of TCP-Peach for Satellite IP Networks," *IEEE Commun. Lett.,* vol. 6,no. 7, pp. 303-305, 2002.

[16] I.F. Akyildiz, X. Wang and W. Wang, "Wireless Mesh Networks: A Survey," *Computer Networks,* vol. 47,no. 4, pp. 445-487, 2005.

[17] M. Allman, V. Paxson and W. Stevens, "TCP Congestion Control," *RFC 2581,* 1999.

[18] A. Bakre, B.R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," *Proc. Int'l Conf. Distributed Computing Systems,* pp. 136-146, 1995.

[19] H. Balakrishnan, S. Seshan and R.H. Katz,"Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks," *Wireless Networks,* vol. 1, no. 4, pp.469-481,1995.

[20] H. Balakrishnan and R.H. Katz, "Explicit Loss Notification and Wireless Web Performance," *Proc. IEEE Global Comm. Conf.,* 1998.

[21] S. Banerjee and J. Goteti, "Extending TCP for Wireless Networks," *University of Maryland, College Park,* 1997.

[22] P. Bhagwat, P. Bhattacharya, A. Krishna, S.K. Tripathi, "Using Channel State Dependent Packet Scheduling to Improve TCP Throughput over Wireless LANs," *ACM/Baltzer Wireless Networks Journal,* vol. 3, no. 1, pp.91-102,1997.

[23] S. Bhandarkar, N. Sadry, A.L.N. Reddy, and N. Vaidya, "TCP-DCR: A Novel Protocol for Tolerating Wireless Channel Errors," *IEEE Tran. Mobile Computing,* vol. 4, no. 5,pp517-529, 2004.

[24] S. Biaz, N. Vaidya et al, "TCP over Wireless Networks using Multiple Acknowledgment," *Texas A&M University, Technical Report 97-001,* 1997.

[25] S. Biaz and X. Wang, "Can ECN Be Used to Differentiate Congestion Losses from Wireless Losses?" *Technical Report CSSE04-04*, Computer Science and Software Engineering Dept., Auburn University, 2004.

[26] E. Blanton and M. Allman, "On making TCP more Robust to Packet Reordering," *ACM SIGCOMM Computer Communication Review,* vol. 32,no. 1, pp.20-30, 2002.

[27] B. Braden et al, "Recommendations on Queue Management and Congestion Avoidance in the Internet," *RFC 2309,* 1998.

[28] L.Brakmo, S.O'Malley and L.Peterson, "TCP Vegas: New techniques for Congestion Detection and Avoidance," *Proc. ACM Conf.Applications, Technologies, Architectures, and Protocols for Computer Communications,* pp.24-35, 1994.

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

[30] D.A. Eckhardt and P. Steenkiste, "Improving wireless LAN Performance via Adaptive Local Error Control," *IEEE Int'l Conf. Network Protocols,* 1998.

[31] S. Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communication Review,* vol. 24, no. 5, pp. 10-23,1994.

[32] S.Floyd and V.Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Trans. Networking,* vol. 1, no. 4, pp. 397-413,1993.

[33] R. Jain, "A Timeout-based Congestion Control Scheme for Window Flow-controlled Networks," *IEEE Journal on Selected Areas in Communications,* vol. 4, no. 7, pp. 1162-1167, 1986.

[34] K. Leung , V. Li and D. Yang, "An Overview of Packet Reordering in Transmission Control Protocol (TCP): problems, solutions, and challenges," *IEEE Tran. Parallel and Distributed Systems,* vol 18, no. 4, pp.522-535,2007.

[35] C. Liu and R. Jain. "Improving Explicit Congestion Notification with the Mark-front Strategy," *Computer Networks,*, vol. 35, no. 2-3,pp.185-201,2001.

[36] R. Ludwig, A. Konrad, A.D. Joseph, "Optimizing the End-to-end Performance of Reliable Flows over Wireless Links," *ACM Annual Int'l Conf. Mobile Computing and Networking,* pp. 113-119, 1999.

[37] R. Ludwig, R.H. Katz, "The Eifel Algorithm: Making TCP Robust against Spurious Retransmissions," *ACM Computer Communication Review,* vol. 30, no. 1, pp.30-36, 2000.

[38] C. Ma and K. Leung, "Improving TCP Reordering Robustness in Multipath Networks," *IEEE Conf. Local Computer Networks,* pp. 409-410, 2004.

[39] M. Meyer, "TCP Performance over GPRS," *IEEE Wireless Communications and Networking Conf.,* pp.1248-1252, 1999.

