5-2010

CROSS-LAYER SCHEDULING PROTOCOLS FOR MOBILE AD HOC NETWORKS USING ADAPTIVE DIRECT-SEQUENCE SPREAD-SPECTRUM MODULATION

Brian Wolf
Clemson University, bjwolf@clemson.edu

CROSS-LAYER SCHEDULING PROTOCOLS FOR MOBILE AD HOC NETWORKS USING ADAPTIVE DIRECT-SEQUENCE SPREAD-SPECTRUM MODULATION

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Electrical Engineering

---

by
Brian James Wolf
May 2010

Accepted by:
Dr. Harlan Russell, Committee Chair
Dr. Daniel Noneaker
Dr. Kuang-Ching Wang
Dr. Jim Martin

ABSTRACT

We investigate strategies to improve the performance of transmission schedules for mobile ad hoc networks (MANETs) employing adaptive direct-sequence spread-spectrum (DSSS) modulation. Previously, scheduling protocols for MANETs have been designed under the assumption of an idealized, narrowband wireless channel. These protocols perform poorly when the channel model incorporates distance-based path loss and co-channel interference. Wideband communication systems, such as DSSS systems, are more robust in the presence of co-channel interference; however, DSSS also provides multiple-access capability that cannot be properly leveraged with a protocol designed for narrowband systems. We present a new transmission scheduling protocol that incorporates link characteristics, spreading factor adaptation, and packet capture capability into scheduling and routing decisions. This provides greater spatial reuse of the channel and better adaptability in mobile environments. Simulation results demonstrate the merits of this approach in terms of end-to-end packet throughput, delay, and completion rate for unicast traffic. We also discuss two variations of the protocol: one provides a method for enhancing the network topology through exchange of local information, and the other leverages multi-packet reception (MPR) capability to enhance the network topology. We show that each approach is useful in networks with sparse connectivity. We conclude by studying the capacity of the networks used in previous sections, providing insight on methods for realizing further performance gains.

DEDICATION

First and foremost, this work is dedicated to my lovely wife, Joi, who has shared my sacrifice, and now shares my achievement. You make everything worthwhile.

I also dedicate this work to Darius Jones, who has been a friend, mentor, teacher, and inspiration during my time at Clemson.

ACKNOWLEDGEMENTS

I am deeply grateful to my advisor, Dr. Harlan B. Russell, for his commitment, insight, and patient guidance throughout my graduate studies. I will greatly miss our weekly meetings and wide-ranging discussions.

I wish to thank the members of my committee, Dr. Daniel L. Noneaker, Dr. Kuang-Ching Wang, and Dr. James J. Martin for their valuable input during the preparation of this manuscript.

I would also like to thank the instructors in the wireless communications group at Clemson University: Dr. Michael B. Pursley, Dr. Carl W. Baum, and Dr. John J. Komo, as well as my advisor and committee members. It has been a privilege to receive my education from such a distinguished group of experts.

Finally, I thank my family and friends for their love and support.

TABLE OF CONTENTS

Page

Table of Contents (Continued)

Page

Table of Contents (Continued)

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

List of Figures (Continued)

Figure Page

List of Figures (Continued)

Figure Page

CHAPTER ONE

INTRODUCTION

A mobile ad hoc network (MANET) is a special type of wireless network in which terminals self-organize to communicate. Information is sent as packets, which must often traverse several network terminals to reach their destinations. Consequently, terminals in a MANET must be designed to act not only as a receiver and transmitter for the primary user, but also as traffic routers for other network users. MANETs are characterized by their unpredictability: network membership may change as terminals enter and leave the network, the quality of communication links varies due to terminal mobility, terrain features, and interference, and traffic demands fluctuate as users exchange voice, data, and video packets.

MANETs are designed to provide communications capability when wired and/or wireless communications infrastructure is not available. This is often the case in military operations, or during disaster relief when the existing infrastructure has been damaged. Some emerging commercial standards take cues from the MANET paradigm, including the IEEE 802.11 ad hoc mode for peer-to-peer communication, and the IEEE 802.16e standard for mobile internet access. The key features required for these applications are rapid deployment and robust, adaptive operation in a wide range of environments. Military applications additionally require security features, such as jamming resistance, low probability of intercept, confidentiality, and distributed control so that there is no single point of failure.

1.1 Medium Access Control Overview

The wireless channel is a shared communication medium, so protocols are required to govern how and when terminals may access the channel. These medium-access control, or MAC, protocols fall into one of two categories: contention-based, and contention-free. Contention-based protocols, such as ALOHA [1], CSMA [2], MACA [3], and other variants, allow a terminal to compete for access to the channel whenever a packet is available for transmission. Assuming the network is not heavily loaded, this is a very efficient strategy because terminals can attempt to access the channel at any time, and terminals with no traffic do not consume channel resources. However, as the traffic load increases, contention mechanisms break down. This results in transmission failures, unfairness, and excessive delay.

Contention-free MAC protocols divide the channel into separate sub-channels based upon time (TDMA), frequency (FDMA) or code (CDMA), which are then reserved by terminals for transmissions. These guaranteed reservations provide stable operation at high traffic loads. Contention-free medium access control is particularly beneficial when supported applications have quality-of-service (QoS) requirements since access to the channel is pre-determined. If contention-based access is used, there is a chance a terminal may go an extended period of time without successfully contending for use of the channel. By reserving dedicated time, frequency, or code sub-channels, terminals are guaranteed regular access to the channel. As a result, end-to-end delay and throughput vary less than in networks using contention-based access.

We focus on spatial TDMA (STDMA), which improves the utilization of pure TDMA by allowing terminals which are sufficiently far apart to reuse the channel (i.e., schedule transmissions in the same time slots). Specifically, we investigate protocols supporting broadcast transmissions to neighboring terminals. Broadcast transmission scheduling achieves slightly lower spatial reuse than protocols which only require successful link activation, but there are several advantages. First, not all network data traffic is unicast; many applications require sending data to a set of destinations, or all terminals in the network. Broadcast transmission scheduling makes this process more efficient since each transmission reaches many neighboring terminals.

Secondly, network control packets are often required to be sent to all nearby terminals, especially in support of routing protocols for ad hoc networks. For example, AODV routing [4] requires route request (RREQ) packet flooding to achieve route discovery; the flooding process is much more efficient when a single transmission can reach all neighboring terminals. If OLSR [5] is used to perform routing, then HELLO packets must be periodically broadcast to all neighbors and topology control (TC) messages must be exchanged between all relay terminals to disseminate link information. Since MANET terminals are expected to perform routing operations and handle traffic for a variety of applications, the stability and flexibility provided by broadcast transmission scheduling makes it an appropriate MAC strategy.

1.2 The Benefits of Direct-sequence Spread-spectrum In Ad Hoc Networks

In a direct-sequence spread-spectrum (DSSS) system, information bits are modulated at the transmitter by a higher rate pseudonoise, or PN, spreading sequence

known to the receiver. This process spreads the energy of the transmitted signal over a much larger bandwidth than is necessary for communication. At the receiver, the information bits may be recovered through synchronous correlation of the received signal with the spreading sequence. This results in several advantages over narrowband modulation. The noise-like qualities of the transmitted signal make it more difficult for third parties to detect active transmitters. Since the signal occupies a larger bandwidth, a hostile jammer must use more energy to disrupt communications. Furthermore, since the spreading sequences must be known to the receiver, eavesdropping is difficult. These advantages make DSSS modulation particularly applicable in military communication systems.

In addition to the above security features, DSSS provides multiple-access capability that is applicable in both military and commercial applications. In particular, if multiple signals overlap in time, space, and frequency at a receiver which is correlating to a particular DSSS signal, then the energy from the interfering transmissions is attenuated. This phenomenon, commonly referred to as *spreading gain*, results in more robust link performance. In an ad hoc network where link quality fluctuates rapidly and interference from other users is unpredictable, this added robustness may greatly improve network performance, as well as simplify protocol design.

For a single user, DSSS is less bandwidth-efficient than narrowband modulation approaches, such as BPSK. Also, reception of DSSS signals requires precise synchronization with the incoming signal during an *acquisition* phase. Acquiring and maintaining synchronization during reception is a challenging problem in its own right,

and the interested reader may consult a standard text, such as [6], for a more detailed examination of this topic. While we use a standard, simplified model for reception of DSSS signals designed to reflect the typical performance of such systems, the actual performance depends upon the correlation properties of the spreading sequences, signal acquisition and tracking performance, and the relative power levels of signals from multiple transmitters at a receiver. For example, performance analysis of DSSS systems which considers the correlation properties of the spreading sequences is provided in [7].

There are several strategies for assigning spreading sequences, also called *spreading codes*, in networks utilizing DSSS modulation. These include *common code*, *receiver-oriented* code assignment, and *transmitter-oriented* code assignment. In common code systems, all transmissions use the same spreading sequence. In this case, acquisition of a signal from a particular transmitter is difficult because there is no easy way to differentiate between transmitters. If receiver-oriented code assignment is used, transmitters use a code associated with the receiver to which they are sending a packet. Broadcasting a message to multiple receivers is difficult because they all use different codes. If transmitter-oriented code assignment is chosen, each transmitter uses its own unique spreading sequence for all transmissions. Receivers must select which transmitter sequence to correlate with before attempting to receive a packet. The benefit of this method is that broadcasting data to multiple receivers is simpler since all intended receivers can correlate with the transmitter's spreading sequence.

1.3 Modeling the Wireless Channel

The evaluation of any wireless communication protocol relies heavily upon the model of the wireless environment used for testing. While wireless environment models are examined in greater detail in later chapters, we note here that one of the main contributions of this work is that the protocols are designed to operate in the *physical interference model*. This model accounts for some of the key features of the wireless environment, such as large-scale fading proportional to signal propagation distance and aggregate multiple-access interference (MAI) from distant transmitters. Successful packet reception is possible when a signal-to-interference-plus-noise, or SINR, threshold is satisfied at the receiver.

In contrast, much of the previous work in distributed transmission scheduling protocols has assumed a simplified graph model which accounts for neither fading proportional to distance nor aggregate MAI. However, in real systems, there is a significant interaction between the transmission schedule, MAI, and the quality of communications links. Hence, schedules produced under a graph model may perform quite differently in reality. In particular, in several recent papers (e.g., [8], [9], [10], and [11]) it is noted that schedules developed under a graph model perform poorly in the physical interference model. This motivates the development of new protocols which are explicitly designed under the physical interference model.

1.4 Problem Statement

We develop distributed protocols to support reliable communications in highly dynamic MANETs using DSSS modulation and broadcast transmission scheduling under

the physical interference model. These protocols adapt the schedule to the changing environment without incurring large overhead costs. At the same time, the protocols maintain a high transmission success rate and an efficient channel assignment.

We assume network membership and topology is dynamic. Terminals operate on a single communication channel using half-duplex transceivers; hence, they may not simultaneously receive and transmit. We use a novel DSSS transmission format employing both common code and transmitter-oriented code assignment. Packet transmissions must satisfy a signal-to-interference-plus-noise (SINR) threshold at a receiver which has pre-selected the transmitter spreading sequence to be successfully received. Terminals have no knowledge of signal path or channel gain, but are able to form estimates of SINR for received packets.

We present a protocol which leverages the multiple access capability of DSSS to achieve improved spatial reuse and faster adaptation in mobile environments when compared to traditional scheduling approaches. The use of common-code DSSS modulation allows each terminal to identify neighboring terminals. The set of neighboring terminals detected in this manner is used in a distributed scheduling algorithm to determine appropriate transmission times. Periodic control packets allow terminals to establish communication links with appropriate neighboring terminals, and determine times at which these links may be utilized to support point-to-point and broadcast transmissions using transmitter-oriented DSSS modulation. For broadcast transmissions, terminals employ a conservative spreading gain to maximize coverage. For point-to-point transmissions, terminals use link-SINR estimates to dynamically adjust

spreading gain for each transmission, allowing link performance to be improved. Terminals employ a queue management policy designed to take advantage of these variable link data rates and reduce control packet overhead. We also describe new routing metrics to take advantage of these capabilities.

We show, using network-level simulations with end-to-end packet statistics, how this approach results in appreciable gains in performance. We also develop and analyze two variations of the protocol. First, we show how additional topology information may be shared among terminals to provide a higher level of network connectivity. Second, we show how multi-packet reception capability improves network connectivity by increasing the availability of receivers. In the final chapter, we use an idealized channel access strategy to study the capacity of wireless networks. Specifically, we analyze features used by the distributed protocol, such as transmission rate adaptation and multi-packet reception, in this setting to determine their influence on achievable throughput capacity.

The rest of this manuscript is organized as follows: background material and related work are presented in Chapter 2. In Chapter 3, we evaluate several distributed scheduling algorithms to identify which is best for implementation in a distributed protocol. In Chapter 4, we present channel and receiver models which are used throughout the manuscript, as well as simulation settings. In Chapter 5, the distributed protocol is motivated and presented; performance results are given in Chapter 6. Chapters 7 and 8 develop the two protocol variants discussed above. Chapter 9 provides a study of throughput capacity using a centralized algorithm, and concluding remarks are given in Chapter 10.

CHAPTER TWO

BACKGROUND AND RELATED WORK

The design of scheduled MAC protocols for MANETs has received considerable attention in the literature. The *Time Slot Assignment Problem*, or TSAP, is a classic formulation of the problem. In its most general form, the goal of the TSAP is the assignment of transmission opportunities (time slots) to network terminals in a repeating frame satisfying some set of constraints [12]. The network is modeled as a graph $G=(V,E)$, where the vertex set V represents the wireless terminals and the edge set E represents links between terminals that may communicate directly. Two terminals are deemed *1-neighbors* if they are connected by an edge, *2-neighbors* if they have a common 1-neighbor, and so on. The schedule is required to be *collision-free*, where the term *collision* refers to co-channel interference that leads to transmission failure. In the graph model, a collision occurs at a receiver when two or more 1-neighbors transmit in the same slot; since terminals cannot transmit and receive at the same time, a collision also occurs if two 1-neighbors are assigned the same transmission slot.

Collision-free schedules are defined differently depending upon whether *broadcast scheduling* or *link scheduling* is used. For broadcast schedules, the graph vertices are assigned transmission slots and collisions are disallowed at all 1-neighbors. For link schedules, directional edges are assigned transmission slots and collisions are disallowed only at the intended receiver [13]. Formally, a collision-free broadcast schedule allows two terminals i and j to transmit in the same slot if:

(i-b) edge $(i, j) \notin E$ and edge $(j, i) \notin E$, and

(ii-b) there exists no terminal k for which edge $(i, k) \in E$ and $(j, k) \in E$.

A collision-free link schedule allows concurrent link activation of links (i, j) and (k, l) if:

(i-l) i, j, k and l are mutually distinct

(ii-l) $(k, j) \notin E$ and $(i, l) \notin E$

In Figure 1 (a) and (b), collisions for broadcast schedules are illustrated which violate rule (i-b) and rule (ii-b), respectively. If the transmitting terminals (highlighted) are separated by at least two hops, as in Figure 1 (c), collisions are avoided. In Figure 2, collisions for link schedules are illustrated, with active links denoted by arrows. In part (a), rule (i-1) is violated, while in part (b), rule (ii-l) is violated.

Myriad variations on the TSAP are realized by considering additional objectives, such as designing schedules which minimize end-to-end delay [14][15], balance traffic loads [16][17][18], or minimize the length of the repeating transmission frame to enable more frequent transmissions [19]. Other ways in which protocols vary are centralized vs. distributed implementation, use of different wireless channel models, such as the physical interference model, and the use of special signaling and/or contention periods to aid in scheduling transmissions.

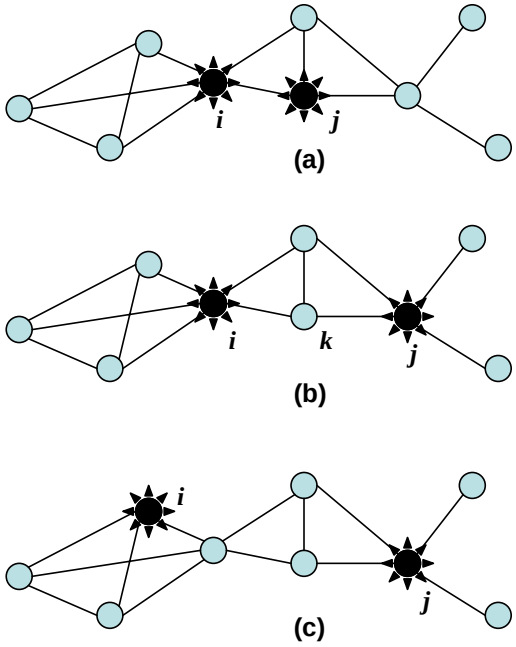


Figure 1. Under broadcast scheduling, transmitting terminals (highlighted) must be more than two hops apart. Scenarios (a) and (b) are prohibited, while (c) is allowable.

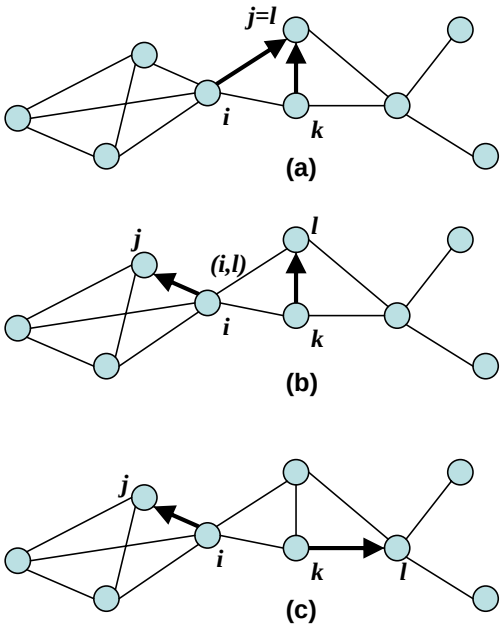


Figure 2. Under link scheduling, any transmission configuration that ensures receiving terminals are within range of only one transmitter is allowed. Scenarios (a) and (b) are prohibited, while (c) is allowable.

2.1 Centralized Scheduling Algorithms

Centralized scheduling algorithms depend upon global information to create transmission schedules. The resulting schedules achieve optimal or near-optimal performance, depending upon the scheduling criteria used. Centralized algorithms scale poorly to large networks due to the burden involved in collecting and distributing global information. However, they remain an important area of research since they aid in appreciating the complexity of the scheduling problem, give insight into the design of distributed protocols, and provide useful performance benchmarks.

The broadcast scheduling problem, or *BSP*, is defined as the generation of the minimum-length collision-free broadcast transmission schedule which guarantees each terminal at least one transmission per frame. In [20], the BSP is shown to be NP-hard, motivating solutions based upon sophisticated optimization algorithms. In [21], mean-field annealing is used to generate minimal-length schedules, while similar schedules are generated in [22] using a Hopfield Neural Network, in [23] using a genetic algorithm, and in [24] with a mixed neural-genetic algorithm. In [25], simulated annealing is used to generate schedules which achieve maximal stable throughput for a given traffic load and frame length. Direct solution of this problem involves finding maximally-constrained vertex cliques, and this sub-problem is itself NP-hard, as shown in [26].

In [27], a linear program is used to compute minimal-length schedules using the physical interference model as a constraint. Linear programming has been frequently used to solve the problem of determining a transmission schedule, packet routing, and/or transmission parameters (power/rate) for a given network topology satisfying a given

input traffic vector. In this case, the problem may be solved as a multi-commodity flow linear program [28]. In [29], linear programming is used to compute minimal-length link schedules with link flow constraints, also under the physical interference model. Minimal-length scheduling with joint rate-control is similarly considered in [30]. In [31], joint scheduling and power control is achieved by alternating between two algorithms that address each problem individually. There are many other centralized variations; however, we now turn our attention to distributed scheduling protocols.

2.2 Distributed Graph-Based Scheduling Protocols

Numerous distributed scheduling protocols have been developed using the graph model described above. These may be classified as frame-based or random scheduling approaches. In some frame-based approaches (e.g. [20] and [32]), the transmission frame length is a global parameter; in the worst case, a fully-connected network requires a frame length equal to the number of network terminals to support broadcast transmissions. Specifically, in [20], a skeleton schedule is created by setting the frame length equal to the number of terminals and assigning each terminal the transmission slot corresponding to its ID number. Additional transmission slots are assigned by priority after exchange of local topology information using the skeleton schedule. In [32], the network graph is colored so that no terminals within two hops have the same color; the frame length is equal to the number of colors used. Unfortunately, if the topology changes due to terminal mobility, the schedule may no longer be collision-free. In [33] and [34], the network graph is colored in a similar fashion, but frame lengths are a power-of-two and may vary depending upon local terminal density. Multiple authors have

independently developed power-of-two scheduling, for example [35] and [36]. In [37], protocols are developed which maintain collision-free operation of schedules based upon the algorithm in [33] in mobile environments. In [38], the algorithm in [33] is also used in a protocol which allows construction of broadcast schedules in network initialization scenarios.

Some hybrid protocols use structured contention to allow terminals to reserve transmission slots. For example, in [39], periodic contention frames allow terminals to negotiate a new schedule with neighbors. As long as the contention frames occur fairly regularly in comparison to topology changes, the schedule is largely collision-free. In [36], periodic *bootstrap* slots allow terminals to make reservations for transmissions in later frames. In [40], each transmission slot is preceded by a contention period. A skeleton schedule, similar to the one used in [20], is used in [40], but terminals may contend for access in slots that are not assigned to them on the condition they do not interfere with regularly scheduled transmissions which are given priority. This requires a four-phase contention mechanism before data is transmitted: priority RTS, priority CTS, contention RTS, and contention CTS. In [41] a schedule is created by iterating through several rounds in which terminals run a lottery for requesting slots. After the lottery is completed terminals begin using the schedule. If the topology changes the lottery must be run again.

Other protocols do no use a transmission frame at all; instead, random priority generation in each slot dictates channel access. In [42], each terminal is assigned a random number seed which is shared with 1 and 2-neighbors. In each slot, if a terminal

has a packet to transmit it becomes active with probability p by generating a pseudorandom number between 0 and 1, using its own seed. Transmissions are not collision-free, but collisions are greatly reduced by requiring that active terminals transmit with a probability that is inversely proportional to the number of other active 1 and 2-neighbors. In [43], a protocol called NAMA (node activation multiple access) uses hash functions to compute pseudorandom priorities for terminals in each time slot; a terminal may transmit if it generates the highest priority among its 1- and 2-neighbors.

One drawback of the graph model is that strong interference may be caused by terminals which are more than two hops away in the topology graph. For a simple example, consider Figure 3 in which terminals separated by n-hops are actually in close proximity. If terminal 2 and terminal n+1 are assigned the same transmission slot, interference from terminal n+1 may cause interference at terminal 1 that is not accounted for in the topology graph. This motivates an extended graph model called a *conflict graph*. In a conflict graph, links of the original graph are represented by vertices, and an edge connects two vertices in the conflict graph if the corresponding links in the original graph cannot be successfully activated in the same time slot. Coloring the vertices of the conflict graph so that each vertex has a unique color among its adjacent vertices yields a collision-free slot assignment. An equivalent approach is the use of *interference links* in the original graph; an interference link (e.g. the dotted line in Figure 3) is added between two terminals i and j if $(i, j) \notin E$, but activation of either i or j prevents packet reception at the other terminal. As a result, the interference links function only as additional scheduling constraints. This approach is used in [8], [18], and [44] to develop link

schedules. However, the addition of interference links does not fully solve the problem since it still does not account for link failures caused by the aggregate interference from many terminals transmitting simultaneously.

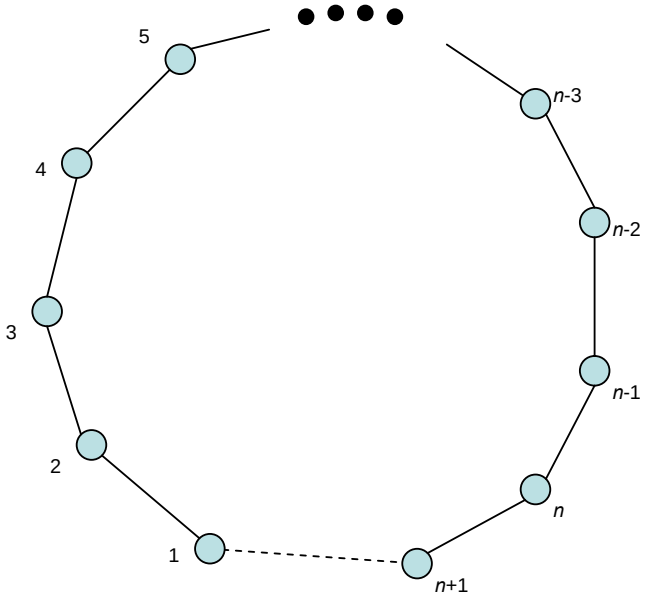


Figure 3. Example network in which terminals 1 and *n*+1 are *n* hops apart in the topology graph, yet may interfere strongly with one another.

2.3 Other MAC Protocols Using the Physical Interference Model

A common theme in the design of the above transmission scheduling protocols is that they first identify available network links, and then develop a schedule that meets a set of criteria, such as collision-free broadcasts or minimum end-to-end packet delay. However, there is a direct and complex interaction among transmission schedules, the MAI environment, and the links present for communication. The set of usable links varies from one slot to the next, depending upon the set of terminals transmitting in each slot. This results in the poor performance of the graph based schedules when used in the physical interference model, as noted in Section 1.3 ([8], [9], [10], and [11]).

The problem has also been observed in IEEE 802.16 mesh networks. The 802.16 Coordinated Distributed Scheduling (CDS) protocol partially addresses the MAI problem by utilizing an extended 3-hop transmission scheduling mode (via *ExtendedNeighborType*) when necessary. While it may eliminate some collisions, this strategy results in increased overhead and decreased spatial reuse. In [10], it is shown that even the extension to a 3-hop mode does not ensure collision-free operation under more realistic channel models. Their proposed modification to CDS, termed *collision-free* CDS or CF-CDS, allows terminals to monitor and detect collisions, and adapt the schedule when necessary.

In a few recent papers, MAC protocols have been explicitly designed based on the physical interference model. In [45] the authors develop distributed link scheduling called Randomized Contention Aware Multiple Access (RCAMA) which converges asymptotically over time to throughput optimality. RCAMA requires a total of eight transmissions per slot: 3 RTS/CTS exchanges with different transmitter sets to obtain knowledge of the interference environment, followed by a DATA/ACK exchange. In addition, optimality is only guaranteed if the physical environment is static. Lastly, efficient distributed implementation of RCAMA requires the path-loss exponent between each terminal to be bounded, and thermal noise must be bounded in terms of the interference. In [46], a contention-based MAC protocol uses DSSS modulation to enable clusters of terminals to transmit at the same time. While the clustering of transmitters does improve throughput in a DSSS system, the clustering requires additional overhead in the form of two RTS/CTS/RTS exchanges, followed by a DATA/ACK exchange, for a

total of eight transmissions per slot. Also, as mentioned in the introduction, there are numerous applications for which scheduled access is preferable to contention-based access.

CHAPTER THREE

EVALUATION OF DISTRIBUTED SCHEDULING ALGORITHMS

In Section 2.2, several distributed, graph-based scheduling algorithms are mentioned. These algorithms are designed to provide collision-free schedules in the graph-based channel model given a fixed set of known neighboring terminals. In contrast, the major contribution of this work is a protocol which develops the neighbor set of each terminal based upon information received from nearby terminals in the physical interference model. Efficient performance of this protocol requires a method of assigning transmission slots based upon the neighbor set. Several of the algorithms described in Section 2.2 can satisfy this requirement. In this chapter, we present a detailed study of three such algorithms to determine which approach most efficiently allocates channel resources to terminals.

3.1 Overview of Distributed Scheduling Algorithms

We define the *neighborhood* of a terminal *i*, denoted N_i, to be the set of local terminals which influence scheduling decisions at *i*, inclusive of *i* itself. The key property of this set is that whenever *i* transmits, all terminals in N_i / i are in receive mode, and when one or more terminals in N_i / i transmit, terminal *i* is in receive mode. We examine the performance of three priority-based broadcast scheduling algorithms in terms of the neighborhood size, $|\mathrm{N}_i|$, allowing evaluation of how well channel resources are provisioned under each algorithm. Performance differences between these algorithms

arise from how they handle problems of *priority chaining* [47] and *priority starvation*, and by how they classify contenders for channel access.

Denote the priority of terminal *i* to transmit in slot *t* as $P(i,t)$. *Priority chaining* occurs when some terminal is preempted in the schedule by a terminal with higher priority, which is itself preempted by a third terminal with even higher priority. Consider the example network in Figure 4 under the rules of a collision-free broadcast schedule (rules i-b and ii-b in chapter 2). Terminals *i* and *j* cannot transmit in the same slot because they are two hops apart; similarly, terminals *j* and *k* cannot transmit in the same slot. If, in the current slot *t*, $P(i,t) < P(j,t) < P(k,t)$, then *i* does not transmit in slot *t* because it is preempted by *j*. However, *i* could transmit in slot *t* since *j* is itself preempted by *k*.

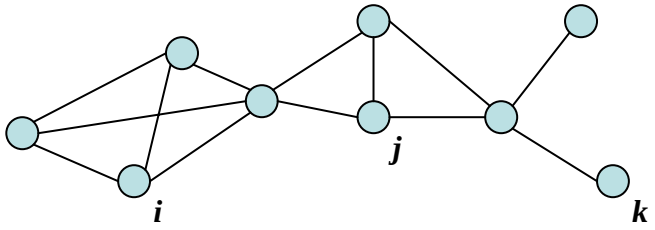


Figure 4. Example network used to illustrate priority chaining.

Priority chaining leads to inefficient spatial allocation of the channel. One strategy for decreasing the rate of priority chaining is to allow only a subset of terminals to be eligible to transmit in each slot. The ineligible terminals would then have a priority of 0. However, this leads to another problem termed *priority starvation*, which occurs when terminal *i* and all of its neighbors, N_i, are assigned priority 0. As an example, in

Figure 5 we define set $\mathrm{N}_i = \{i, j, k_1, k_2, k_3\}$. In this example, priority starvation occurs if $P(i,t) = P(j,t) = P(k_1,t) = P(k_2,t) = P(k_3,t) = 0$.

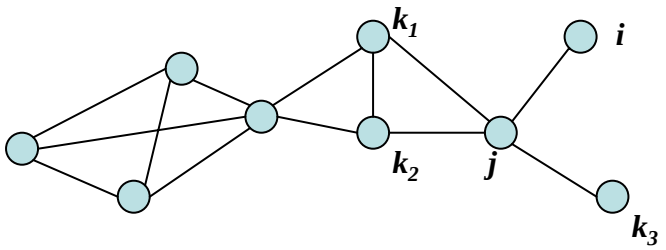


Figure 5. Example network used to illustrate priority starvation.

In the following sections, we describe a scheduling algorithm which is affected by priority chaining only, another algorithm which is affected by priority starvation only, and a third algorithm which is affected by both priority chaining and priority starvation, but to a smaller extent.

3.2 NAMA

NAMA is described in [43]. NAMA uses neighborhood-aware contention resolution (NCR), a hash-based priority assignment algorithm which operates as follows: in each slot *t* terminal *i* calculates its priority as

$$P(i,t) = MD(t \oplus i) \oplus i \,. \tag{3.1}$$

The function MD(*x*) is a deterministic hashing function designed to generate a uniformly distributed pseudorandom number based upon bit-wise hashing of *x*; '$\oplus$' acts as the bit-wise concatenation operator. In this work, we use a deterministic hashing function derived from the MD5 message digest algorithm [48]. In each slot, terminal *i* generates the priority for itself and each neighbor in the set N_i. If, in slot *t*, terminal *i* generates the largest priority among its neighbors, *i* transmits in slot *t*. Since the terminal

ID is bit-wise appended to the priority derived by the hash function, each terminal generates a unique priority in each time slot. Also, each terminal has a non-zero priority to transmit in each slot, hence priority starvation does not occur because some terminal has priority to transmit in each neighborhood.

3.3 UxDMA

UxDMA [13], used in the context of TDMA, assigns transmission slots in a repeating transmission frame based on color numbers assigned to a terminal and other terminals in its neighborhood. UxDMA requires each terminal i to have a unique color number among its neighbors, N_i, to avoid collisions. UxDMA is not a completely distributed algorithm as it requires global knowledge of the largest color number assigned, c_{max}, which in turn defines the length of a transmission frame. However, a distributed implementation of UxDMA, called DRAND [41], employs a lottery process to assign colors to terminals up to c_{max}.

Under UxDMA, the transmission priority of a terminal i with color c_i in slot t is

$$P(i,t) = \begin{cases} 1, & t \bmod c_{\max} = c_i \bmod c_{\max} \\ 0, & otherwise \end{cases}. \qquad (3.2)$$

Hence, in each transmission frame, terminals are only candidates to transmit in the slot corresponding to their color number. Since each terminal has a unique color amongst its neighbors, if a terminal is a candidate to transmit then it is the only candidate in its neighborhood. Thus, priority chaining cannot occur. However, priority starvation does occur because terminals calculate the frame length based upon the largest color

number in the network. For example, any neighborhood which does not contain color c_{max} experiences starvation in the last transmission slot of every frame.

3.4 Lyui's Algorithm

Lyui's scheduling algorithm was first described in [33], and a brief description of the algorithm and it properties is given in [34]. Lyui's algorithm also assigns transmission slots in a repeating frame based on color numbers, and also requires each terminal i to have a unique color number among its neighbors in N_i. For an arbitrary color number c, let p(c) represent the smallest power of 2 greater than or equal to c. Let $c_{i,max}$ represent the largest color number found in set N_i. The frame size of terminal i is p($c_{i,max}$). Frame lengths may vary across the network depending on local terminal density and the resulting color assignment; however, since the frames are all a power of two, frames of differing lengths nest together evenly.

Under Lyui's algorithm, the transmission priority of terminal i with color number c_i in slot t is

$$P(i,t) = \begin{cases} c_i, & t \bmod p(c_i) = c_i \bmod p(c_i) \\ 0, & otherwise \end{cases}. \tag{3.3}$$

In Table 1, this rule is used to indicate the slots in which the first eight color numbers have nonzero priority. If a terminal has the highest candidate color number in its neighborhood, then it has priority to transmit. Terminals are guaranteed to transmit at least once in each frame, and possibly in additional slots depending on the coloring of neighbors. For example, if a terminal is assigned color 2 but has no neighbors with color 4, then that terminal may also transmit in the fourth slot of each frame.

Priority chaining is possible under Lyui's algorithm; for example, consider Figure 4 with t=8, c_i = 2, c_j =4, and c_k =8. In slot 8, color number 8 has the highest transmission priority; color number 4 has the next highest transmission priority, followed by color numbers 2 and 1, respectively. Terminal i does not transmit in slot 8 because terminal j is a candidate to transmit and has higher priority, and terminal j does not transmit because terminal k is a candidate to transmit and has higher priority than terminal j. In this case, terminal i could transmit without generating a collision.

Priority starvation is also possible using Lyui's algorithm; for example, consider Figure 5 with t=2 and terminals $\{i, j, k_1, k_2, k_3\}$ assigned colors {3,4,5,6,7}, respectively. In slot 2, terminals with color number 2 have the highest priority to transmit, and terminals with color number 1 may transmit if there is no neighbor with color number 2. Terminal i does not transmit in slot 2 because color number 3 is not a candidate to transmit in slot 2. No neighbors of terminal i transmit in slot 2 because no neighbors have color number 1 or color number 2. In this case, terminal i could transmit without generating a collision.

		Candidate Transmission Slot Numbers															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Color Number	1	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
	2		×		×		×		×		×		×		×		×
	3			×				×				×				×	
	4				×				×				×				×
	5					×								×			
	6						×								×		
	7							×								×	
	8								×								×

Table 1. Colors and slots in which they are candidates.

3.5 Performance Evaluation

We use simulations to evaluate NAMA, UxDMA, and Lyui's algorithm based upon the average number of assigned transmissions per terminal per slot. We also evaluate the channel access delay of each approach by computing the average number of slots between successive transmissions for each terminal, as well as the maximum number of slots between successive transmissions for all terminals in the network. Results are averaged over a set of 100 networks, each containing 200 terminals placed at random locations in a square of area 1,414 m^2. Statistics for each test network are obtained over a period of 1024 slots. To eliminate edge effects and produce a constant average terminal density, opposite edges of each test network are stitched together to form a torus. Letting *w* and *h* represent the width and height of the area, the distance between terminals *i* and *j*, located at positions (x_i, y_i) and (x_j, y_j), respectively, is given by

$$d(i,j) = \sqrt{\left(\frac{w}{2} - \left|\frac{w}{2} - \left|x_i - x_j\right|\right|\right)^2 + \left(\frac{h}{2} - \left|\frac{h}{2} - \left|y_i - y_j\right|\right|\right)^2} \,. \tag{3.4}$$

For these simulations, network connectivity is modeled using a graph, similar to the approach described in Section 2.2. The *communications range*, *R*, is used to define the neighborhood size for each terminal in the following manner: all terminals *j* for which $d(i,j) \le R$ are in set N_i^1, and the neighborhood of terminal *i* is $\mathrm{N}_i = i \bigcup \mathrm{N}_i^1 \bigcup \left(\bigcup_{j \in \mathrm{N}_i^1} \mathrm{N}_j^1 \right)$. Thus, the neighborhood of each terminal is nondecreasing with *R*.

An interesting point of reference is the performance of a centralized algorithm designed to maximize assigned transmissions per terminal per slot and minimize delay

between transmissions. This has been a topic of some interest, resulting in several centralized optimization algorithms (see Section 2.1). We have developed a centralized algorithm which augments the slot assignment of each distributed algorithm to achieve a Pareto optimal transmitter configuration in each slot. In particular, after transmission slots are assigned using a specified algorithm (NAMA, UxDMA, or Lyui), the list of terminals is traversed to pack additional transmissions into the schedule. As each terminal is visited, if the terminal is in receive mode and all terminals in its neighborhood are in receive mode, then the terminal is switched to be a transmitter. As a result, priority chaining and priority starvation are eliminated.

Figure 6 shows the number of transmissions per terminal per slot, averaged over all test networks, as a function of the neighborhood size used by the terminals. As the neighborhood size increases, terminals must share the channel with more neighbors, and thus gain access to the channel less often. The centralized packing algorithm results in approximately equal performance when used with all three scheduling approaches. Lyui's algorithm provides the highest level of spatial reuse for all neighborhood sizes, while the performance of NAMA is noticeably lower due to priority chaining. UxDMA performs better than NAMA when the average neighborhood size is large; however, if the average neighborhood size is small, NAMA performs better than UxDMA. This is because the transmission frame length used by UxDMA is dictated by the largest neighborhood in the network; as a result, there are many smaller groups of terminals which are forced to use an unnecessarily long transmission frame, resulting in priority starvation since they are not eligible to transmit in later slots of the frame.

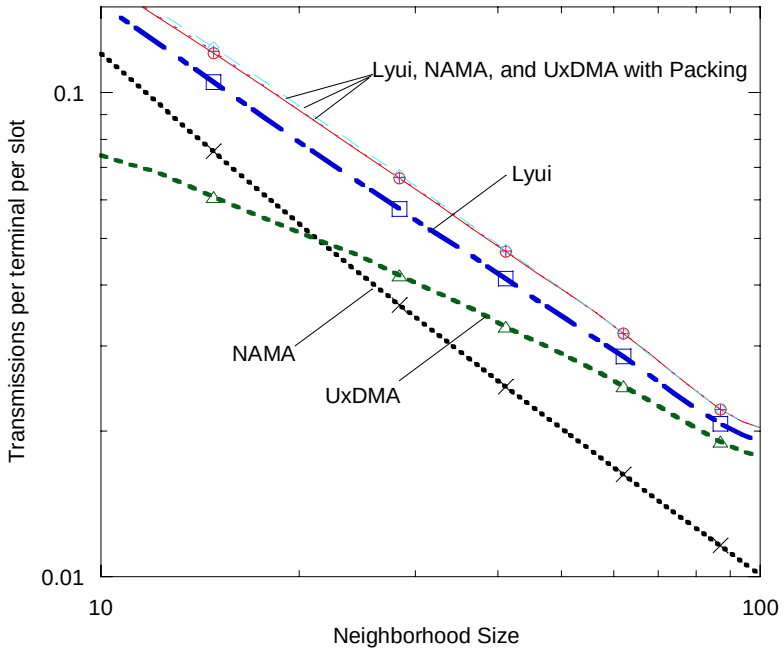


Figure 6. Log-log plot of spatial reuse as function of neighborhood size for NAMA, UxDMA, and Lyui's algorithm.

The average delay between transmissions using each algorithm is shown in Figure 7. The average delay is inversely related to the transmissions per terminal per slot: if more terminals transmit in each slot, then it is natural that fewer slots elapse between each transmission by a terminal. However, by examining the maximum delay between transmissions, one may observe a large difference. In Figure 8, the average maximum delay between transmissions is calculated by finding the maximum time between transmissions for any terminal in each test network, and then averaging over all test networks. Under NAMA, terminals may go an extended period of time without winning the right to transmit in a slot. Thus, there is a significant probability that in a network of 200 terminals, some terminal gets far fewer transmission opportunities purely by chance. UxDMA and Lyui's algorithm, on the other hand, both guarantee that terminals may

transmit once per frame, resulting in much smaller maximum delay values. For UxDMA, the average maximum delay between transmissions is equal to the average delay in Figure 7; this is because all terminals transmit exactly once per frame. Under Lyui's algorithm, terminals use a frame size that is a power of 2. This results in longer transmission frames and greater maximum delay between transmissions, yet the average delay in Figure 7 is lower because some terminals transmit in more than one slot per frame.

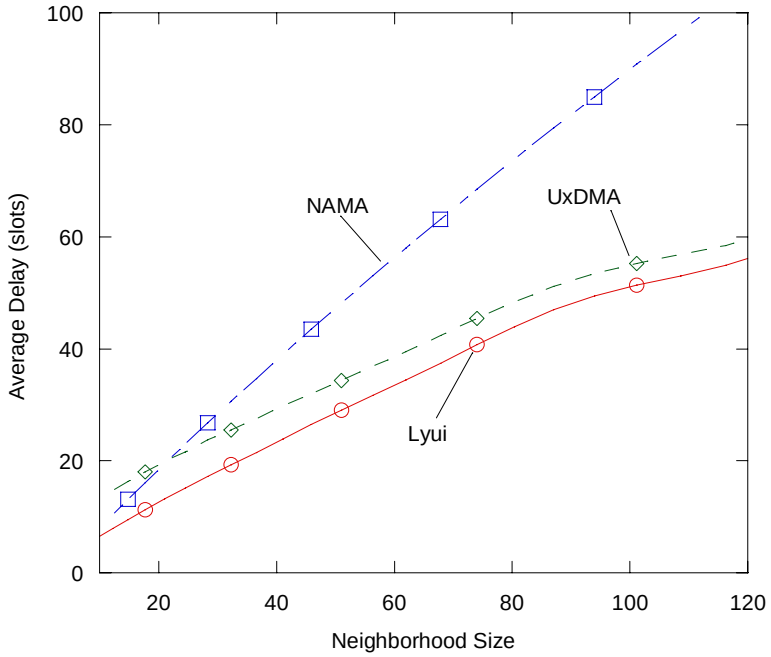


Figure 7. Average channel access delay for each scheduling algorithm.

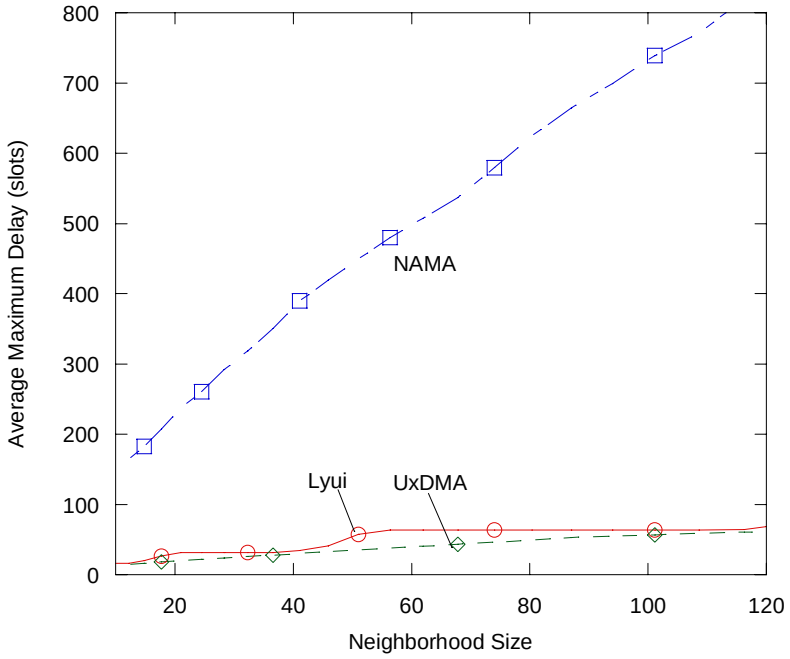


Figure 8. Average maximum channel access delay for each scheduling algorithm.

In designing a scheduling algorithm, one intuitive objective is to have each terminal share the channel fairly with all terminals in its neighborhood. Thus, if the neighborhood size of terminal *i* is $|N_i|$, then a fair allocation results in at least $\frac{1}{|N_i|}$ transmissions per slot for terminal *i*. Using this approach, we define the *spatial reuse efficiency* for a scheduling algorithm as the number of transmissions per terminal per slot per neighbor. In Figure 9, the spatial reuse efficiency is shown for each scheduling approach. Due to the complex manner in which priority chaining, priority starvation, and neighborhood membership are affected by *R*, the spatial reuse efficiency varies as neighborhood size increases. Under NAMA, the spatial reuse efficiency is approximately 1 for sufficiently large densities. Under Lyui's algorithm and UxDMA,

the spatial reuse efficiency is higher (between 1.5 and 2.1 for Lyui's algorithm) because multiple neighbors of a terminal may not be neighbors themselves. For example, if terminal *i* has neighbors $j_1 \in \mathrm{N}_i^1$ and $j_2 \in \bigcup_{j \in \mathrm{N}_i^1} \mathrm{N}_j^1$, it may be the case that $\mathrm{N}_{j_1}^1 \bigcap \mathrm{N}_{j_2}^1 = \varnothing$. In this case, j_1 and j_2 may use the same color and transmit at the same time.

To better illustrate this point, in Figure 10 we show the average neighborhood size as *R* increases. When *R* is approximately 300m, the average neighborhood size is 100 but the average maximum color number found in the neighborhood of a terminal is only about 50. Terminals using Lyui's algorithm and UxDMA use color numbers to contend for channel access. Terminals using NAMA, on the other hand, contend using their ID numbers, which are all distinct. Due to the overlap of neighbor colors, terminals observe a lower number of contenders under Lyui's algorithm and UxDMA than under NAMA. Hence, they operate as if they have a smaller neighborhood and enjoy higher spatial reuse efficiency.