[40] C. Parsa and J.J.Garcia-Luna-Aceves, "Improving TCP Congestion Control over Internets with Heterogeneous Transmission Media," *Proc. Seventh Annual Int'l Conf. Network Protocols,* pp.213-221,1999.

[41] K. Ramakrishnan and S. Floyd, "A Proposal to Add Explicit Congestion Notification (ECN) to IP," *RFC 2481,*1999.

[42] K. Ramakrishnan, S. Floyd and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," *RFC 3168,* 2001.

[43] A. Raniwala and T. Chiueh, "Architecture and Algorithms for an IEEE 802.11-based Multi-channel Wireless Mesh Network." *IEEE Conf. Computer Communications,* pp. 2223-2234,2005.

[44] P. Sinha, N. Venkitaraman, R. Sivakumar and V. Bharghavan, "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks," *ACM Annual Int'l Conf. Mobile Computing and Networking,* pp.301-316,1999.

[45] UCB/LBNL/VINT Network Simulator - ns, http://www-mash.CS.Berkeley.EDU/ns/.

[46] N. H. Vaidya, M. Mehta, C. Perkins and G. Montenegro, "Delayed Duplicate Acknowledgments: a TCP-unaware Approach to Improve Performance of TCP over Wireless," *Technical Report 99-003,* Computer Science Dept., Texas A&M University,1999.

[47] K. Xu, Y. Tian and N. Ansari, "TCP-Jersey for Wireless IP Communications," *IEEE Jour. Selected Area in Comm.,* vol. 22, no. 4, pp.747-756, 2004.

[48] S. Xu, T. Saadawi and M. Lee, "On TCP over Wireless Multi-hop Networks," *IEEE Military Comm. Conf.,* pp.282-288, 2001.

[49] G. Xylomenos and G.Polyzos, "Intenet Protocol Performance over Networks with Wireless Links," *Computer Networks,* vol. 37, no. 5, pp.601-615, 2001.

[50] G. Xylomenos, "Multi Service Link Layers: An Approach to Enhancing Internet Performance over Wireless Links," *PhD dissertation at University of California, San Diego,*1999.

[51] M. Zhang, B. Karp, S. Floyd, and L. Peterson, "RR-TCP: A Reordering-Robust TCP with DSACK," *IEEE Conf. Network Protocols,* pp.95-106, 2003.

[52] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[53] "IEEE 802.15. 4, ZigBee standard," http://www.zigbee.org/en/index.asp, 2004.

[54] O. Younis, S. Fahmy, "HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Trans. Mobile Computing*, vol. 3, no. 4, pp. 366–379, Dec. 2004.

[55] J. Deng, Y. S. Han, W. B. Heinzelman, and P. K. Varshney, "Scheduling Sleeping Nodes in High Density Cluster-based Sensor Networks," *ACM/Kluwer Mobile Networks and Applications (MONET) Special Issue on "Energy Constraints and Lifetime Performance in Wireless Sensor Networks"*, vol. 10, no. 6, pp. 825–835, Dec. 2005.

[56] F. Chen and Q. Zhao, "On the Lifetime of Wireless Sensor Networks," *IEEE Communications Letters*, vol. 9, no. 11, pp. 976–978, Nov. 2005.

[57] V. Rai and R. N. Mahapatra, "Lifetime Modeling of a Sensor Network," *Proc. Conf. Design, Automation and Test in Europe*, vol. 1, pp.202–203, Mar.2005.

[58] H. Ritter, T. Voigt, "Experimental Evaluation of Lifetime Bounds for Wireless Sensor Networks," *Proc. Second European Workshop Wireless Sensor Networks*, pp. 25–32, Feb. 2005.

[59] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage Problems in Wireless Ad-hoc Sensor Networks," *The 20th Conf. Computer Comm.*, pp. 1380–1387, 2001.

[60] N. Ahmed, S. S. Kanhere, S. Jha, "Probabilistic Coverage in Wireless Sensor Networks," *Proc. IEEE Conf. Local Computer Networks 30th Anniversary*, 2005.

[61] H. Zhang, J. Hou, "Maintaining Sensing Coverage and Connectivity in Sensor Networks," *Int'l Workshop on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer Networks*, Feb. 2004.

[62] B. Liu, D. Towsley, "On the Coverage and Detectability of Wireless Sensor Networks," *Proc. of WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

[63] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A Two-tier Data Dissemination Model for Large-scale Wireless Sensor Networks," *The Annual Int'l Conf. Mobile Computing and Networking* , pp.148–159, 2002.