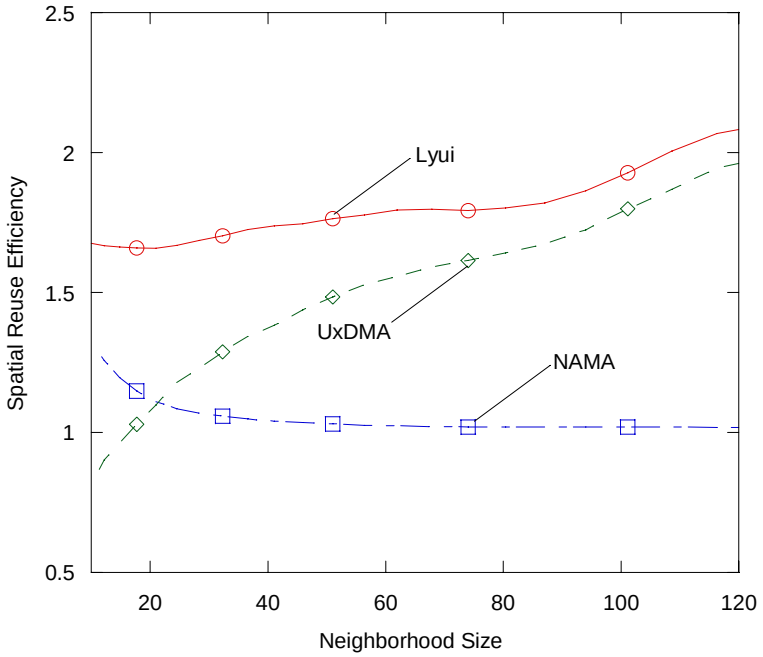


Figure 9. Spatial reuse efficiency metric for each scheduling algorithm.

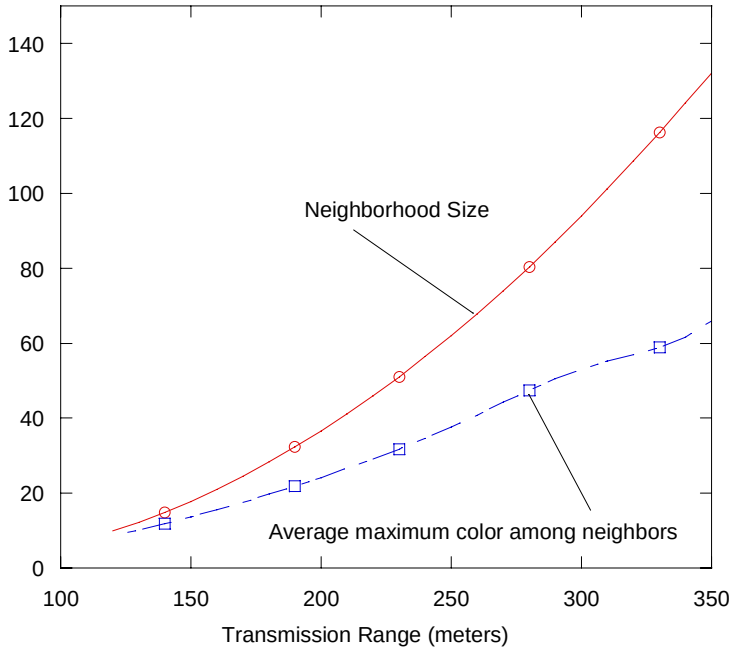


Figure 10. Neighborhood size and average number of colors required as R increases.

There are clear advantages to using color numbers instead of terminal ID numbers when scheduling transmissions. Since terminals using a color-based transmission scheduling algorithm tend to have multiple neighbors with identical colors, the number of contenders in each slot is reduced in comparison to a system in which each neighbor generates a distinct priority, such as NAMA. As a result, Lyui's algorithm provides between 1.5 to 2 times as much spatial reuse as NAMA for a given neighborhood size. A smaller number of contenders results in a greater rate of spatial reuse and a lower rate of priority chaining, but introduces the problem of priority starvation since only certain colors are eligible in each slot. Lyui's algorithm, which allows multiple colors to be candidates for transmission in each time slot, performs better than UxDMA, which only allows one color to be a candidate in a time slot. In addition, Lyui's algorithm does not require the frame size to be a global parameter, as it is in UxDMA, making it well-suited for implementation in a distributed network.

CHAPTER FOUR

SYSTEM DESIGN AND EVALUATION

In this chapter we describe in detail the wireless channel and receiver model used throughout the remainder of this work. We also provide an overview of network traffic generation, multi-hop routing, and the method for computation of results used throughout the rest of this work; this organization allows for an efficient discussion of protocol features and performance in later chapters.

4.1 Channel and Receiver Model

We assume terminals are synchronized to slot boundaries using an external GPS signal, as described in [39], or they may establish local synchronization in a distributed manner similar to [49]. Also of note is the distributed protocol in [50], which is designed to allow groups of terminals with independent local synchronization to agree on a common slot reference. All terminals use identical transmission power, except in cases noted in Section 8.4 where terminals may use a reduced transmission power for certain transmissions to influence the network topology.

Terminals communicate over a common channel using DSSS modulation with a fixed chip rate. Link gain is symmetric between two terminals, and constant for the duration of a transmission slot. A transmission is successfully received only if the SINR at the receiver exceeds a threshold, β. Specifically, when a link (i,j) is activated for a transmission from terminal i to terminal j, the SINR for the link, denoted $\xi_{i,j}$, must satisfy

$$\xi_{i,j} = \frac{P_r(i) N_{i,j} T_c}{N_0 + \sum_{\forall k \neq i} P_r(k) T_c} > \beta . \tag{4.1}$$

In the above calculation, $P_r(i)$ is the power received from terminal i at j, T_c is the chip duration, $N_{i,j}$ is the spreading factor employed for this transmission, and N_0 is the receiver noise power. This model for DSSS modulation assumes spreading codes are pseudo-noise (PN) sequences and the sequences are long enough so that there are a large number of possible sequences. We assume spreading sequences are pre-assigned to terminals and automatically provided to all other terminals.

To simulate the capture effect, we consider two distinct cases: transmissions using a common spreading code known to all terminals, and transmissions using a transmitter-oriented spreading code with a unique code for each transmitter (see [51]). For common-code transmissions, if several terminals begin transmitting at the start of a time slot, we assume the slight clock differences at the transmitters and variations in propagation times cause the signals arriving at receiver j to be chip-asynchronous, enabling capture of a single transmission.

We model capture as follows. In a time slot in which all transmitters employ a common spreading code, each receiver j calculates the SINR for each transmitter i and forms the set $S_j \equiv \left\{ i \middle| \xi_{i,j} > \beta \right\}$ of candidate signals to capture. If S_j is not empty, j randomly selects one element from the set using a uniform distribution. This capture model reflects the message-retraining capture model described in [52] with retraining threshold $\gamma_{MR} \cong \beta$. For time slots in which all transmitters employ transmitter-oriented

spreading codes, a receiver must pre-select a transmitter-oriented code to monitor and it receives a transmission only if this transmitter is active and (4.1) is satisfied.

Terminals in receive mode are supplied with sample SINR estimates for each successful transmission using a common or transmitter-oriented code. The sample SINR estimates are made relative to a maximum spreading factor, $N_{\max}$. Thus if a transmission from j to i uses spreading factor $N_{j,i}<N_{\max}$, then the sample SINR estimate from the receiver is multiplied by $\frac{N_{\max}}{N_{j,i}}$. The incoming link SINR estimate, denoted $\hat{\xi}_{j,i}$, is computed by terminal i as the minimum of the last ten sample SINR estimates for transmissions from j.

We assume terminals are able to detect certain instances in which a transmission fails due to insufficient SINR, although a reliable SINR estimate is not provided. Specifically, if a receiver has selected a transmitter-oriented code to monitor, but a transmission using that code fails due to insufficient SINR, then receiver forms a sample SINR estimate of β for that transmission. If a receiver has selected a transmitter-oriented code to monitor, but no transmission is made, then no sample SINR estimate is provided to the receiver.

4.2 Simulation Settings

The channel parameters used in simulations are listed in Table 2. The power received from transmitter i at receiver j is

$$P_r(i) = P_t G_{i,j} = P_t \left(\frac{\lambda}{4\pi d_{i,j}} \right)^{\alpha} . \tag{4.2}$$

This depends on the transmitted energy P_t, the wavelength λ of the carrier frequency, the transmission distance $d_{i,j}$, and the path-loss exponent α.

Parameter	Value
β	8
T_c	2.9e-7 s/chip
N_0	4.0e-21 J/Hz
N_{max}	32, 64, or 96
λ	0.125m
α	3.5

Table 2. Channel parameters used in simulations.

In the simulations, the *communications range* of a terminal, denoted *R*, is the maximum distance between a transmitter and a receiver so that (4.1) holds, assuming no MAI and use of the largest spreading factor, $N_{\max}$. Given *R*, we set the transmission power, P_t, as

$$P_t = \left(\frac{4\pi R}{\lambda}\right)^{\alpha} \frac{\beta N_0}{T_c N_{\max}}. \qquad (4.3)$$

4.3 Packet Generation and Forwarding

The network packet generation rate, γ, is equal to the expected number of unicast data packets generated by the network in each slot. In each slot, each terminal generates a unicast data packet with probability γ/N, where *N* is the total number of network terminals. At the time of generation, the packet's destination is chosen uniformly from

the set of remaining terminals; for networks with N terminals, there are $N(N-1)$ distinct flows.

Packets are not acknowledged, nor are dropped packets retransmitted. Packets are dropped in four situations. A terminal may enqueue up to 20 packets (data and control); a packet arriving to a full queue is dropped. If a packet arrives at a terminal which has no route to the packet's destination, the packet is dropped. A packet is dropped if the SINR is less than or equal to β at the receiver. A packet is also dropped if the intended receiver is not correlating to the transmitter's spreading sequence.

4.4 Routing of Multi-hop Traffic

Routing protocols significantly affect system performance, and are themselves influenced by the network topology and transmission schedule. Distributed routing protocols may require significant overhead in an ad hoc network. However, this work focuses on the performance of a cross-layer scheduling protocol. As a result, we use a centralized, min-cost routing algorithm to compute the forwarding tables used by terminals. Each terminal computes a link cost to each of its neighbors, and these link costs are used in Dijkstra's algorithm to compute min-cost routes between each pair of terminals. At the end of the process, the forwarding table for each terminal is updated automatically. In simulations, the min-cost routes are recomputed every 64 time slots based upon the current link costs stored by each terminal.

The cost metric computed by terminal i for a neighboring terminal, j, is

$$Cost_{i,j} = \frac{\zeta(\hat{\xi}_{i,j})\left(1+U_j\right)}{ETR(i)\times LinkRate(i,j)}. \tag{4.4}$$

In (4.4), *ETR*(*i*) represents the *effective transmission rate* of terminal *i*, calculated as the number of slots in each frame in which *i* transmits divided by the frame length of terminal *i*. *LinkRate*(*i*,*j*) is the number of packets per slot which may be sent over this link, and this depends upon the outgoing link SINR estimate $\hat{\xi}_{i,j}$. The factor U_j is a number between 0 and 1 which represents the assigned slot utilization of terminal *j*, measured as the fraction of transmission slots assigned to *j* in which *j* transmits a packet (data or control). Whenever terminal *j* is a candidate to transmit in a given slot, *j* updates its utilization estimate as follows:

$$U_j' = (0.95)U_j + (0.05)T(j) \tag{4.5}$$

The function $T(j)$ is an indicator function which is equal to 1 if *j* transmits a packet in the current slot and 0 otherwise. This is an exponentially weighted moving average with a smoothing factor of 5%. The utilization estimate results in a higher link cost for neighbors with large traffic loads, while neighbors with low traffic loads are assigned lower link costs to avoid traffic congestion. Links with low SINR estimates are more sensitive to MAI, so a scaling function, $\zeta(x)$, is used to deemphasize routing over low-SINR links. $\zeta(x)$ is defined as

$$\zeta(x) = \begin{cases} \infty, x \le \beta \\ 1 - \ln\left(\dfrac{x-\beta}{\beta}\right), \beta < x \le 2\beta \\ 1, x > 2\beta \end{cases}. \tag{4.6}$$

Several other routing metrics were examined in preliminary work. These included min-hop routing, and several variations on (4.4) which did not take into account utilization, link rate, or effective transmission rate. We determined that a simple min-hop routing metric performs poorly in this environment when compared to metrics that account for link rate and link SINR. We also determined that including utilization in the routing metric leads to significantly better overall performance. The metric in (4.4) may be implemented by periodically requiring terminals to exchange utilization estimates.

4.5 Generation of Network Performance Statistics

In steady-state simulations, packet statistics are collected after a warm-up period of 3000 slots to allow queue lengths to reach their stationary distributions. This is followed by a period of 1000 slots, during which time marked packets are generated. The simulations end when all marked packets are accounted for. End-to-end packet statistics reported represent the average values over all test networks. Formally, we may define the following:

- $t(i,j,k)$: total number of marked packets successfully received for flow (j,k) in network instance i
- $d(i,j,k)$: sum of packet delay for all marked packets successfully received for flow (j,k) in network instance i
- $g(i,j,k)$: total number of marked packets generated for flow (j,k) in network instance i
- Υ : the set of test networks
- N: the number of terminals in each test network

- Δ: the duration for which marked packets are created, measured in slots

The following calculations are used to measure packet throughput, delay, and completion rate in simulations:

$$Throughput = \frac{1}{|\Upsilon|}\sum_{i=1}^{|\Upsilon|}\frac{1}{N(N-1)}\sum_{j=1}^{N}\sum_{\substack{k=1\\k\neq j}}^{N}\frac{t(i,j,k)}{\Delta}, \tag{4.7a}$$

$$Delay = \frac{1}{|\Upsilon|}\sum_{i=1}^{|\Upsilon|}\frac{1}{N(N-1)}\sum_{j=1}^{N}\sum_{\substack{k=1\\k\neq j}}^{N}\frac{d(i,j,k)}{t(i,j,k)}, \tag{4.7b}$$

and

$$Completion\ \ Rate = \frac{1}{|\Upsilon|}\sum_{i=1}^{|\Upsilon|}\frac{1}{N(N-1)}\sum_{j=1}^{N}\sum_{\substack{k=1\\k\neq j}}^{N}\frac{t(i,j,k)}{g(i,j,k)}. \tag{4.7c}$$

CHAPTER V

IMMEDIATE NEIGHBOR SCHEDULING

The multiple-access capability of DSSS modulation motivates a different design philosophy for transmission scheduling protocols since transmissions can be received in the presence of interference from other users. In particular, terminals may use a smaller neighborhood to schedule transmissions, resulting in greater spatial reuse and less control overhead. Suppose a network of nine terminals is constructed so that there are three clusters of three terminals apiece, arranged as shown in Figure 11. Terminals in cluster 1 can communicate directly with terminals in cluster 2, and terminals in cluster 2 can communicate directly with terminals in cluster 3, but terminals in clusters 1 and 3 cannot communicate directly. Two scheduling scenarios are examined: in the first scenario, the neighborhood of a terminal used for scheduling transmissions consists of all terminals within two hops. In the second scenario, the neighborhood of a terminal consists of all terminals within one hop. Each terminal is required to have a unique color in its neighborhood, so the first scenario requires 9 total colors. In the second scenario the two non-neighboring clusters reuse colors, requiring 6 total colors.

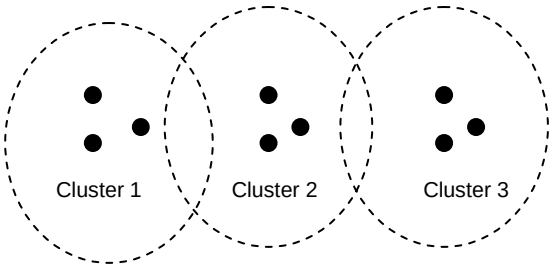


Figure 11. Example network with nine terminals arranged into three clusters.

In scenario 1 only one terminal may transmit per slot, allowing each terminal in cluster 2 to establish links with each terminal in clusters 1 and 3. In scenario 2, when two terminals in clusters 1 and 3 which are assigned the same color transmit simultaneously, the terminals in cluster 2 can capture only one of the transmissions, even if both may satisfy the SINR requirement in (3.1). As a result, each terminal in cluster 2 can establish up to three bidirectional inter-cluster links. Figure 12 shows two simulated topologies corresponding to the two scenarios. Despite the loss of some communications links, connectivity between clusters is maintained in scenario 2 due to the random manner in which capture occurs. The advantage to this is that instead of transmitting in 1 out of 9 slots on average, as in scenario 1, terminals are able to transmit in 1 out of 6 slots on average – a gain of 50%. At the same time, the diameter of the network increases from 2 to 3; however, only a fraction of the network traffic is affected by this increase. Also, in a mobile ad hoc network, the schedule changes whenever the neighborhood membership changes. Since scenario 2 uses a smaller neighborhood for scheduling, the schedule changes less frequently in mobile scenarios, and less information must be exchanged to adapt the schedule.

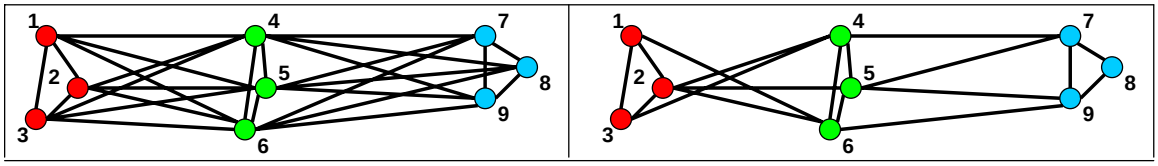


Figure 12. Network topology for scenario 1 (left), where the neighborhood is all terminals within two hops, and network topology for scenario 2 (right), where the neighborhood is all terminals within one hop.

The benefit of using a smaller neighborhood to schedule transmissions in conjunction with DSSS modulation motivates the design of our protocol. Since the

protocol collects information only from terminals which are near at hand to schedule transmissions, we denote this approach as the *immediate neighbor scheduling* (INS) protocol. The protocol defines how terminals collect, exchange, and use neighbor information, which is then used by Lyui's algorithm (described in Section 3.4) to schedule transmissions.

5.1 Summary of INS Properties

The key properties of the INS protocol are summarized as follows:

- Transmission slots are divided into an identification interval and a data interval (see Figure 13). The neighborhood used by Lyui's algorithm for scheduling transmissions is based upon neighbors detected during identification intervals.
- Terminals maintain basic information about neighbors using a *neighbor table*.
- Terminals in receive mode use a *receive vector* to determine which neighbor transmits in each slot. The receive vector for terminal *i* is denoted $\mathbf{r}^{(i)}$. If element r_s^i of $\mathbf{r}^{(i)}$ is equal to *j*, then *i* attempts to receive a packet from neighbor *j* in slot *s* of the frame.
- Terminals in transmit mode use a *transmit matrix* to determine which neighbors may receive a transmission in the current slot. The transmit matrix for terminal *i* is denoted $\mathbf{T}^{(i)}$. Element $t_{j,s}^i$ of $\mathbf{T}^{(i)}$ is equal to 1 if neighbor *j* may receive a transmission from *i* in slot *s*, and 0 other wise.

- Entries in the neighbor table are formed by receiving a FLAG packet in the identification interval. Entries in $\mathbf{r}^{(i)}$ are formed by receiving a packet in the data interval. Entries in $\mathbf{T}^{(i)}$ are formed via exchange of periodic *neighbor acknowledgement*, or NBR_ACK, control packets.
- Neighbors are grouped into two sets. The *communicable neighbor* set of *i*, N_1^i, is the set of nearby terminals for which bidirectional communication is possible. The *detectable neighbor* set, N_2^i, is the set of nearby terminals for which communications links are intermittent or unidirectional.
- Each terminal maintains a color number that is unique among the terminals in its neighborhood.

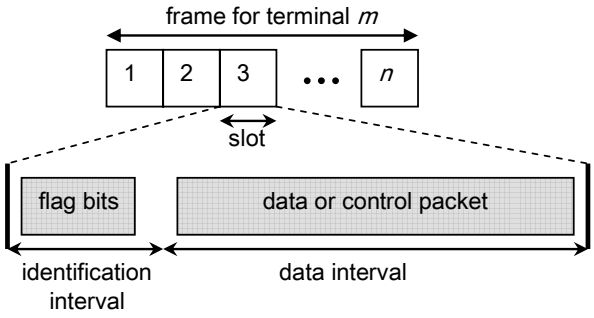


Figure 13. Formatting of transmission slots.

5.2 INS Description

Transmissions during the identification interval use a common spreading code to facilitate detection of neighboring terminals, while transmissions during the data interval use a transmitter-oriented spreading sequence unique to the transmitting terminal. When terminal *i* is assigned to transmit in slot *s*, it transmits a FLAG packet in the identification

interval with probability ½. A FLAG packet from *i* contains the ID and color number of *i*. Since fewer terminals transmit in the identification interval, multiple-access interference is much lower during this time and neighbors are more easily detected. If *i* does not transmit a FLAG in the identification interval, *i* may receive a FLAG from another transmitter, even if *i* transmits in the data interval.

Since transmissions during the data interval use a transmitter-oriented spreading sequence, the receive vector for terminal *i*, $\mathbf{r}^{(i)}$, allows *i* to determine which transmitter-oriented spreading code to monitor in each slot in which it does not transmit. The transmit matrix of *i*, $\mathbf{T}^{(i)}$, specifies, for each slot, which terminals monitor *i*'s unique spreading sequence. The neighbor table of *i* stores terminal ID, color number, link parameters (outgoing link SINR Estimate, transmission rate, and cost for use in routing), and expiration slots for the neighbor table entry, $\mathbf{r}^{(i)}$, and $\mathbf{T}^{(i)}$.

Entries in the neighbor table, $\mathbf{r}^{(i)}$, and $\mathbf{T}^{(i)}$ expire after a period of time if they are not refreshed so that the schedule remains efficient as neighbors cease operating or move away. The *neighbor timeout* parameter, N_{to}, determines the number of transmission frames to store information that is not refreshed. When *i* receives a FLAG from terminal *j* in slot *s*, the information for *j* in the neighbor table is updated and the neighbor table entry's expiration slot is set be $2n_iN_{to}$ slots from the current slot, where n_i is the frame length calculated by *i*. When *i* receives a packet from terminal *j* in the data interval of slot *s*, *i* sets $r_s^i = j$, and sets the receive vector expiration slot for entry r_s^i to be n_iN_{to} from the current slot. Using a longer timeout for neighbor table entries is intuitive since it

promotes greater stability in neighborhood membership in situations where link SINR degrades slowly due to increasing distance between a transmitter and receiver.

Entries in $\mathbf{T}^{(i)}$ are formed via periodic exchange of NBR_ACK packets. A NBR_ACK packet transmitted by terminal *j* contains the receive vector of *j*, $\mathbf{r}^{(j)}$, as well as the estimate of the incoming SINR, $\hat{\xi}_{i,j}$, for each terminal *i* listed in $\mathbf{r}^{(j)}$ and the current utilization estimate U_j (used for determining link costs in the routing algorithm). For each terminal *i* that receives this NBR_ACK packet, *i* sets $t^i_{j,s} = 1$ for all slots *s* such that $r^j_s = i$. It also sets the transmit vector expiration slot for entry $t^i_{j,s}$ to be $n_i N_{\text{to}}$ slots from the current slot. Lastly, *i* updates its neighbor table with the outgoing link SINR estimate to *j*. To ensure periodic broadcast of NBR_ACK packets, when terminal *j* transmits a NBR_ACK packet, *j* schedules another NBR_ACK packet to be generated in a slot uniformly distributed in the interval $\left(\frac{\left(n_j N_{to}\right)}{2}, \left(n_j N_{to}\right) \right)$ slots from the current slot.

NBR_ACK packets are also automatically generated when a packet is received in the data interval from a terminal that is not listed in the receive vector.

There are two ways in which a terminal may receive a packet in the data interval from a terminal not listed in the receive vector. During slot *s*, if terminal *i* is in receive mode and $r^i_s = 0$ (i.e., no transmitter is associated with the current transmission slot), then if *i* receives a FLAG from a terminal *j*, we assume *i* processes the information from *j*, and infers *j*'s spreading code so it can attempt to receive a packet from *j* in the data interval. If $r^i_s = 0$ and no FLAG is received, *i* attempts to receive a packet from the

terminal in its neighbor table with the highest transmission priority in the current slot (if multiple terminals satisfy this then a simple tie-breaker, such as lowest ID, is used). Successful packet reception from terminal *j* in the data interval of slot *s* allows *i* to set $r_s^i = j$.

Every terminal is required to have a unique color number among the terminals listed in its neighbor table. If after updating its neighbor table, a terminal has the same color number as one of its neighbors, it changes its color number to the smallest color number not found in the neighbor table. If, after updating neighbor information, the neighbor table information is inconsistent with the receive vector or transmit matrix, then the invalid entries in $\mathbf{r}^{(i)}$, and $\mathbf{T}^{(i)}$ are automatically corrected. This may happen if a neighbor *j* changes its color, for example, from 1 to 2. Neighbor *j* can no longer transmit in slot 1. If $r_1^i = j$, then *i* sets $r_1^i = 0$. If $r_2^i = 0$, *i* sets $r_2^i = j$; otherwise r_2^i does not change.

The neighbors of terminal *i* are divided into two sets: *detectable* neighbors, N_2^i, and *communicable* neighbors, N_1^i. For a neighbor *j*, if $r_s^i = j$ for some slot *s*, and $t_{j,l}^i = 1$ for some slot *l*, then $j \in N_1^i$; otherwise, *j* is a member of N_2^i. The reason for this separation is that, due to varying levels of MAI, not all links are bidirectional. Unidirectional links are not reliable since transmission of NBR_ACK packets fails in one direction. Consequently, the link cost to neighbors in N_2^i is set to infinity so they are not considered for routing multi-hop data traffic.

5.3 INS Example

Figure 14 shows an example network during an initialization phase with six terminals, **A**, **B**, **C**, **D**, **E**, and **F**, arranged in a linear configuration. Each row corresponds to a time slot, and the color, transmission, and reception status for each terminal is shown in each slot. Initially, each terminal is assigned color 1. There are two special considerations during network initialization. First, since a terminal with color 1 and no detected neighbors has priority to transmit in every slot, we force the terminals to transmit with probability ¼ until at least one neighboring terminal is detected. In addition, the minimum frame size for a terminal is set to 4 slots.

In slot 1 of the example, only terminals **B** and **D** elect to transmit. **B** transmits both a FLAG packet and a packet in the data interval, while **D** transmits only during the data interval. The FLAG packet from **B** causes **A** and **C** to add **B** as a neighbor. Since **B** has color 1, **A** and **C** both change their color to 2. **D**'s transmission is unsuccessful because **C** and **E** have not yet detected **D** and do not know to monitor **D**'s spreading code. In slot 2, **A** and **C** transmit because they now have color 2. **A** transmits in the data interval only, while **C** transmits in both intervals. **B** detects the FLAG from **C,** adds **C** as a neighbor and receives the subsequent transmission from **C** in the data interval; no color change is necessary. In slot 3, **E** transmits a FLAG that is detected by **D** and **F**. **D** changes its color to 3 since it now has detected neighbors with colors 1 and 2. **F** changes its color to 2 since it has only detected a neighbor with color 1.

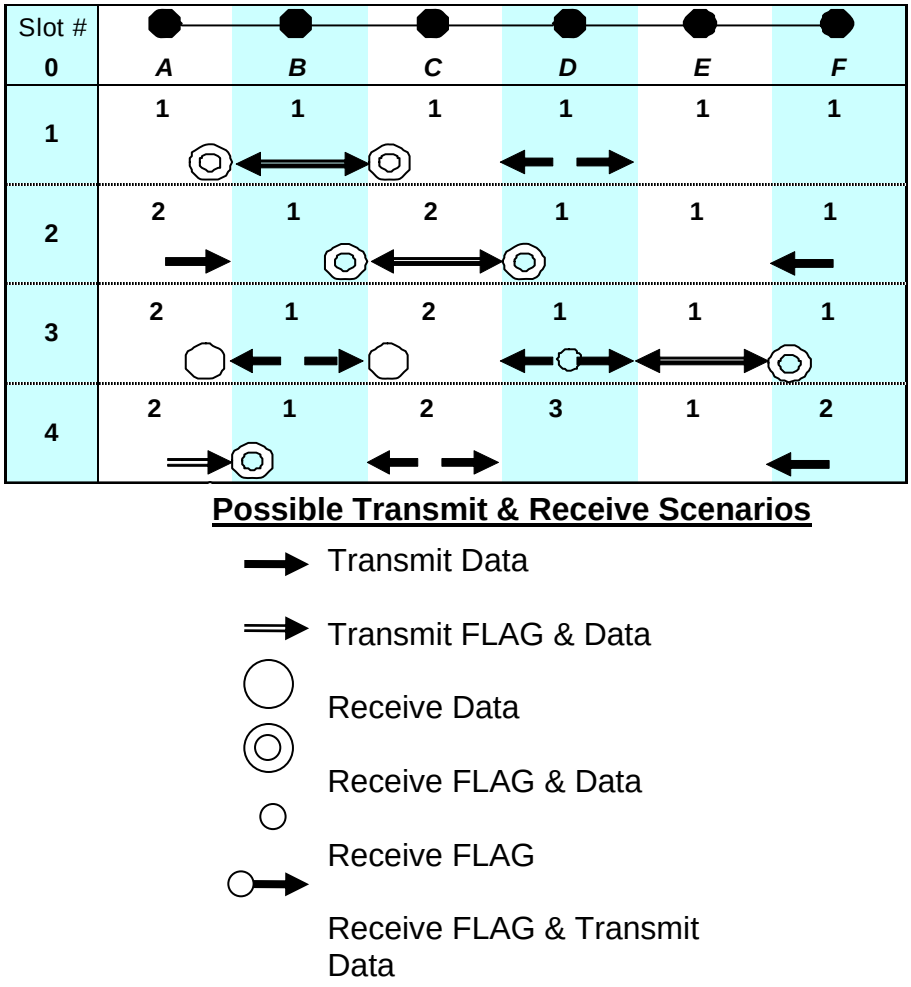


Figure 14. Example illustrating neighbor detection and color selection via FLAG reception.

We now examine the neighbor table of **B**, $\mathbf{r}^{(B)}$, and $\mathbf{T}^{(B)}$ at the end of slot 4. **B** received a packet from **C** in slot 2 and a packet from **A** in slot 4, so $\mathbf{r}^{(B)}$=(0,**C**,0,**A**). Assuming that the packets transmitted by **A** and **C** were NBR_ACK packets generated when they detected **B**, **C**'s NBR_ACK informs **B** that it can transmit a packet to **C** in slot 1 of the frame. **A**'s NBR_ACK informs **B** that it can transmit a packet to **A** in slots 1 and 3. Thus, $\mathbf{T}^{(B)}$ is:

$$\mathbf{T}^{(B)}=\begin{bmatrix}1 & 0 & 1 & 0\\ 0 & 0 & 0 & 0\\ 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}.$$

Lastly, the set of communicable neighbors for **B** is N_1^B ={**A**,**C**}since **A** and **C** appear in the receive vector and have entries in the transmit matrix.

As of slot 4, **C**'s receive vector is **r**=(**B**,0,**B**,0). Thus, **C** never attempts to receive a packet from **D** and multi-hop communication between **B** and **D** is not possible. In a larger network, it is likely that additional terminals lying between **B** and **D** would supply additional links, and thus provide a greater chance that a route exists between **B** and **D**. For example, in Figure 12, network connectivity is preserved when a single-hop neighborhood is used. However, in a sparse topology this can be a significant problem. In Chapter 6, we discuss the challenges of using INS in networks with low terminal density, and show through simulation results that the problem is mitigated as terminal density increases. In Chapter 7, we discuss an extension to the INS protocol which allows terminals to improve connectivity by sharing additional neighbor information. In Chapter 8, we describe a further extension which improves connectivity by leveraging multi-packet reception capability of terminals equipped with advanced receiver hardware.

5.4 Link-based Adaptation of Spreading Factor

When terminal *i* transmits a unicast data packet to neighbor *j*, the outgoing link SINR estimate, denoted $\hat{\xi}_{i,j}$, may be used to adjust the spreading factor to take advantage of links with high SINR. Utilizing the maximum spreading factor, $N_{\max}$, *i* can transmit

one packet in the payload interval. When $\hat{\xi}_{i,j}$ is above a threshold, *i* reduces the spreading factor to increase the data rate. For these investigations, *i* can reduce the spreading factor by a factor of 2 or 4 as shown in Table 3. A 3dB margin must be satisfied before terminals attempt to increase the spreading factor. Broadcast data and network control packets (e.g., NBR_ACK packets) are always transmitted using the maximum spreading factor.

Link SINR Estimate	Link Spreading Factor	Link Rate
$\beta < \hat{\xi}_{i,j} \leq 4\beta$	$N_{\max}$	1 packet per slot
$4\beta < \hat{\xi}_{i,j} \leq 8\beta$	$N_{\max}/2$	2 packets per slot
$8\beta < \hat{\xi}_{i,j}$	$N_{\max}/4$	4 packets per slot

Table 3. Allowable transmission modes for various link SINR estimates.

Incoming link SINR estimates $\hat{\xi}_{i,j}$ are computed using the method described in Section 4.1. The current incoming link SINR estimate $\hat{\xi}_{i,j}$ is sent to *i* in each NBR_ACK packet, providing *i* with an outgoing link SINR estimate.

5.5 Intelligent Queue Management

Queue management is utilized to reduce control packet overhead and to exploit spreading factor adaptation. In a slot in which a terminal is scheduled to transmit, it scans its queue for candidate packets. Broadcast data and network control packets are always candidates for transmission; if the first candidate packet is a broadcast data, BLOCK, or NBR_ACK packet, the search stops and the packet is transmitted. A unicast

data packet is a candidate only if the next hop for the packet is listed in the transmit matrix for this slot. If the first candidate packet is unicast data packet, and the next hop is a neighbor for which a reduced spreading factor will be employed, then the search continues for other candidate unicast data packets which may be sent using a reduced spreading factor. If no candidate packet is found, the terminal does not transmit in the data interval. Figure 15 shows several allowable transmission scenarios. For example, suppose terminal i has packets $\{p_1, p_2, p_3, p_4\}$ enqueued for j_2, j_1, j_2, and j_1 respectively, and suppose j_1 and j_2 are both listed in the transmit matrix for this slot. Terminal i first dequeues p_1 for transmission. Since the slot is not full, i next dequeues p_2. The next packet, p_3, cannot be transmitted since there is not enough remaining time in the slot. However, p_4 can be transmitted, resulting in a scenario identical to (D) in Figure 15.

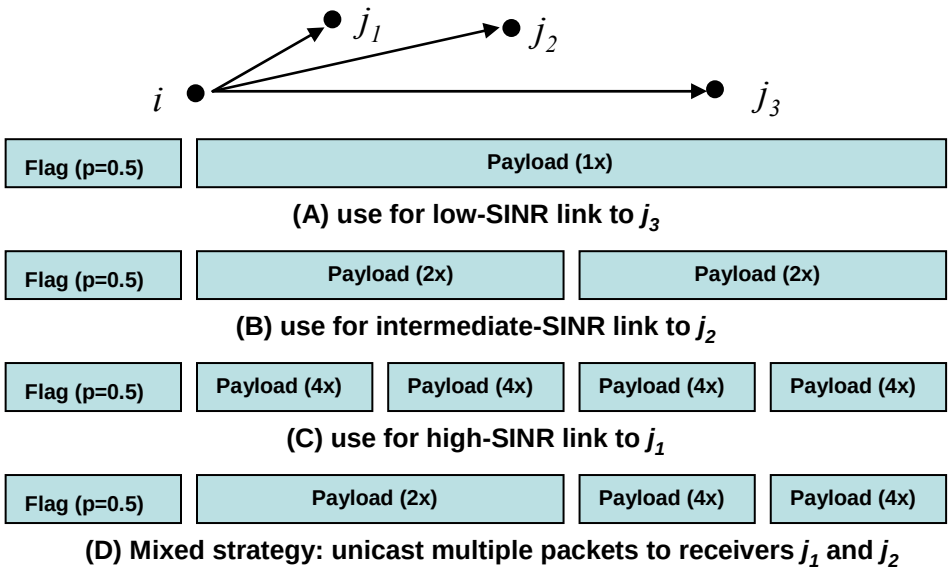


Figure 15. Example showing some feasible transmission scenarios for unicast data packets from terminal *i*.

CHAPTER SIX

EVALUATION OF INS PROTOCOL

In this section we analyze the performance of immediate neighbor scheduling using simulations. For each simulation, the number of terminals, N, is 100, and the number of trials, $|\Upsilon|$, is also 100 . *Neighbor density*, which is a measure of the average number of neighbors of a terminal, is varied by adjusting the communications range, R. Values of R of interest are {200, 250, 350}, resulting in average 1-hop neighbor counts of approximately {10.1, 15.2, 27.1} respectively. For values of R below 200, network performance is dominated by low connectivity and unavailability of routes. End-to-end packet statistics are used to compare the performance of schedules which use a 2-hop neighborhood, as used in collision-free graph-based schedules, and immediate neighbor schedules, which use a smaller neighborhood to schedule transmissions.

Section 6.1 contains a description of the 2-hop scheduler, referred to as the *Broadcast Transmission Scheduler* (BTS). In Section 6.2, we present steady-state results for networks with stationary terminals. In Section 6.3, we describe a new mobility model, and use end-to-end packet statistics to evaluate the performance of the INS protocol in mobile networks. In Section 6.4, we examine how the adaptive transmission protocol improves performance.

6.1 Centralized Collision-free Scheduler Implementation

For comparison, we have implemented a centralized broadcast transmission scheduler (BTS), designed to generate collision-free broadcast transmission schedules

assuming use of the graph-based connectivity model. The purpose for this is to determine how the gain in spatial reuse provided by the INS protocol balances with the greater number of communications links provided a traditional scheduling protocol. A greater number of communication links results in more robust network topologies. The additional links also tend to span longer distances, decreasing the average hop count for a packet to reach its destination.

When the BTS is used, the communications range is used to generate a topology graph (i.e., the 1-neighbors of a terminal are all terminals within communications range, and the 2-neighbors of a terminal are all additional terminals within communications range of 1-neighbors) which is then supplied to each terminal. Terminals are assigned unique colors among their 1- and 2-neighbors to ensure a collision-free schedule. All other details of the simulation are identical, including routing, queueing policies, spreading, and the requirement on SINR given by (3.1). Note that while BTS generates collision-free transmission schedules under the graph-based connectivity model, not all transmissions are successful because the SINR requirement may not be satisfied.

In the BTS tests, NBR_ACK packets play the same role in determining the transmit matrix and receive vector for terminals, which are then used for identifying communicable neighbors which may be used by the routing algorithm. This results in approximately equal control packet overhead for the INS and BTS tests. In a real system, a scheduling protocol using a 2-hop scheduling neighborhood would require additional overhead to manage recoloring of terminals in response to local changes in network topology. For example, the protocol in 0 describes a method for exchanging local

information across multiple hops to facilitate this type of coordination. In the BTS, terminals are automatically notified of local changes in network topology, and recoloring of terminals to account for these changes is performed automatically.

6.2 Performance in Stationary Networks

We examine the steady-state performance of the INS protocol and the centralized BTS protocol. To support our claim of higher spatial reuse with the INS protocol, in Figure 16 we show the average number of transmissions per slot under the INS protocol and the BTS for R=200m and R=250m. Under the INS, terminals have fewer neighbors with which they must share the channel, so they may be able to operate with shorter transmission frames and transmit more often.

When more transmissions are allowed per slot, the MAI is greater and some neighboring terminals are more difficult to reach. As a result, packets tend to take shorter hops and must be forwarded more times to reach their destinations. The average hop count for successful packets is shown in Figure 17 for R=200m and R=250m. The INS protocol results in one to two additional hops per packet when compared to the BTS.

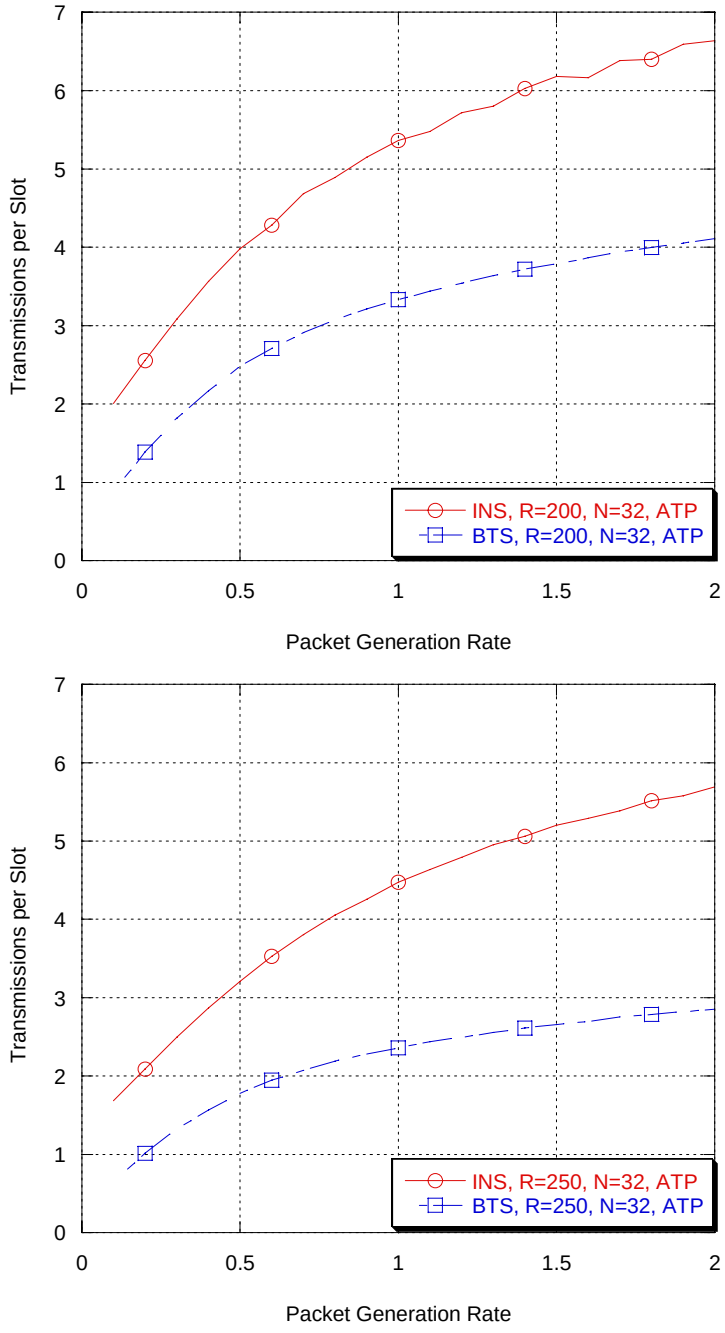


Figure 16. Average number of transmissions per slot for INS and BTS, *R*=200m (top) and *R*=250m (bottom).

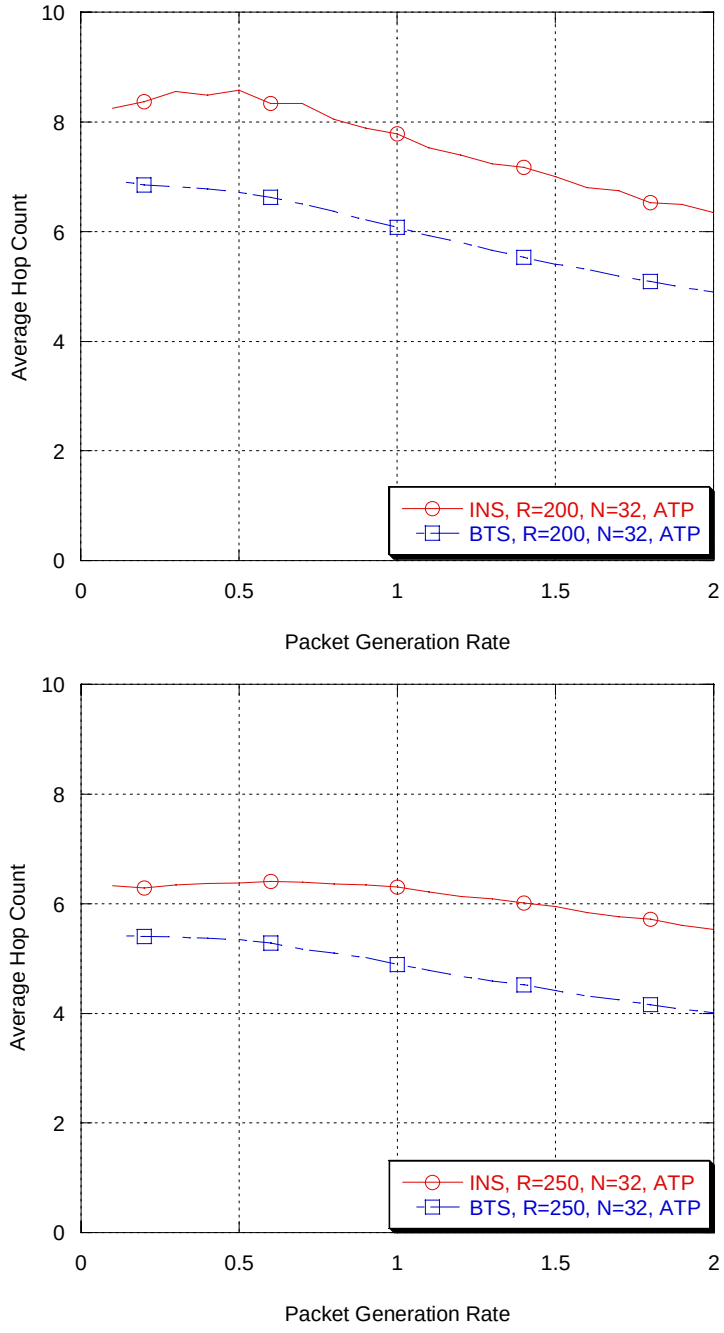


Figure 17. Average hop count of successful packets for INS and BTS, *R*=200m (top) and *R*=250m (bottom).

Steady state end-to-end packet statistics when *R* is 200m, 250m, and 350m, are shown in Figure 18, Figure 19, and Figure 20, respectively. At *R*=200m, the packet delay is lower when the INS protocol is used, and the throughput appears similar. However the packet completion rate is actually much lower than that provided by the BTS. At this density, the major challenge faced by the INS protocol is network connectivity. The BTS also exhibits a packet completion rate slightly below 100% at low packet generation rates for the same reason, but the connectivity is much better overall. A closer examination of the behavior of the INS protocol reveals the causes for this difference.

The INS protocol allows a terminal to detect most terminals within communications range through reception of FLAG packets, but not all of these become communicable neighbors. This is due to two factors: first, terminals may only capture packets from one transmitter in each slot, so if two or more neighbors transmit in the same slot, then only one transmitting neighbor becomes a communicable neighbor. Second, greater spatial reuse in the INS tests results in greater MAI and lower link SINR values; if the SINR of a particular link is very low in the BTS tests, then the SINR of the same link in the INS tests may be below the threshold required for packet reception. A test network is *disconnected* when one or more terminals are isolated from the other terminals in the network. In the INS tests, the combination of greater MAI and fewer communicable neighbors causes more test networks to be disconnected.

Disconnected networks cause more packets to be dropped since there is no route for some source-destination pairs. For example, in the INS tests, when *R*=200m and the packet generation rate $\gamma = 0.5$ packets per slot, 7% of generated traffic is lost due to

routing failures, 3% is lost due to queue overflow; the remaining 5% is lost due to insufficient SINR at the receiver or the wrong transmitter being selected by the receiver. When the BTS tests are run at the same generation rate, 4.4% of generated packets are lost due to queue overflow, and 0.6% are lost due to routing failures.

At very low packet generation rates ($\gamma \leq 0.2$ packets per slot), the packet completion rate of the INS protocol decreases slightly because terminals do not transmit often enough establish reliable transmitter-receiver matchings: for example, if the data input rate to the network is zero, then terminals *only* transmit FLAG packets and periodic NBR_ACK packets. In this situation, it is difficult for a terminal *i* to determine if a neighbor should be in set N_i^1 or set N_i^2 . This results in establishment of fewer links to neighbors, a greater number of disconnected networks, and a larger portion of transmissions which fail due to the receiver correlating to the wrong spreading sequence.

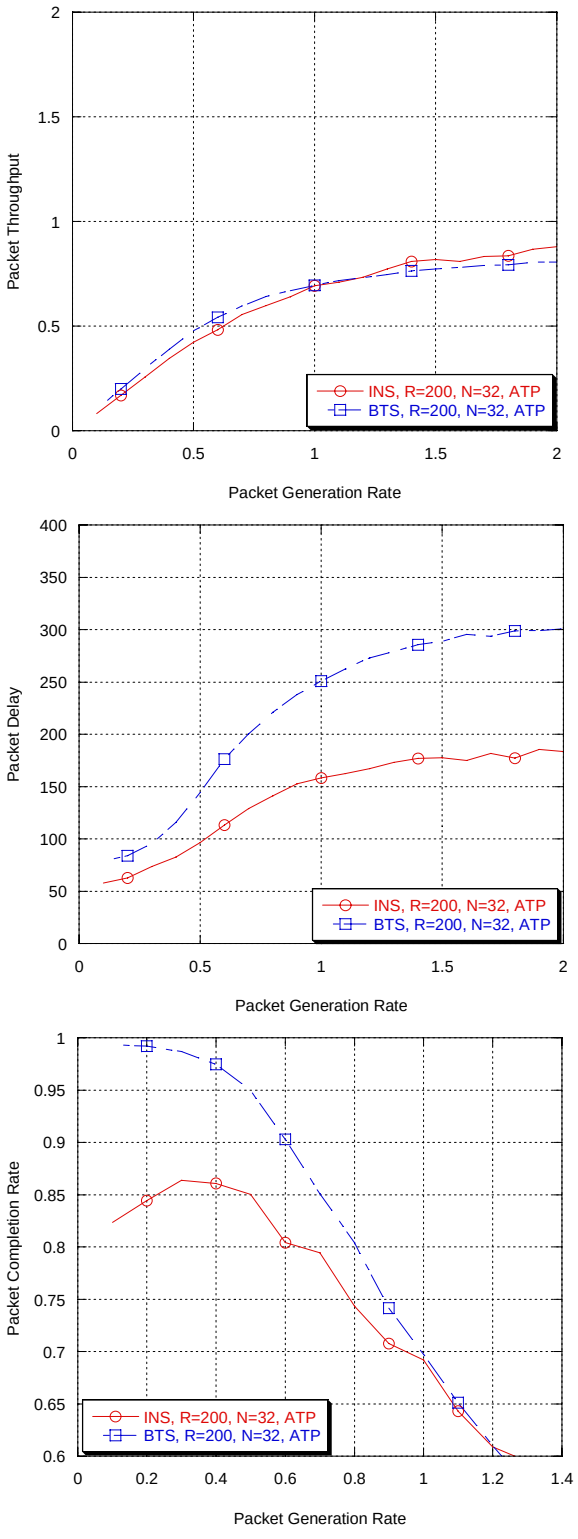


Figure 18. Packet throughput, delay and completion rate when R=200m.

At higher network densities (R=250m and R=350m) network connectivity is more robust, so more links for routing packets are available and eliminating collisions is not as important. When $\gamma > 0.5$ packets per slot, the INS protocol supports higher packet completion rates with much lower packet delay than the BTS tests. The BTS tests provide 100% packet completion rate at lower values of γ, but terminals in these tests are provided with perfect information about all reachable terminals within two hops. In a real distributed network, this information would be very difficult to obtain.

The INS tests achieve a higher overall packet completion rate at high packet generation rates since terminals are able to transmit more often. This improvement comes despite the fact that each transmission, on average, traverses a shorter distance. However, the analysis in [53] shows that theoretical wireless network transport capacity is maximized when neighbor density is just dense enough to ensure network connectivity while allowing for maximal spatial reuse. The gain in performance from scheduling using a smaller neighborhood mirrors this result, and becomes much more noticeable at the higher neighbor densities.

Most packet drops in the BTS tests are due to queue overflow; this is also the case for the INS protocol when R=250m or R=350m. At a packet generation rate of $\gamma = 1.0$, when R=250m, 1% of generated packets are dropped because no route to the destination exists, 10% of generated packets are dropped due to queue overflow, 0.35% are lost due to insufficient SINR, and 0.9% are lost because the receiver is attempting to receive a transmission from a different transmitter. When R=350m, the corresponding rates of packet loss are 0.03%, 3%, 0.15%, and 0.24%, respectively.

It is difficult to compare the performance of the INS and BTS tests across different values of *R*. As *R* increases, the average number of hops required for a packet to reach its destination decreases; this allows the network to support a greater level of traffic. However, the neighbor density also increases with *R*, leading to longer transmission frames and less spatial reuse. This tends to reduce the level of traffic the network may support. Despite this complex tradeoff, we claim that INS protocol has superior scalability as neighbor density increases based upon the following observation: as the value of *R* increases, the performance of the INS schedules continues to improve, while the performance of the BTS schedules remains about the same. This reinforces the above conclusion that the INS scheduling approach better leverages the multiple access capability of DSSS than the approach used for BTS schedules.

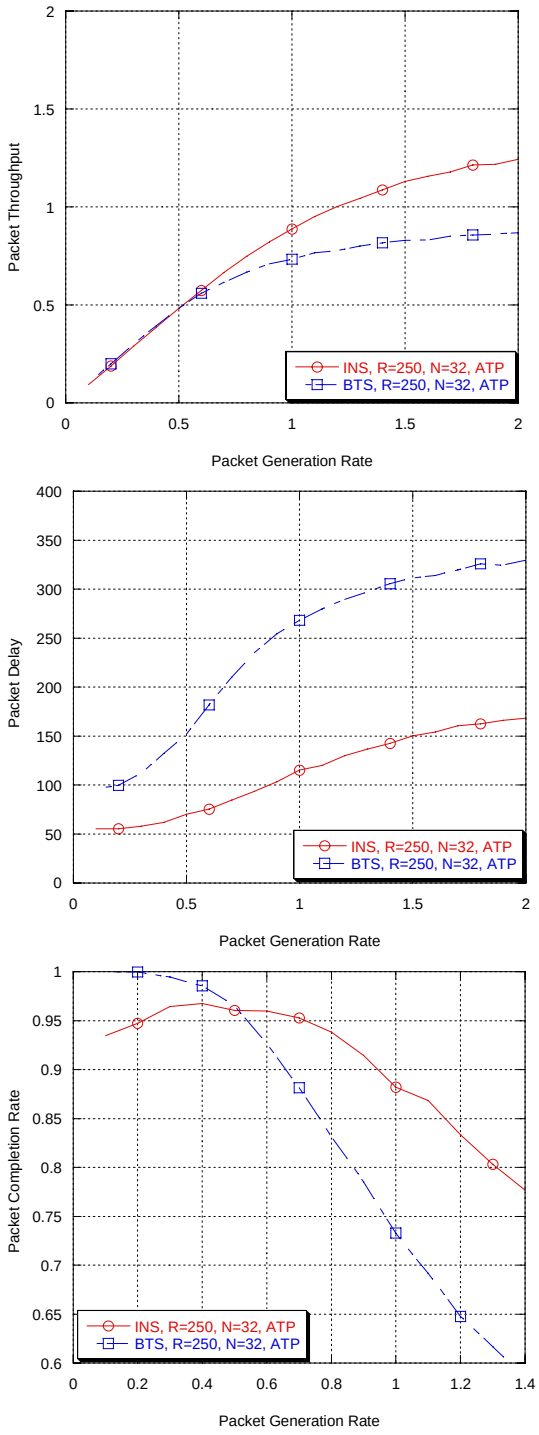


Figure 19. Packet throughput, delay, and completion rate when R=250m.

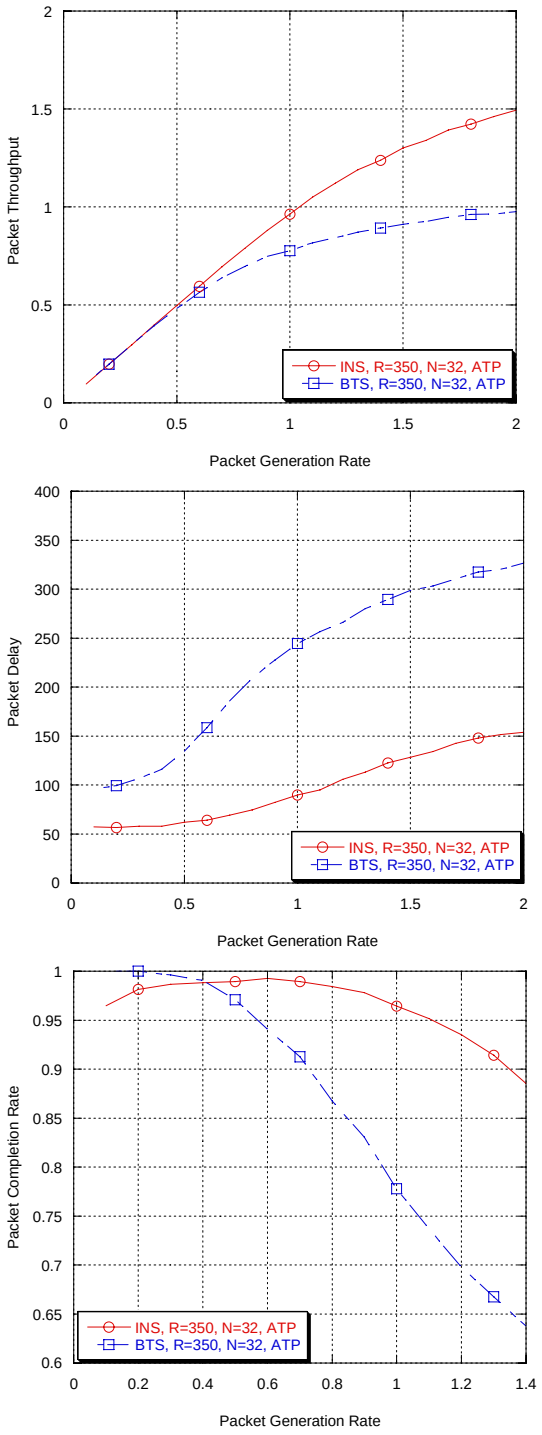
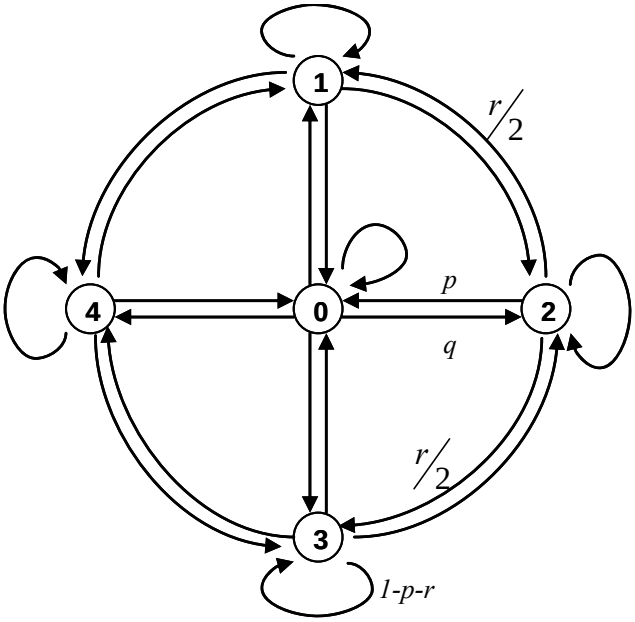


Figure 20. Packet throughput, delay and completion rate when R=350m.

6.3 Performance in Mobile Networks

To examine the performance of the INS protocol in dynamic environments, we use a five-state discrete-time Markov chain to model terminal mobility. In state 0, a terminal is stationary. In states 1, 2, 3, and 4 a terminal may move north, east, south, and west, respectively. Terminals in motion all move at the same speed. The rate of mobility-induced topology changes depends upon the speed of mobile terminals, as well as two probability parameters, denoted p and q, which define how long terminals are in the mobile states (states 1 – 4) and how long a terminal is in the stationary state (state 0). A third parameter, r, defines the rate at which terminals in motion turn right or left. When a stationary terminal begins moving, its initial direction is equally likely among the four mobile states. When a terminal changes direction, it turns left or right with equal probability. The mobility state of each terminal is updated in every slot. A diagram of the mobility model and its state transition probability matrix are shown in Figure 21. We use this model in lieu of the random waypoint mobility model [54] because it is easier to create special-case mobility scenarios, and because it maintains a more even spatial distribution of terminals. Table 4 shows the values of the mobility parameters used in simulations.



$$\begin{bmatrix} 1-q & \frac{q}{4} & \frac{q}{4} & \frac{q}{4} & \frac{q}{4} \\ p & 1-p-r & \frac{r}{2} & 0 & \frac{r}{2} \\ p & \frac{r}{2} & 1-p-r & \frac{r}{2} & 0 \\ p & 0 & \frac{r}{2} & 1-p-r & \frac{r}{2} \\ p & \frac{r}{2} & 0 & \frac{r}{2} & 1-p-r \end{bmatrix}$$

Figure 21. Mobility model and state transition matrix.

If we define a D as the random variable representing the number of slots a terminal moves in a single direction, then the expected value of D is

$$E[D] = \sum_{n=1}^{\infty} n\left(1-\left(1-p-r\right)\right)^n = \frac{1}{p+r}. \qquad (6.1)$$

Similarly, when a terminal stops moving the expected amount of time spent in the stationary state is $1/q$ slots. When mobile terminals reach a boundary, they are reflected back in the opposite direction.

p: probability a mobile terminal stops moving	0.000065
q: probability a stationary terminal begins moving	0.02
r: probability a mobile terminal turns 90º left or right (equally likely)	0.000065

Table 4. Mobile parameters used in simulations.

In the INS mobility tests, terminals are initialized with color 1 and no knowledge of surrounding terminals, as in the stationary tests. We use the performance of the BTS in mobile networks for comparison. Terminals in the BTS mobility tests are initialized with a greedy coloring based upon the topology of the network at the beginning of the simulation. In each slot, the underlying topology graph (based upon communications range, R) is updated, and the neighbor tables of all terminals are updated to reflect the current topology. In both the INS and BTS tests, the centralized routing is recomputed every 64 slots. The terminals move according to the schedule in Table 5. Speed is quoted in meters/second, and we assume there are 150 time slots per second in the simulation. During low mobility periods, there are approximately 0.25 and 0.28 link changes per slot for R=200m and R=250m, respectively. During high mobility periods, there are approximately 0.5 and 0.55 link changes per slot for R=200m and R=250m, respectively. In all cases, the packet generation rate is 0.5 packets per slot for the duration of the mobile simulations.

Time Period (slots)	Speed (m/s)
(1,5000)	0
(5001,10000)	10.0
(10001, 15000)	20.0
(15001, 25000)	0

Table 5. Terminal mobility in mobile scenarios.

To evaluate performance in mobile scenarios, we use a windowed throughput measurement: the simulation maintains a running total of delivered packets for the network, and every 64 slots an instantaneous throughput estimate, measured in packets per slot, is obtained by dividing the total delivered packets by 64. The running total is then reset and a new window starts. Figure 22 shows the throughput during the 4-stage simulation for R=200m and R=250m. The INS protocol maintains a high level of throughput during mobile periods, showing that the schedule is able to continuously adapt coloring and neighbor tables to the changing topology. In all tests, the majority of packet drops are due to routing failures. When two terminals move within communications range, they must detect each other through FLAG transmissions and successfully exchange NBR_ACK packets before they can both use the other for routing packets. This process takes longer if terminals transmit less often, as in the BTS tests. Thus, while coloring conflicts are automatically resolved by the BTS, it still takes the BTS some time to recover from topology changes. This process is faster in the INS tests because terminals transmit more often, allowing faster recovery from topology changes.

As terminals move, it may happen that a terminal that has a high color number because it is in a dense area of the network moves to a less-dense area of the network. In this case, the terminal would keep its original color number since no coloring conflicts occur. However, this is inefficient if the terminal could choose a new, lower color number. As a result, a slight, gradual decrease in performance occurs during mobile periods, particularly in the BTS tests. To correct for this, at time slot 15,000 terminals in

the BTS tests are recolored, and throughput diminishes until neighbor tables are reconstructed. The coloring is not modified in this manner in the INS tests.

Even though it is a distributed protocol, the INS does not require an extended initialization period to identify neighbors and compute a schedule. Figure 23 shows a close-up of the second graph in Figure 22 (R=250) to demonstrate the initialization behavior. As terminals detect new neighbors, they may change color several times and generate several NBR_ACK packets. By time slot 400, however, the INS protocol is delivering packets at a higher rate than the BTS, clearing backlog that built up as neighbors were detected.

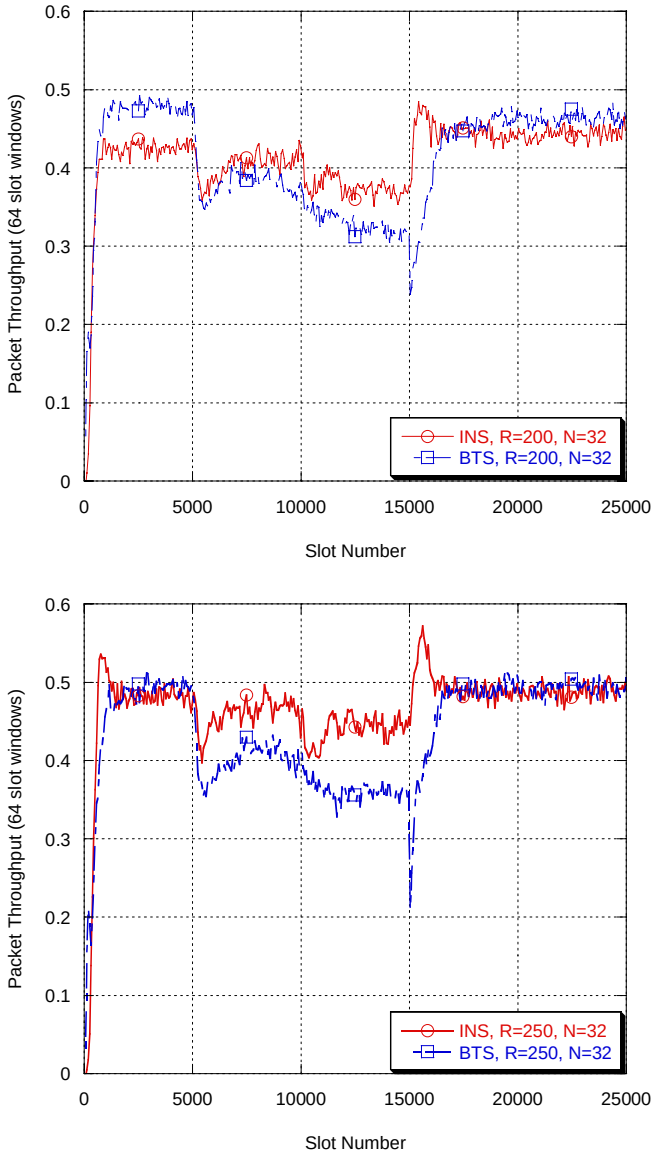


Figure 22. Packet throughput during 4-stage mobile scenarios for *R*=200 and *R*=250.

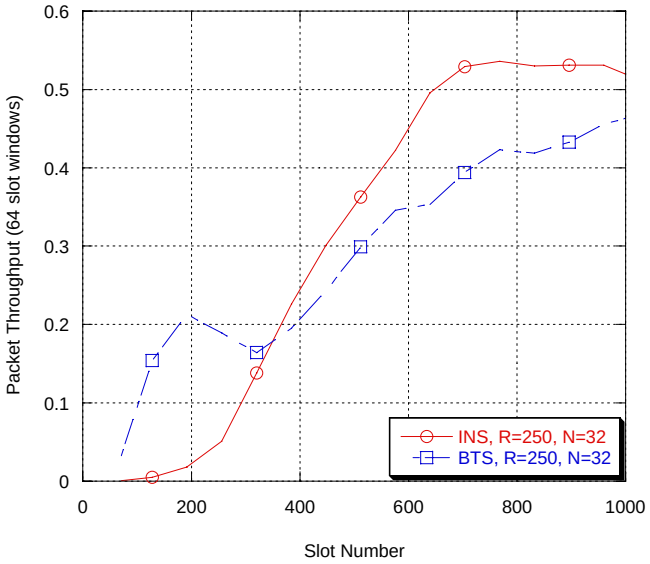


Figure 23. Close-up of packet throughput measurements to show initialization behavior.

6.4 Utility of Spreading Factor Adaptation

Link-based adaptation of spreading factor is used in both the INS tests and the BTS tests to a substantial degree. For example, at R=350m, both the INS and BTS tests use the highest transmission rate (lowest spreading factor) for approximately 80% of transmissions, regardless of the packet generation rate; approximately 5% of packets are transmitted using the highest spreading factor, and around 15% of packet transmissions use the intermediate spreading factor of 16. When the adaptive transmission protocol (ATP) is deactivated, the maximum stable throughput is much lower. In Figure 24, the packet completion rate for the INS and BTS tests is shown for tests in which the adaptive transmission protocol is deactivated; these are plotted with the results in Section 6.2 for comparison. We consider the same test cases as before: R=200, R=250, and R=350.

Without the adaptive transmission protocol, the BTS tests cannot support a 90% completion rate for γ values greater than 0.5, regardless of the value of R. When the INS is used without the ATP, the performance is especially poor in networks with sparse connectivity. For R=250m and R=350m, the INS tests only support a 90% packet completion rate for γ=0.45 and γ=0.65, respectively, when the ATP is not used. Thus, the ATP improves INS protocol's performance by approximately 100% at R=250m and R=350m. This illustrates the value of the cross-layer protocol design. By combining the greater spatial reuse of the INS protocol with the flexibility of adaptive transmission rates, large performance gains for end-to-end traffic are realized.

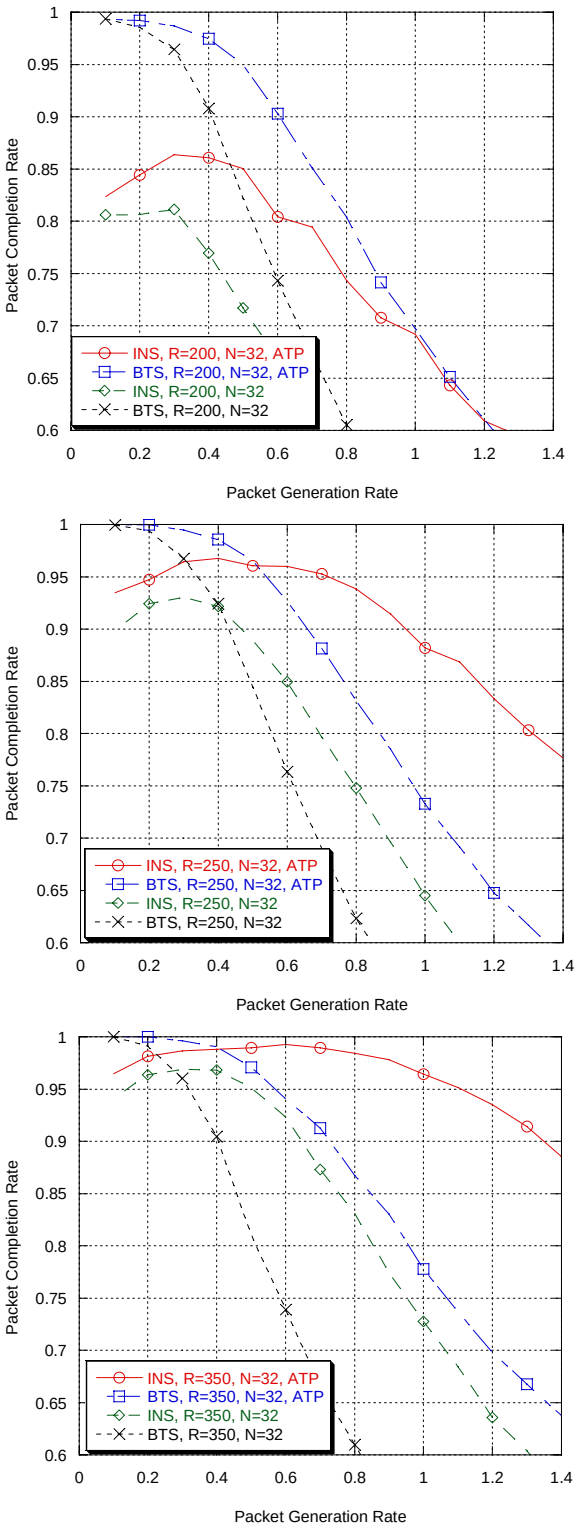


Figure 24. Utility of spreading factor adaptation in the INS and BTS tests.

CHAPTER SEVEN

ENHANCING INS WITH SELECTIVE COLLISION ELIMINATION

In dense networks with a random distribution of terminal locations, the information obtained from immediate neighbors is sufficient to generate a schedule and network topology which supports multi-hop communications between any pair of terminals in the network. In sparse networks, however, important links which are necessary for connectivity are not available when the INS protocol is used, causing the network to become disconnected and some destinations to be unreachable. This is noted in the simulation results of Chapter 6 when the neighbor density is low (R=200m).

This can also be seen in the example network in Section 5.3, using Figure 14. At the end of the example, **B** can communicate with **A** and **C** because they are both members of N_1^B. However, **C**'s receive vector is $\mathbf{r}^{(C)}$=(**B**,0,**B**,0). **D** is not listed in $\mathbf{r}^{(C)}$, so **D** is a member of N_2^C. Since the schedule does not support bidirectional communication between **C** and **D**, the network is disconnected (i.e., there is no route between one or more source-destination pairs). In sparse networks, such as the network of Figure 14, it is important to establish bidirectional communication with as many neighbors as possible to improve network connectivity.

We extend the INS protocol by allowing terminals to change the transmission schedule in order to increase the number of communicable neighbors. This process is termed *selective collision elimination*. Here the term *collision* refers to the event of two neighboring terminals transmitting in the same time slot, and it does not specifically refer

to the success or failure of a particular transmission. Selective collision elimination is accomplished via messages, termed *BLOCK* packets, which deliver information about non-immediate neighbors to surrounding terminals so that they may be accounted for in the schedule. In practice, a terminal only needs to know about a few of these non-immediate neighbors in order to recover links necessary for connectivity.

The message contained in a BLOCK packet is a four-tuple (*a*, *s*, *c*, *b*), where *a* is the ID of a neighbor, *s* is a slot number, *c* is a color number, and *b* is another terminal ID representing the intended recipient of the BLOCK message. When terminal *i* receives a BLOCK packet with message (*a*, *s*, *c*, *b*), *i* adds terminal *a* to its neighbor table and, if necessary, changes its color to maintain a unique color among the terminals in its neighbor table.

The remainder of this chapter is structured as follows: in the next section we provide an example of selective collision elimination. We describe two approaches for implementing selective collision elimination in Sections 7.2 and 7.3. In Section 7.4, we evaluate selective collision elimination using simulations.

7.1 INS Example Extended with Selective Collision Elimination

To demonstrate the operation of BLOCK packets, we continue the example from Section 5.3 with selective collision elimination enabled. The next sequence of slots is shown in Figure 25 starting with slot 5 (slot 1 of the 4-slot frame). In slot 6, assume **A** transmits in both the identification and data intervals while **C** transmits only in the data interval. This allows **B** to detect that two terminals are transmitting in the same slot if **B** captures the FLAG from **A** and receives the packet from **C** (the receive vector of **B** is

$\mathbf{r}^{(B)}$=(0,**C**,0,**A**)). For this example, assume **B** responds to detection of two transmitting terminals in this slot by generating a BLOCK packet with message (**C**,2,2,**A**). This message is constructed to inform terminal **A** that terminal **C** also transmits in slot 2 of each frame with color 2.

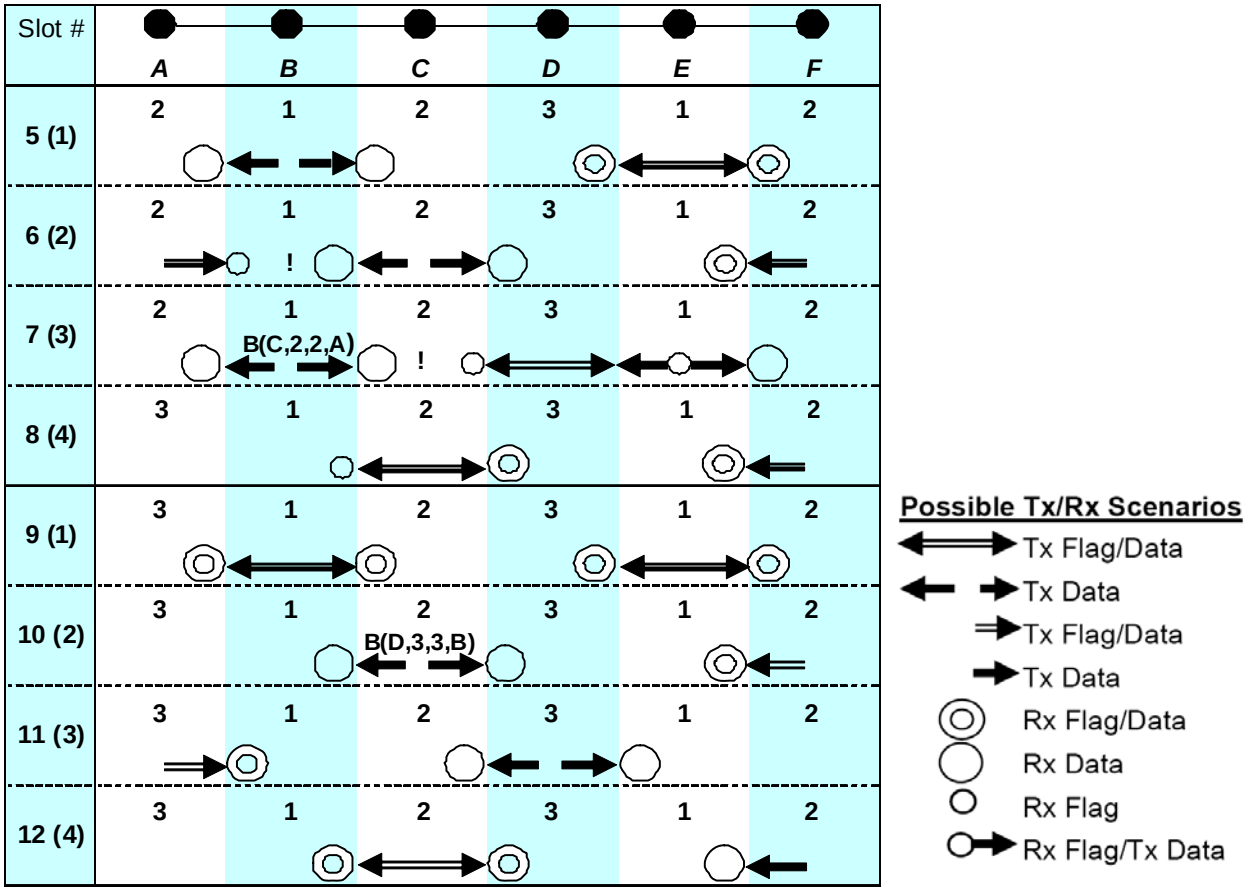


Figure 25. Six-terminal network example extended to demonstrate operation of BLOCK packets.

In slot 7, **B** transmits the BLOCK packet. Terminal **A** receives the BLOCK packet and learns of terminal **C**, which also has color 2; **A** adds **C** to its neighbor table and changes its own color to 3. Meanwhile, terminal **D** transmits in the identification and data intervals of slot 7. **C** detects two terminals transmitting in the same slot when it

receives the FLAG from **D** and the BLOCK packet from **B**. For this example, assume **C** constructs a BLOCK packet with message (**D**,3,3,**B**). **C** waits until slot 10 to transmit the BLOCK packet, since $t_{B,2}^{C} = 1$ and slot 10 is slot 2 of the transmission frame. No color change results from this BLOCK packet; however, terminal **B** adds **D** to its neighbor table and no longer transmits in the 3rd slot of each frame.

Intelligent use of information gained from FLAG bytes can help reduce the amount of time required to regain links when neighbors change color. To see this, note that in slot 8, **B** is still trying to detect a transmission from **A** even though **A** no longer transmits in that slot. In slot 11, **B** learns of **A**'s color change via FLAG reception. Terminals with color 3 cannot transmit in slot 4 of the frame, so **B** clears the receive vector entry listing **A** in the 4th slot of the frame (refer to Section 5.2 for an example).

At this point, the network supports multi-hop communications between any pair of terminals, even when the neighbor entries generated by BLOCK packets expire. This is a consequence of the coloring which was induced by the BLOCK packets. In general, however, this is not the case and more BLOCK packets may be generated as neighbor table entries gained through BLOCK messages expire.

7.2 Implementing Selective Collision Elimination with a Fixed Threshold

A common *blocking threshold* parameter, BT, shared by all terminals, represents the minimum SINR for a received packet that causes a BLOCK packet to be created. When terminal *i* receives a FLAG transmission in slot *s* from terminal *j*, *i* forms an SINR estimate $\hat{\xi}_{j,i}$ for the received FLAG. If $r_s^i = k$, where $k \neq j$ and $k>0$, and $\hat{\xi}_{j,i} >$ BT, then *i*

creates a BLOCK message. If $c_j > c_k$, the BLOCK message is (j, s, c_j, k). Otherwise, the BLOCK message is (k, s, c_k, j).

A lower value for the BT allows for more BLOCK packets to be created. If BT is set to 0, then terminal i generates a BLOCK whenever it receives a FLAG packet from a terminal for which it is not scheduled to receive from during that data interval of the same slot. The resulting schedule is very close to a collision-free broadcast schedule, but the overhead to achieve this can be significant. As a result, while a low value of BT is valuable in networks where connectivity is an issue, it becomes a liability (in terms of overhead) when connectivity is robust. If a larger value of BT is used, then a BLOCK packet is generated only when the transmitter identified in the FLAG is a strong source of interference during the data interval.

7.3 Implementing Selective Collision Elimination with a Variable Threshold

Each terminal i calculates a blocking threshold bt_j for each neighbor j based upon the SINR estimate of link (j, i). Consider a scenario in which terminal i lists j in the receive vector for slot s, i.e. $r_s^i = j$, and another nearby terminal k begins transmitting in slot s. If k generates significant additional MAI, then the SINR requirement for link (j, i) may no longer being satisfied. This condition is expressed as

$$\frac{P_r(j)N_{\max}}{N_0' + P_r(k)} \leq \beta , \tag{7.1}$$

where $P_r(x)$ represents the power received at i from transmitter x, and N_0' represents the sum of thermal noise and aggregate MAI from transmitters besides k. By rearranging terms, (7.1) may be rewritten as

$$\frac{1}{\beta}\frac{P_r(j)N_{\max}}{N_0'} \le 1 + \frac{P_r(k)}{N_0'}. \tag{7.2}$$

The incoming link SINR estimate for neighbor j, $\hat{\xi}_{j,i}$, is made relative to the maximum spreading factor, $N_{\max}$ (see Section 4.1). We make the approximation

$$\hat{\xi}_{x,i} \cong \frac{P_r(x)N_{\max}}{N_0'} \tag{7.3}$$

and substitute into (7.2), yielding

$$\frac{1}{\beta}\hat{\xi}_{j,i} \le 1 + \frac{\hat{\xi}_{k,i}}{N_{\max}}. \tag{7.4}$$

Next, we solve for the value of $\hat{\xi}_{k,i}$ which results in equality in (7.4), and use that as a basis for bt_j. We define the blocking threshold for link (j, i), to be bt_j to be

$$bt_j = cN_{\max}\left(\frac{1}{\beta}\hat{\xi}_{j,i} - 1\right). \tag{7.5}$$

In (7.5), we set c = 0.5 to provide approximately 3dB of additional protection for link (j,i). The SINR estimate for link (j, i) is the minimum of several sample estimates, while the SINR estimate for the transmission from k, $\hat{\xi}_{k,i}$, corresponds to a single sample. The sample link SINR estimates may be imprecise, and may vary from one time slot to another depending upon transient conditions, so the additional 3dB of protection increases the robustness of the protocol.

When using the variable threshold method, the routing table of i is used in conjunction with the neighbor blocking thresholds to determine when to create a BLOCK packet. A BLOCK packet may be created in two cases. In the first case, if i receives a

FLAG transmission from k in slot s, and there is currently no route to k, and $r_s^i \neq k$, then i creates a BLOCK packet with message (k, s, c_k, j), where $j = r_s^i$ (note this does not depend upon bt_j). Terminal i also sets $r_s^i = k$. When terminal j receives this BLOCK packet, j adds k to its neighbor table and, if necessary, changes its color. With k added to the neighbor table, j will no longer transmit in slot s, and it is easier for k to become a communicable neighbor of i. Since no route to k existed previously, this improves network connectivity.

In the second case, if i receives a FLAG transmission from a neighbor k in slot s, and there exists some other terminal j for which $r_s^i = j$, $bt_j < \hat{\xi}_{k,i}$, and the next hop for routing packets to j is j itself, then i creates a BLOCK packet with message (j, s, c_j, k). When terminal k receives this BLOCK packet, k adds j to its neighbor table and, if necessary, changes its color. This ensures that i can continue receiving NBR_ACK packets from j, so that link (i, j) may continue to be used for forwarding packets.

By using the routing table and a variable threshold to determine when to generate BLOCK packets, terminals are better able to control the additional overhead of selective collision elimination. A terminal does not create a BLOCK packet unless doing so improves network connectivity, or preserves links which are used for forwarding packets. In networks with robust connectivity, fewer BLOCK packets are created since the connectivity of these networks is very robust. In networks with sparse connectivity, more BLOCK packets are created to establish and preserve important communications links.

7.4 Evaluation of Selective Collision Elimination

We use simulations of stationary networks to determine the steady-state performance of the INS protocol when selective collision elimination is enabled. We consider the variable threshold method for generating BLOCK packets, as well as the fixed threshold method with BT=100. In preliminary studies, we examined the use of other fixed thresholds, specifically 10, 20, 50, 200, and 1000. The low thresholds (10, 20 and 50) result in poor performance when R=250m or R=350m due to the large amount of additional overhead generated. The larger thresholds (200 and 1000) perform better at higher network densities, but they do not achieve a 90% packet completion rate when R=200m. By setting BT=100, a 90% packet completion rate can be reached when R=200m, and excessive overhead in more dense networks is avoided.

In Figure 26, the packet completion rate is shown as a function of packet generation rate when R is 200m, 250m, and 350m for the variable-threshold and the fixed-threshold implementations. The performance results from Chapter 6 (base INS protocol and BTS) are also included in the plots for comparison. When R=200m, the variable threshold performs almost as well as the BTS, providing a 90% packet completion rate up to γ=0.6. The fixed threshold also improves the performance of the INS protocol, but to a lesser extent. When the neighbor density is increased (R=250m and R=350m), both methods result in decreased performance when compared to the base INS protocol. There are two reasons for this. First, the BLOCK packets themselves create additional overhead in the network. Second, the additional neighbor table entries created by BLOCK packets result in less spatial reuse.

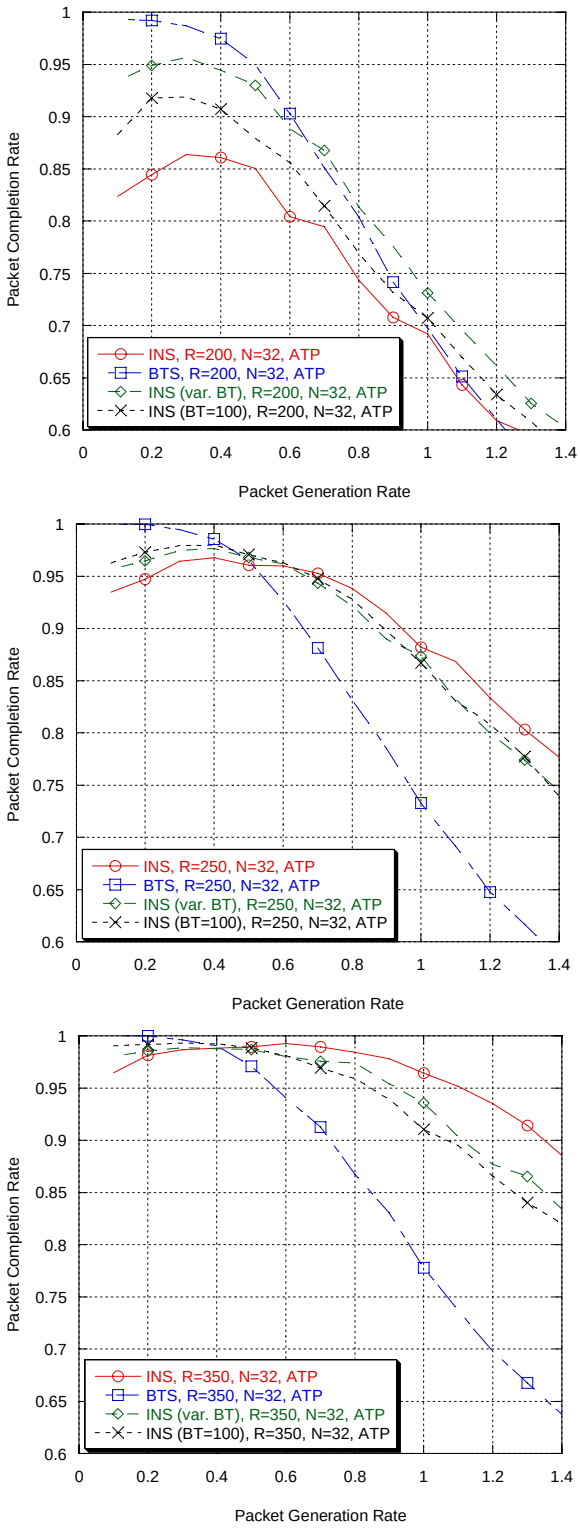


Figure 26. Performance of selective collision elimination using the fixed-threshold and variable-threshold methods.

Comparing the performance of the fixed and variable thresholds, it is evident that the variable threshold performs considerably better in networks with sparse connectivity, and maintains a slight performance edge over the fixed threshold in networks with robust connectivity. This is because the variable threshold is able to adapt the level of overhead generated from BLOCK packets to the network conditions. To better illustrate this point, Figure 27 shows the overhead from BLOCK packets, expressed in BLOCK packets transmitted per slot, for the fixed and variable thresholds and for various values of R. Using the fixed threshold, the overhead is similar in networks with different neighbor densities. The variable threshold, however, is able to achieve reduced overhead in networks with high neighbor densities.

In summary, the variable blocking threshold dramatically improves the performance, in terms of packet completion rate, of networks with sparse connectivity at low to medium traffic loads. The fixed blocking threshold also improves performance in these networks, but to a smaller extent. The BTS has a greater packet completion rate for these scenarios (sparse connectivity and low traffic load), but it achieves this because it is provided with topology information that is not available to the distributed INS protocol. In networks with robust connectivity and high traffic loads, selective collision elimination imposes a performance penalty to the INS protocol due to increased control packet overhead and decreased spatial reuse. In the next chapter, we modify the INS protocol to leverage multi-packet reception capability; this approach does not increase control packet overhead or decrease spatial reuse.

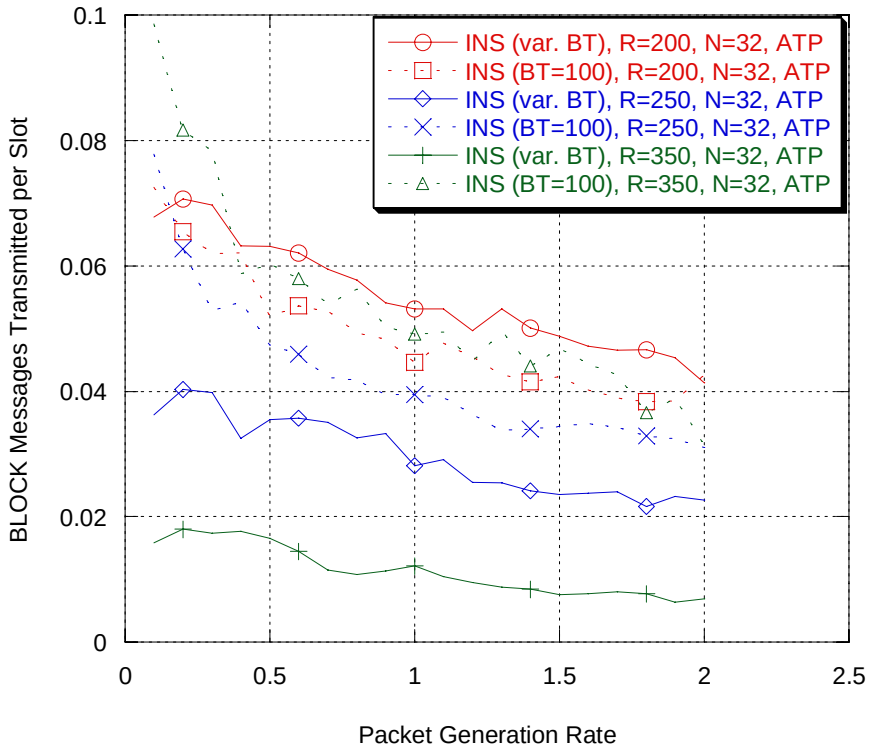


Figure 27. Overhead due to BLOCK packets using the fixed-threshold (dashed) and variable-threshold (solid) methods.

CHAPTER EIGHT

ENHANCING INS WITH MULTI-PACKET RECEPTION CAPABILITY

The 1-hop scheduling neighborhood used by the INS protocol allows two or more transmitters within communications range of a receiving terminal to transmit in the same slot. In the investigations of the INS protocol and selective collision elimination thus far, we have assumed that a terminal can only receive from one transmitter during the data interval of a time slot. However, the current generation of software-defined radios may exploit this feature of the INS protocol, receiving data from multiple transmitters in parallel. This capability is termed *multi-packet reception* (MPR). In this chapter, we introduce MPR as a means to improve connectivity in networks using the INS protocol without increasing control packet overhead. In the next chapter we investigate how MPR may improve the throughput capacity of a network.

When the INS protocol is used in sparse networks, there are two reasons for lower network connectivity: first, some links which are critical for connectivity have very low link margin, so they can tolerate little MAI. These links may be available when a 2-hop scheduling neighborhood is used, but unavailable when the INS protocol is used due to increased MAI as a result of using a 1-hop scheduling neighborhood. Second, under the INS protocol, fewer neighboring terminals are *available* to receive from a transmitting terminal. This is because these neighboring terminals may be attempting to receive a packet from another transmitter in the same slot. MPR capability improves *receiver availability* since a terminal in receive mode can select multiple neighboring terminals

from which to attempt to receive a packet in each slot. As a result, more bidirectional communications links are formed.

In the following section, we describe the model for MPR used in this work, as well as an analysis of transmission scenarios for which MPR is feasible. Section 8.2 provides a description of how the INS protocol is modified to account for MPR capability. In Section 8.3 we evaluate the network performance of the INS protocol with modifications for MPR capability using simulations.

8.1 Analysis of MPR Feasibility

The wireless channel is modeled using the *physical interference model* defined in Chapter 4. Throughout the remainder of this work, it is convenient to use the composite SINR threshold $\beta'_{i,j} \equiv \frac{\beta}{N_{i,j}}$ as the requirement for successful reception of a transmission from terminal *i* to terminal *j* employing a spreading factor $N_{i,j}$ In addition, we use $N'_0 \equiv \frac{N_0}{T_c}$ to represent the thermal noise power normalized by chip duration. Using the modified SINR threshold $\beta'_{i,j}$, when a link (*i*,*j*) is activated for a transmission from terminal *i* to terminal *j*, the modified SINR for the link, denoted $\xi'_{i,j}$, must satisfy

$$\xi'_{i,j} = \frac{P_r(i)}{N'_0 + \sum_{\forall k \neq i} P_r(k)} > \beta'_{i,j}. \tag{8.1}$$

For successful reception, it is required that $\xi'_{i,j} > \beta'_{i,j}$. Multi-packet reception (MPR) capability is modeled as the ability to receive from two or more transmitters simultaneously, provided the SINR, for each transmission satisfies this requirement.

To facilitate the analysis in this section, the power received from a transmitter, *i*, at a receiver, *k*, is the product of the transmission power, P_t, and the channel gain, $G_{i,k}$, which is a function of the distance d_i between *i* and *k*, as well as a path-loss exponent α. Thus, we have

$$G_i = d_i^{-\alpha} . \tag{8.2}$$

We assume all terminals use identical transmission power and chip rate, and all receivers experience the same level of thermal noise per chip, N_0'. The communications range, *R*, represents the maximum transmission distance when there is no MAI and the maximum spreading factor, $N_{\max}$, is employed (note that in the presence of MAI, the feasible transmission range is less than the communications range). Given the SINR threshold $\beta' \equiv \frac{\beta}{N_{\max}}$, thermal noise, N_0', and communications range *R*, the transmission power is set as

$$P_t = \beta' N_0' R^\alpha . \tag{8.3}$$

Consider the simultaneous reception of signals from transmitters *i* and *j*, both using spreading factor $N_{\max}$, at a receiver *k*. In the following, the symbol $MAI \equiv \sum_{k \neq i,j} P_t d_k^{-\alpha}$ represent the interference from transmitters other than *i* and *j*. The SINR calculations for transmissions from *i* and *j* to *k* are

$$\xi_{i,k}' = \frac{\left(\beta' N_0' R^\alpha\right) d_i^{-\alpha}}{N_0' + MAI + \left(\beta' N_0' R^\alpha\right) d_j^{-\alpha}} . \tag{8.4a}$$

and

$$\xi'_{j,k} = \frac{\left(\beta' N'_0 R^{\alpha}\right) d_j^{-\alpha}}{N'_0 + MAI + \left(\beta' N'_0 R^{\alpha}\right) d_i^{-\alpha}}. \tag{8.4b}$$

Because we require $\xi'_{i,k} > \beta'$ and $\xi'_{j,k} > \beta'$ for the successful activation of links (i, k) and (j, k) the transmission distances must satisfy

$$d_i < \left(\frac{d_j^{\alpha}}{\beta' + \left(\frac{d_j}{R}\right)^{\alpha} \left(\frac{N'_0 + MAI}{N'_0}\right)} \right)^{1/\alpha} \tag{8.5a}$$

and

$$d_j < \left(\frac{d_i^{\alpha}}{\beta' + \left(\frac{d_i}{R}\right)^{\alpha} \left(\frac{N'_0 + MAI}{N'_0}\right)} \right)^{1/\alpha}. \tag{8.5b}$$

The inequality in (8.5a) provides an upper bound on d_i; however, a lower bound on d_j may be obtained from the same inequality. Likewise, the inequality (8.5b) provides an upper bound on d_j, as well as a lower bound on d_i. Rearranging terms yields

$$d_i > \left(\frac{\beta' d_j^{\alpha}}{1 - \left(\frac{d_j}{R}\right)^{\alpha} \left(\frac{N'_0 + MAI}{N'_0}\right)} \right)^{1/\alpha} \tag{8.6a}$$

and

$$d_j > \left(\frac{\beta' d_i^{\alpha}}{1 - \left(\frac{d_i}{R}\right)^{\alpha} \left(\frac{N'_0 + MAI}{N'_0}\right)} \right)^{1/\alpha}. \tag{8.6b}$$

In summary, for simultaneous reception from transmitters *i* and *j*, d_i must satisfy $A(d_j) < d_i < B(d_j)$, where

$$A(d_j) = d_j \left(\beta'\right)^{1/\alpha} \left(1 - \left(\frac{d_j}{R}\right)^{\alpha} \left(\frac{N_0' + MAI}{N_0'}\right)\right)^{-1/\alpha} \tag{8.7a}$$

and

$$B(d_j) = d_j \left(\beta' + \left(\frac{d_j}{R}\right)^{\alpha} \left(\frac{N_0' + MAI}{N_0'}\right)\right)^{-1/\alpha}. \tag{8.7b}$$

Both $A(d_j)$ and $B(d_j)$ go to 0 as d_j goes to 0. However, as d_j increases, $A(d_j)$ becomes larger than $B(d_j)$, and reception of both transmissions becomes impossible. The threshold distance, *d*, for which MPR becomes impossible in the two transmitter example is

$$d = R \left(\frac{1 - \left(\beta'\right)^2}{\left(1 + \beta'\right)\left(\frac{N_0' + MAI}{N_0'}\right)} \right)^{1/\alpha}. \tag{8.8}$$

If there is no interference from other users (i.e., MAI=0), (8.7a) reduces to

$$A(d_j) = \left(\frac{\beta'}{d_j^{-\alpha} - R^{-\alpha}} \right)^{1/\alpha}, \tag{8.9a}$$

and (8.7b) becomes

$$B(d_j) = \left(\frac{1}{\beta' d_j^{-\alpha} + R^{-\alpha}} \right)^{1/\alpha}. \tag{8.9b}$$

Given a value for the distance of transmitter *i* to the receiver, d_i, the bounds defined in (8.9a) and (8.9b) are plotted in Figure 28 for *R*=100 and values of Beta $\equiv \beta' = \{0.1, 0.25, 0.4, 0.55\}$. The horizontal axis represents the distance d_i, while the vertical axis represents the distance d_j. For example, if Beta is 0.25 and d_i is 60 units, then MPR is feasible if d_j is between 45 and 75 units, assuming there are no other sources of MAI and an identical path loss exponent for each link. For a given value of Beta, the threshold distance identified in (8.8) for which MPR becomes impossible is the point in Figure 28 where the functions $A(d_j)$ and $B(d_j)$ intersect. For example, if Beta=0.55, then $d \approx 80$.

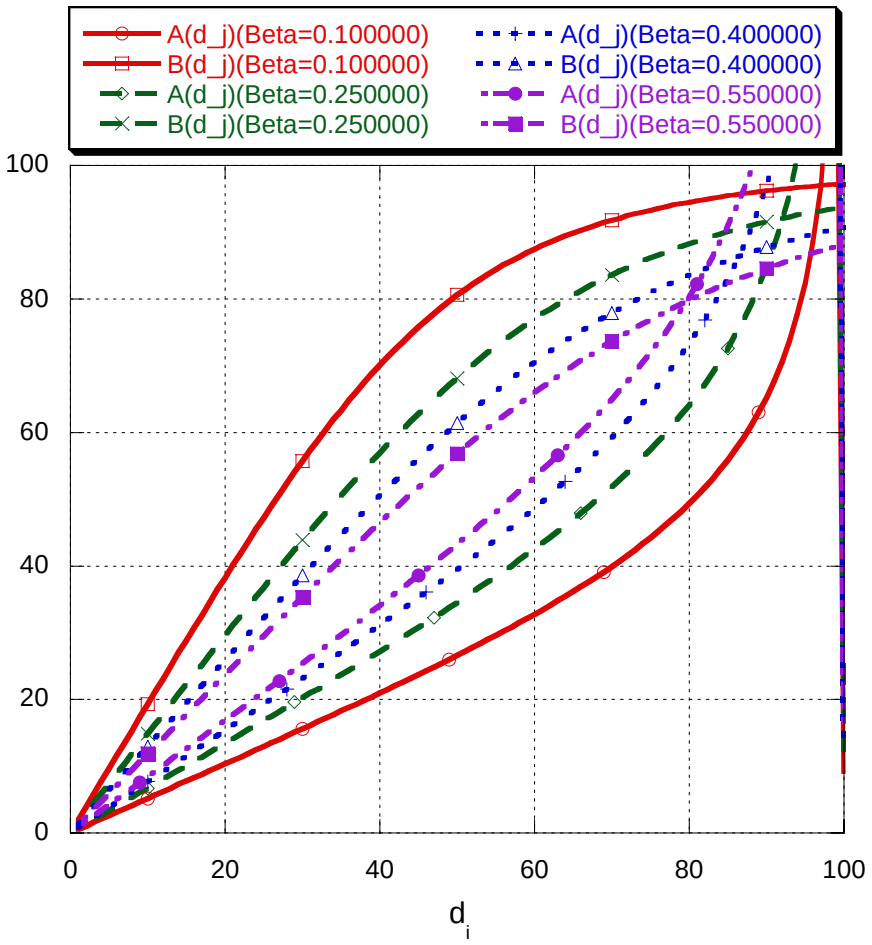


Figure 28. Minimum and maximum transmission ranges for reception from two terminals given a fixed transmission range.

As shown in (8.7a) and (8.7b), MPR is feasible when transmitters *i* and *j* are placed so that $A(d_j) < d_i < B(d_j)$. However, in a mobile ad hoc network terminals may be in random locations. For a random placement of terminals, we define d_j^* as the value of d_j which maximizes the probability of a second terminal, *i*, being located so that $A(d_j) < d_i < B(d_j)$. In particular, given a uniform spatial distribution of terminals, the expected number of terminals in the disc defined by $\left(A(d_j), B(d_j)\right)$ is proportional to $\pi\left(B^2(d_j) - A^2(d_j)\right)$. Under this spatial distribution, we have

$$d_j^* = \max_{d_i} \int_{A(d_i)}^{B(d_i)} 2\pi r dr = R\left(\frac{\hat{\beta}+1}{\hat{\beta}-\beta'}\right)^{-\frac{1}{\alpha}}, \tag{8.10}$$

where

$$\hat{\beta} = \left(\beta'\right)^{\left(\frac{\alpha-2}{\alpha+2}\right)}. \tag{8.11}$$

This result is obtained by setting to the derivative, with respect to d_j, of $\pi\left(B^2(d_j) - A^2(d_j)\right)$ equal to 0 and solving for d_j.

Similar analysis on MPR feasibility may be done for reception of 3 or more transmitters; however, the set of feasible transmitter distances is considerably smaller and more difficult to graphically represent. However, given β', it is straightforward to determine how many transmissions may satisfy the SINR requirement at the receiver. If we assume *n* transmissions are successfully received under ideal conditions (equal received power from each transmission, neglible thermal noise, and no other sources of

interference besides the n transmitters) then the reception requirement for each transmitter is

$$\frac{P_t d^{-\alpha}}{(n-1)P_t d^{-\alpha}} = \frac{1}{n-1} > \beta' \tag{8.12}$$

Since n is an integer, we have

$$n \le \left\lfloor \frac{1}{\beta'} \right\rfloor + 1. \tag{8.13}$$

This final result is also given in [55] and [56]. Clearly, in the presence of noise, reception from more than one transmitter is feasible if and only if $\beta' < 1$.

8.2 Integrating MPR Capability with the INS Protocol

In this section we describe the operation of the INS protocol when terminals have MPR capability; this extended protocol is referred to as the INS-MPR protocol. We define a constraint, MPR_MAX, to represent the maximum number of transmissions a receiver may acquire in a slot. Since terminals may receive from multiple transmitting neighbors in each slot, the receive vector, $\mathbf{r}^{(i)}$, is replaced with a receive matrix $\mathbf{R}^{(i)}$. Entry $r_{j,s}^{i}$ of $\mathbf{R}^{(i)}$ is equal to 1 if terminal i attempts to receive a packet from neighbor j in slot s. At all times, each column s of $\mathbf{R}^{(i)}$ must satisfy

$$\sum_{\forall j} r_{j,s}^{i} \le \text{MPR\_MAX}. \tag{8.14}$$

Once a terminal selects MPR_MAX neighbors to monitor in slot s, no additional entries may be made in column s of $\mathbf{R}^{(i)}$ until an existing entry in column s expires.

Next, we design the INS-MPR protocol use a *pair-wise compatibility* requirement for the entries in column *s* of $\mathbf{R}^{(i)}$. The pair-wise compatibility requirement helps to ensure that if terminal *i* indicates (via NBR_ACK transmission) that it can receive from two neighbors, *j* and *k*, in slot *s*, then simultaneous transmissions from *j* and *k* to *i* in slot *s* will both exceed the required SINR thresholds, $\beta'_{j,i}$ and $\beta'_{k,i}$. Transmissions from two neighbors, *j* and *k*, in slot *s* are not pair-wise compatible if simultaneous transmission from *j* and *k* to *i* in slot *s* results in one or more failed transmissions due to insufficient SINR.

The rule for pair-wise compatibility is developed as follows. Given a terminal *i* in receive mode, suppose two nearby terminals, *j* and *k*, both transmit in slot *s* of the transmission frame. To simplify notation we define $N' \equiv \frac{N_0}{T_c} + \sum_{l \neq j,k} P_r(l)$ to represent the sum of the thermal noise and MAI at *i* in slot *s*, exclusive of the transmissions from *j* and *k*. For reception from *j* and *k*, it is required that

$$\frac{P_r(j)}{N' + P_r(k)} > \beta'_{j,i} \tag{8.15a}$$

and

$$\frac{P_r(k)}{N' + P_r(j)} > \beta'_{k,i}. \tag{8.15b}$$

(8.15a) and (8.15b) may be rewritten as

$$\frac{P_r(j)}{N'} > \beta'_{j,i}(1 + \frac{P_r(k)}{N'}) \tag{8.16a}$$

and

$$\frac{P_r(k)}{N'} > \beta'_{k,i}(1 + \frac{P_r(j)}{N'}). \tag{8.16b}$$

Next, because SINR estimates are made relative to $N_{\max}$, we have the following approximations:

$$\frac{P_r(j)}{N'} \cong \frac{\hat{\xi}_{j,i}}{N_{\max}} \tag{8.17a}$$

and

$$\frac{P_r(k)}{N'} \cong \frac{\hat{\xi}_{k,i}}{N_{\max}} \tag{8.17b}$$

The left-hand side of the inequalities in (8.16a) and (8.16b) are substituted by (8.17a) and (8.17b), respectively. The right-hand side of the inequalities in (8.16a) and (8.16b) are substituted by (8.17b) and (8.17a), respectively. Multiplying both sides by $N_{\max}$, the transmissions from *j* and *k* are *pair-wise compatible* when

$$\hat{\xi}_{j,i} > \beta'_{j,i}(N_{\max} + \hat{\xi}_{k,i}) \tag{8.18a}$$

and

$$\hat{\xi}_{k,i} > \beta'_{k,i}(N_{\max} + \hat{\xi}_{j,i}) . \tag{8.18b}$$

When MPR_MAX>2, pair-wise compatibility is no longer a sufficient predictor of the success or failure of a particular set of transmissions. For example, if there are three non-zero entries in column *t* of $\mathbf{R}^{(i)}$, then any two neighbors scheduled to transmit in the slot may transmit successfully to *i*, but transmission failure occurs when all three transmit at the same time. However, one feature of both the INS and the INS-MPR protocol is that terminals scheduled to transmit in the same slot are generally well-separated; if they are close enough to receive each other's FLAG transmissions, then they transmit in separate slots so that they may receive each other's transmissions. It is

uncommon for a terminal to have more than two neighbors which transmit in the same slot which are all pair-wise compatible. Even in this case, three or more would have to actually be transmitting in the slot for a failure to occur.

The INS-MPR protocol is designed to generate the same level of network control overhead as the INS protocol defined in Chapter 5. For the INS protocol defined in Chapter 5, the only control packets generated are NBR_ACK packets. NBR_ACK packets are generated periodically. Additional NBR_ACK packets are generated when a neighbor *j* is detected which causes an entry in the receive *vector*, $\mathbf{r}^{(i)}$, to change from 0 to *j*. This allows fast feedback to neighbors so that new communications links may be established, and so that communications links which have timed out may be recovered. For the INS-MPR protocol, NBR_ACK packets are also generated periodically. However, additional NBR_ACK packets are generated only if a terminal *i* receives a packet from *j* in a slot *s* for which there is no *x* such that $r_{x,s}^{i} = 1$ (i.e., the additional NBR_ACK packets are generated only for the first transmitter detected in a slot). When additional neighbors are detected in a slot, they are acknowledged through the periodically generated NBR_ACK packets.

8.3 Evaluation of the INS Protocol with Modifications for MPR Capability

In this section we evaluate the performance of the INS-MPR protocol, using the INS protocol described in Chapter 5 as a performance benchmark. Selective collision elimination is used in neither the INS nor the INS-MPR protocol. In simulations of the INS-MPR protocol, MPR_MAX is 2.

For results reported in previous chapters, $\beta = 8$ and $N_{\max} = 32$; hence, $\beta' = \frac{\beta}{N_{\max}} = 0.25$. In practice, the value of β' may be reduced through the use of larger maximum spreading factor, lower code rate, lower order modulation, or improved receiver design. In these systems the potential influence of MPR increases. Consider Figure 28: as Beta decreases, the "eye" opens and MPR is feasible for more transmission scenarios. One may also consider (8.18a) and (8.18b): for smaller values of $\beta'_{j,i}$ and $\beta'_{k,i}$, the pair-wise compatibility requirement is more easily satisfied. Consequently, we explore the performance of the INS-MPR protocol for $\beta' = \left\{0.25, 0.125, 0.08\overline{3}\right\}$. In the simulations, $\beta' = 0.125$ is achieved by setting $N_{\max} = 64$, and $\beta' = 0.08\overline{3}$ is achieved by setting $N_{\max} = 96$. Larger values of $N_{\max}$ change the spreading factors which may be used by the adaptive transmission protocol. When $N_{\max} = 64$, the adaptive transmission protocol may set the spreading factor to 64, 32, or 16. When $N_{\max} = 96$, the adaptive transmission protocol may set the spreading factor to 96, 48, or 24.

The packet generation rate is measured in packets per slot. For results with *R*=200m and *R*=250m, results are shown for packet generation rates up to 1.4 packets per slot. For results with *R*=350m, the horizontal axis extends to packet generation rates of up to 2 packets per slot because these networks are able to support greater end-to-end packet completion rates.

In Figure 29, the average packet completion rate is shown for the INS-MPR protocol for *R*=200m, *R*=250m and *R*=350m; results for the INS protocol and the BTS are

also shown for comparison. Note that there is no reason to examine the performance of the BTS with MPR extensions because the BTS ensures that there is only one transmitter within range of a receiver. At *R*=200m, the INS-MPR protocol shows significantly better performance when compared to the INS protocol. The reason for this is that more terminals are able to successfully receive each transmission. To better illustrate this point, we measure the average number of receivers successfully decoding each control packet (i.e., the NBR_ACK packets) transmitted; these packets are used for measurements since they are transmitted using the maximum spreading factor. For $\gamma = 0.6$ packets per slot, R=200m, and $N_{\max} = 32$, the average number of terminals successfully receiving each NBR_ACK packet is approximately 9.7 for the BTS, 7.5 if the INS protocol is used, and 8 if the INS-MPR protocol is used. Since more terminals are able to receive each transmission, there are more communications links and the connectivity of the network improves. It is also noteworthy that the performance improvement for the INS-MPR protocol does not require additional communication overhead from BLOCK packets. Consequently, for higher packet generation rates, the INS-MPR protocol performs at least as well as the INS protocol.

When *R*=250m and *R*=350m, INS-MPR provides smaller performance gains when compared to the INS protocol. For *R*=250m, if there is a 90% packet completion rate requirement, the INS-MPR protocol supports a packet completion rate of $\gamma = 1.1$, while the INS protocol only satisfies this requirement up to $\gamma = 0.95$; the performance gain from MPR is approximately 16% in this case. At *R*=350m, there is almost no perceptible difference in performance between the two approaches. Receiver availability

is not a concern in very dense networks since terminals have a large number of neighbors to which they may transmit. Allowing terminals to increase the number of communicable neighbors, through MPR capability, is rarely helpful in these networks.

In addition, the adaptive transmission protocol precludes gains from MPR. When N_{max}=32, the available spreading factors for a transmission are 32, 16, and 8. At R=350m, approximately 80% of all data packet transmissions use the smallest available spreading factor, 8, as a result of the link cost metric (this is true in both the INS and the INS-MPR protocols). When a spreading factor of 8 is used on a link (j, i), $\beta'_{j,i} = 1$ and MPR is not feasible.

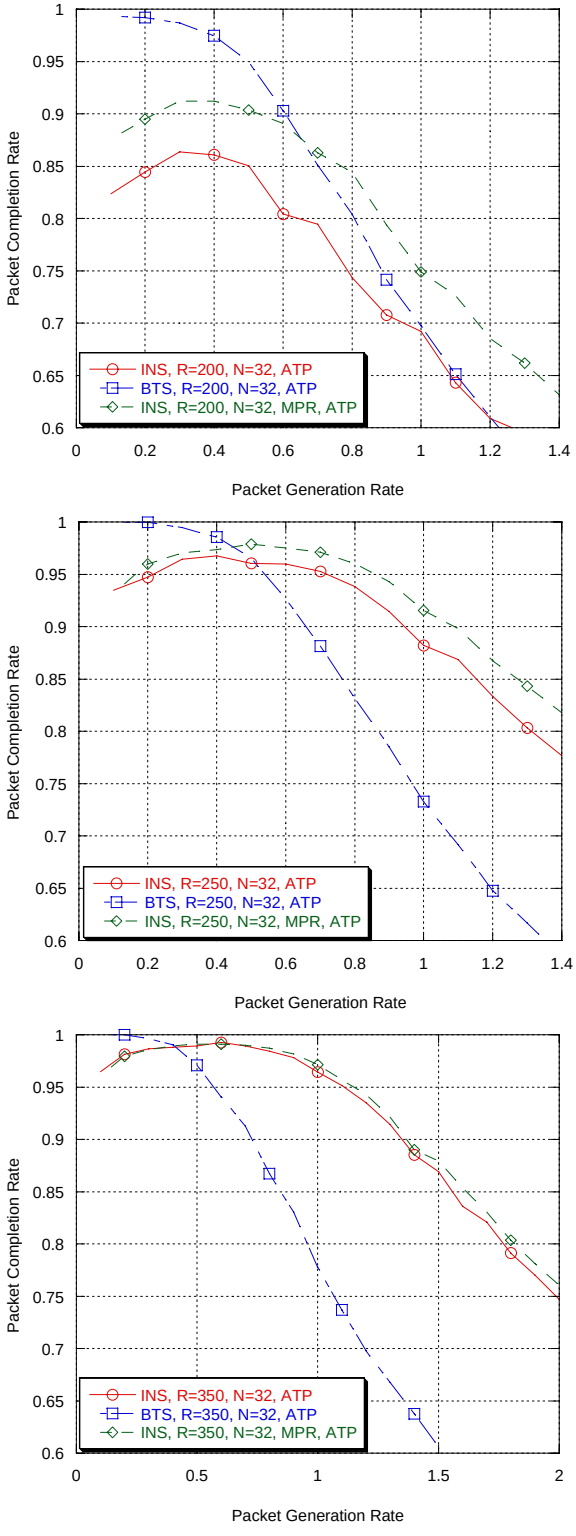


Figure 29. Performance of the INS-MPR protocol for N_{max} = 32.

We next consider systems that have a smaller value of β' because $N_{\max}$ is increased. In these systems, the performance of both the INS protocol and the INS-MPR protocol is improved. For example, if $N_{\max}$ is 64 as in Figure 30, when *R*=200m, the INS-MPR protocol provides a 90% packet completion rate up to a generation rate of almost 0.75, while the BTS only provides this up to approximately 0.65. If $N_{\max}$ is 96 as in Figure 31, when *R*=200m, the INS-MPR protocol provides a 90% packet completion rate up to a generation rate of almost 0.85. The performance of the BTS does not change significantly when β' is decreased. At higher values of *R* (250m and 350m), MPR provides additional gains in network performance when the system uses a larger spreading factor. In these systems, MPR is feasible even when the smallest spreading factor is used for transmissions (the smallest available spreading factor is 16 when $N_{\max} = 64$, and 24 when $N_{\max} = 96$). In systems for which $N_{\max} = 96$ and *R*=250m, the INS-MPR protocol provides a 90% packet completion rate up to $\gamma = 1.4$, while the INS protocol only supports this packet completion rate up to $\gamma = 1.15$; this represents a performance improvement of approximately 20%. When $N_{\max} = 96$ and *R*=350m, the performance improvement is approximately 10% (1.45 for the INS protocol to 1.6 for the INS-MPR protocol).

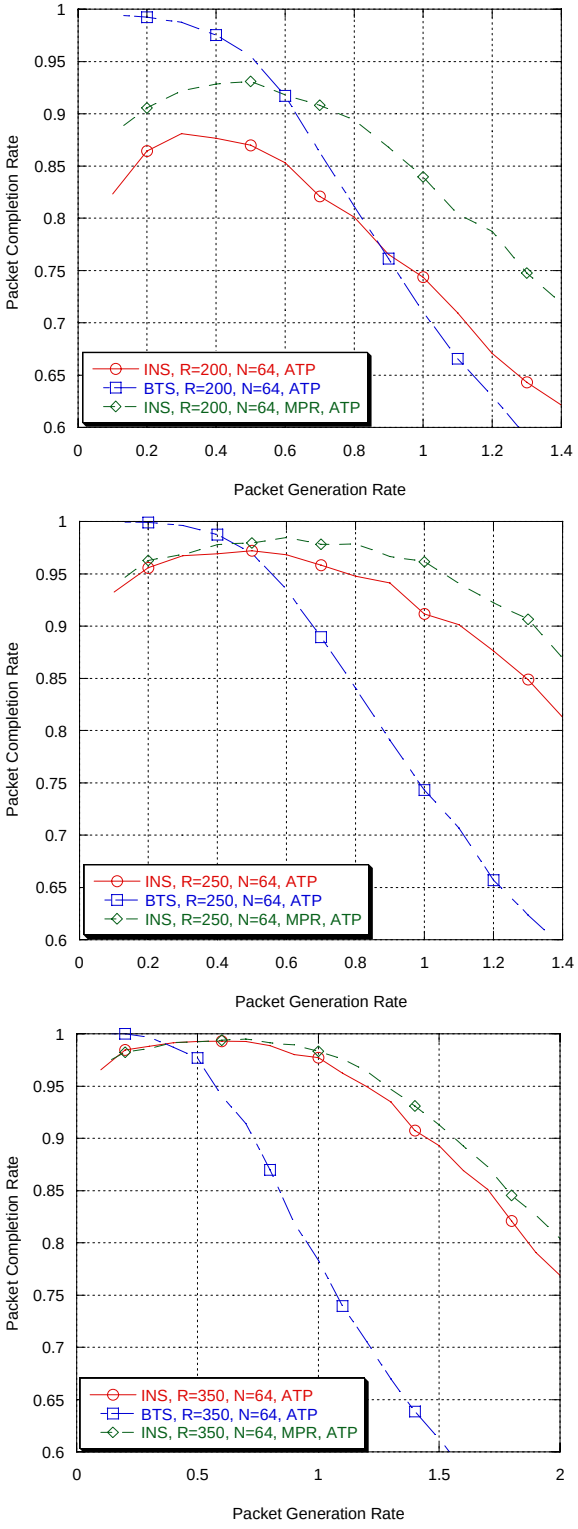


Figure 30. Performance of the INS-MPR protocol for N_{max} = 64.

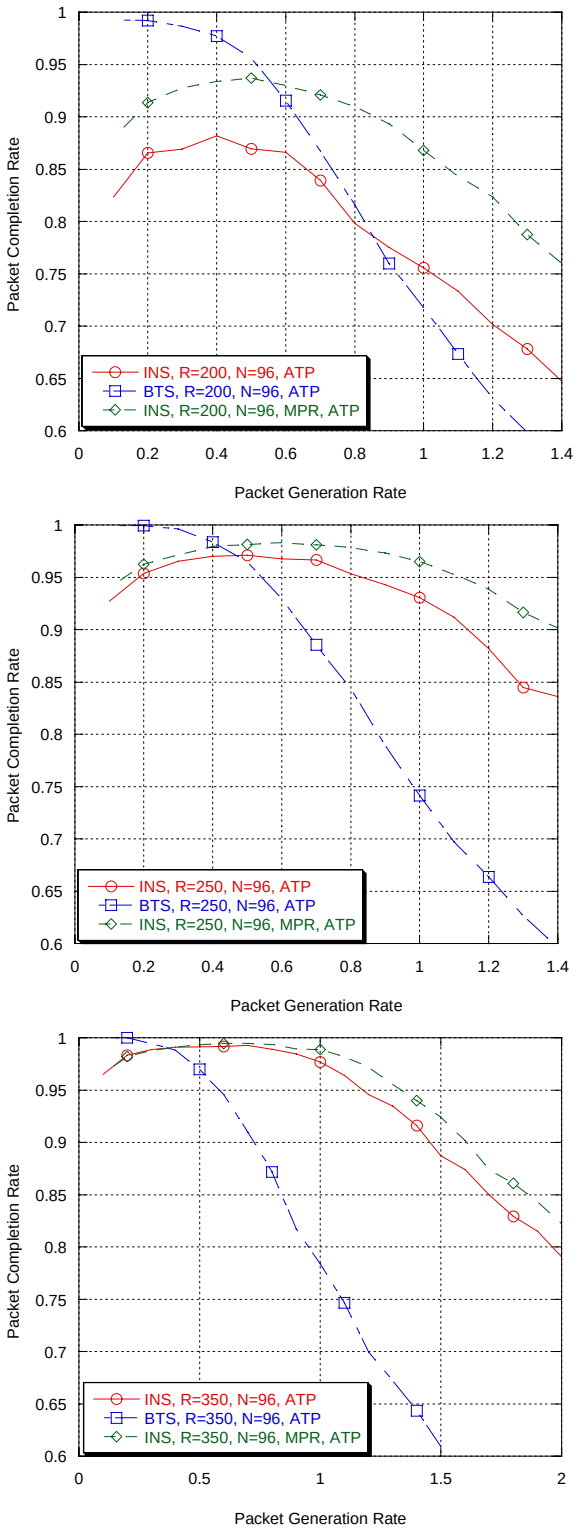


Figure 31. Performance of the INS-MPR protocol for N_{max} = 96.

Based on these results, MPR significantly improves the performance of the INS protocol in sparse networks due to a greater number of communicable neighbors. This improvement does not require the introduction of additional communication overhead, such as BLOCK packets. As a result, in networks with high traffic load, the performance of the INS-MPR protocol is much better than the performance of the INS protocol with selective collision elimination. In all networks, performance gains due to MPR increase as β' is reduced; this is because a greater number of links may be activated at higher data rates.

CHAPTER NINE

ESTIMATING THROUGHPUT CAPACITY

Determining the throughput capacity of a network is a challenging problem, but results on throughput capacity provide a useful benchmark for the performance of distributed scheduling protocols, such as the INS. These studies also provide insight as to the extent to which various features, such as MPR, may improve network throughput. For example, capacity scaling results from [57] suggest that MPR has the potential to significantly improve the asymptotic throughput capacity of a wireless network. In this chapter, we estimate the throughput capacity of wireless networks using a centralized transmission packing algorithm. Centralized algorithms provide a tractable solution to the problem of estimating the throughput capacity of large networks (see [58] and [59] for further discussion). Optimization algorithms, such as those based upon linear programming (e.g., [60]), may be used to determine the optimal resource allocation which achieves the throughput capacity of a given network under certain assumptions; however, these algorithms are only applicable in small networks due to their complexity.

In the next section, we provide a brief survey of the literature to provide background on how MPR has been used to provide enhanced performance. In Section 9.2, we analyze the throughput capacity of a small network to show how MPR, as we have modeled it, may lead to improved network throughput when compared to a network in which terminals are capable of only single-packet reception. In Section 9.3, we describe the centralized algorithm used to generate estimates of throughput capacity. In Section 9.4, results obtained from the centralized algorithm are used to compare the

performance of networks with MPR capability and the performance of networks with only single-packet reception capability. We also examine the influence of the adaptive transmission protocol used by the INS protocol when used with the centralized algorithm.

9.1 Survey of Results on Multi-packet Reception

The model for multi-packet reception we describe in Chapter 8 requires that signals from multiple transmitters satisfy an SINR threshold requirement at the receiving terminal. This model is also used in [56] to improve the performance of single-hop networks with MPR by modifying a standard CSMA/CA backoff mechanism.

There are several other models for MPR which are commonly used in the literature. One approach uses a *capture matrix* to model MPR capability. For a given capture matrix **C**, element $C_{n,k}$ represents the probability that k packets are received by a single terminal when n transmitters are within communications range. In [57] the authors employ a capture matrix for which $C_{n,k}=1$ when $n=k$; using this model, they show that the order capacity of a wireless network is $\mathrm{O}\left(\sqrt{\frac{\log(N)}{N}}\right)$ when terminals are capable of MPR. This is a substantial improvement over the capacity predicted by [53] and [61], where the order capacity without MPR is shown to be $\mathrm{O}\left(\frac{1}{\sqrt{N}}\right)$. Other approaches which model MPR using a capture matrix are as follows. The early work of [62] demonstrated that MPR capability may stabilize the throughput of slotted ALOHA. In [63], MPR is used in conjunction with a hybrid scheduling/contention-based MAC protocol to improve the throughput of Manhattan networks. The drawback to modeling MPR using a capture

matrix is that it does not account for the noise power at a receiver or the distribution of received signal powers.

In [64], the authors design a receiver-initiated MAC scheduling protocol for receivers capable of multi-user detection (MUD). As in Chapter 8, signals from each transmitter must exceed an SINR threshold to be successfully received. Unlike the model in Chapter 8, these receivers are able to mitigate MAI from other users to further increase opportunities for MPR. In addition, the receiver-initiated MAC protocol described in [64] does not allow for broadcast transmissions to many neighbors at once, which is a key goal in this work. In [65], [66], and [67], MPR is achieved through multi-user detection and interference cancellation. This requires accurate estimation of channel parameters, as well as complex receiver design which allows received signals to be processed in multiple stages. The model for MPR we describe in Chapter 8 does not require MUD or successive interference cancellation.

9.2 Performance of an Example Network with MPR

In this section we once again consider a six-terminal network, similar to the network from Section 5.3 where terminals are labeled **A** through **F**. We use this network to compare the optimal performance in the case of single-packet reception to the optimal performance in the case of MPR. In this example, terminals are spaced 120m apart, as shown in Figure 32. The transmit power is set so the communications range, R, is 200m. As a result, each terminal may only communicate directly with its nearest neighbors. Simulations are performed to compare the performance of the network under single

packet reception and multi-packet reception. The spreading factor for all transmission is 64, resulting in an SINR threshold $\beta' = 0.125$.

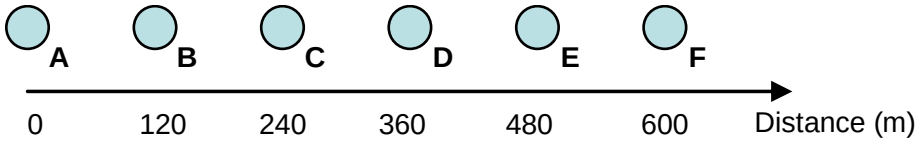


Figure 32. Physical locations of terminals for six-terminal example.

We consider two scenarios, one corresponding to the case of terminals which are not capable of MPR, and one corresponding to the case of terminals which are capable of MPR. In the former case, a three slot transmission frame is required so that terminals are able to transmit to and receive from each neighbor in each transmission frame. This results in a slot assignment of {1,2,3,1,2,3} for terminals {**A**,**B**,**C**,**D**,**E**,**F**}, respectively (note that Lyui's slot assignment algorithm is not being used in this example). In the latter case, terminals with MPR capability are able to receive from two transmitting neighbors simultaneously. The slot assignment {1,2,1,2,1,2} for terminals {**A**,**B**,**C**,**D**,**E**,**F**}, respectively, enables them to alternate between transmitting in one slot and receiving from 0,1, or 2 neighbors in the next slot.

Each terminal generates unicast traffic at equal rate for all possible destinations. In Figure 33, Figure 34, and Figure 33, we show the end-to-end throughput, delay, and completion rate, respectively, for both scenarios. For this test, results are averages over 10 random packet generation scenarios applied to the topology in Figure 32. Without MPR, stable throughput can only be supported up to a generation rate of approximately 0.55 packets per slot. With MPR, throughput is stable for packet generation rates up to

approximately 0.8 packets per slot. A similar comparison may be made by examining the packet completion rate; a 90% completion rate corresponds to a packet generation rate of 0.61 packets per slot with single packet reception and 0.92 packets per slot with MPR. If there is a delay requirement of 10 slots or less, then MPR allows a nearly two-fold increase in supported packet generation rate, from 0.4 to 0.75 packets per slot.

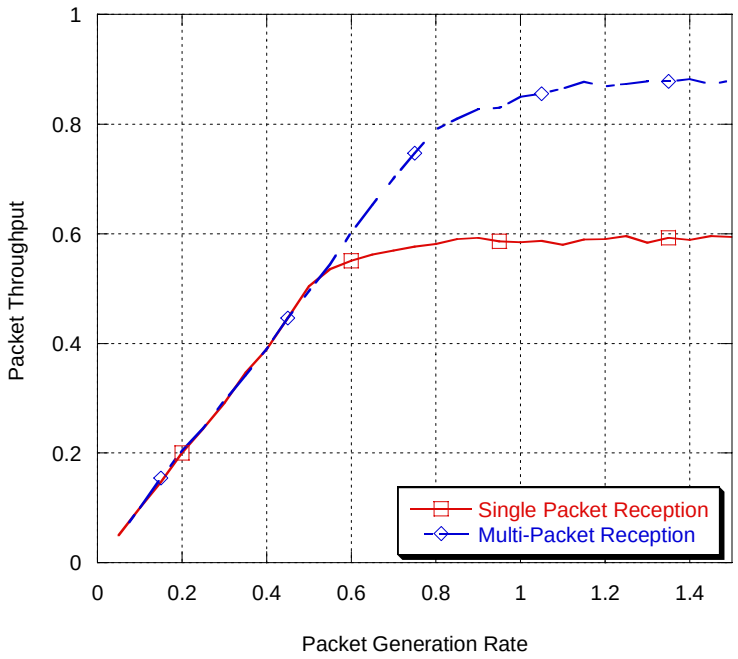


Figure 33. End-to-end packet throughput for single-packet reception and multi-packet reception.

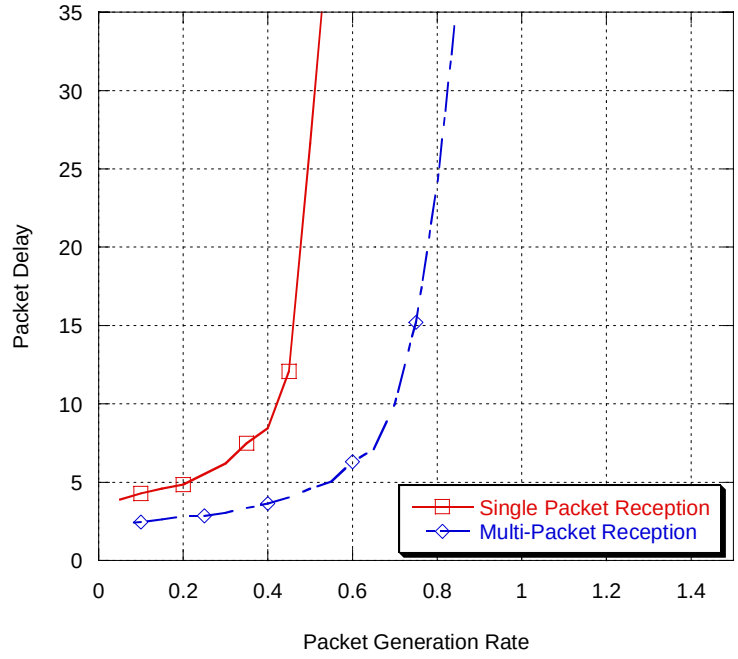


Figure 34. End-to-end packet delay for single-packet reception and multi-packet reception.

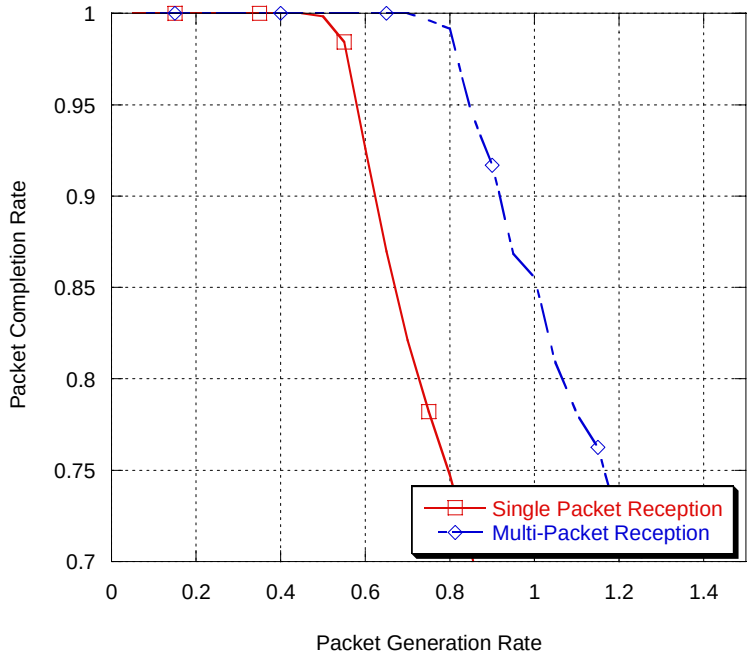


Figure 35. End-to-end packet completion rate for single-packet reception and multi-packet reception.

For more complex network topologies similar performance gains are much more difficult to achieve. There are several reasons for this. First, terminals are not spaced at regular intervals; hence, a receiver within range of two transmitters may suffer from the *near-far* effect, preventing reception of more than one packet. In addition, when there are a large number of unicast traffic pairs, and many available routing options for each traffic flow, determining the optimal scheduling and routing to take advantage of MPR capability is prohibitively complex! In the six-terminal network, there is only one route for each traffic flow, and the optimal schedule may be found via the techniques in [25].

To summarize the process in [25], one may compute the traffic load for each terminal based upon the number of flows for which the terminal must forward traffic. The *load factor* for a terminal is the traffic load of a terminal divided by the fraction of bandwidth assigned to it. Next, examine groups of terminals which must transmit separately, given the requirement that transmissions must be successful. The grouping with the largest summed traffic load is the grouping which limits the maximum stable throughput of the network. For an optimal schedule, it is necessary and sufficient that the full bandwidth be assigned to such a grouping, while remaining terminals are assigned bandwidth such that they have a smaller load factor than terminals in the grouping. For single packet reception, the limiting grouping is {2,3,4} (or {3,4,5} by symmetry). The maximum stable throughput is 0.64 packets per slot; this may be obtained with a 47 slot schedule. The schedule used in the simulation supports a maximum stable throughput of only 0.59 packets per slot, but it is much easier to implement. For the case of multi-packet reception, the limiting grouping is {3,4}. The maximum stable throughput is 0.88

packets per slot; this is realized with the 2 slot schedule used in simulations, representing a 37.5% improvement over the maximum stable throughput of single packet reception. Lastly, terminals may only enqueue up to 20 packets at a time. The variance of the random packet generation causes packet queues to begin dropping packets at input values that are less than the maximum stable throughput. As a result, one would not expect to see, for example, a 100% completion rate when MPR is used and the packet generation rate is 0.87.

9.3 Centralized Transmission Scheduling Algorithm

We use a centralized transmission scheduling algorithm to estimate the throughput capacity of the random networks used in tests of the INS protocol. This algorithm uses global knowledge of link gains, MAI, packet queues, and terminal activity to determine a maximal transmitter configuration in each slot. The centralized algorithm resembles an idealized CDMA-based MAC protocol from [68] in that it considers transmitters for activation in a serial fashion. The algorithm in [68] is also used as a point of comparison in [46], where it is referred to as *serialized contention resolution*. In the latter paper, it is assumed that traffic demands are persistent, i.e. terminals always have a packet to send, and the evaluation is based on the single-hop throughput attained by the scheme. Under these conditions, serialized contention resolution is Pareto optimal since throughput depends only on the size of the active transmitter set, and the final transmitter set cannot be augmented without disrupting an existing transmission.

In this work, we do not assume persistent traffic demands, and we evaluate end-to-end performance. We also introduce a fairness constraint, in the form of a minimum

transmitter set, to ensure bounded delay for each packet. Specifically, in each slot the minimum transmitter set consists of a single terminal, and terminals are placed into this set in a round-robin fashion from one slot to the next. Thus, in a network of N terminals, each terminal is given first consideration for transmission once every N time slots. The order of the remaining terminals is a randomized permutation with a uniform distribution. Given this ordering and the initial transmitter set, the algorithm in Figure 36 is executed to determine the transmitter configuration for the current slot, s. Candidate links are considered for activation based upon the ordering of the transmitters and current state of their packet queues. The qualifications for activating a candidate link (i,j) in slot s are:

- Terminal i must have one or more packets enqueued for transmission
- Terminal i must not be scheduled to receive a packet
- For some enqueued packet p, p's next hop j must be in receive mode
- The SINR requirement for each active link, including (i,j), must be satisfied
- The transmission from i to j must be *feasible* in the sense that it does not violate MPR constraints or constraints associated with the adaptive transmission protocol. The conditions for feasibility are explained in the following paragraphs.

If all conditions are met, then i transmits packet p in slot s.

The feasibility constraint related to MPR is expressed as a limit on the number of transmitters which may be associated with a receiver j in time slot s:

$$\sum_{i=1}^{N} l_{i,j}^{s} \leq \text{MPR\_MAX}. \tag{9.1}$$

The parameter MPR_MAX defines the maximum number of transmitters from which a terminal may attempt to receive a packet. In this chapter, we set MPR_MAX to 4 for scenarios in which terminals have MPR capability; otherwise, MPR_MAX is set to 1 so that only single-packet reception is possible. The upper bound determined via (8.12) is of little use determining a value of MPR_MAX to use since it does not account for thermal noise and MAI. In practice, instances for which a terminal receives from four distinct transmitters are exceedingly rare (<0.03% in all cases we consider). The complexity of determining a valid transmitter configuration grows with MPR_MAX, so it is desirable to use a small value.

The feasibility constraint related to the adaptive transmission protocol is expressed as a limit on the sum of the time slot fractions occupied by packets *p* selected by terminal *i* for transmission to neighboring terminals *j* in time slot *s*:

$$\sum_{\substack{\forall p \\ j=p->next\_hop}} \frac{N_{i,j}}{N_{\max}} \leq 1. \tag{9.2}$$

When the adaptive transmission protocol is used, the spreading factor used for a link (i, j) depends upon the link SINR estimate, $\hat{\hat{\xi}}_{i,j}$, which is stored at terminal *i*. The centralized algorithm provides these samples directly to the terminals. All other details are identical to the description provided in Section 5.4. If the ATP is not used, then all packets are transmitted with the maximum spreading factor, $N_{\max}$.

The procedure *feasibleTransmission*() in Figure 36 uses (9.1) and (9.2) to determine if packet p may be transmitted in slot s.

Several features of the centralized algorithm lead to its excellent performance. First, the algorithm achieves a maximal transmitter set in each slot; if a terminal is not transmitting, then at least one of the following holds true: it is receiving a packet, it does not have a packet it can transmit, or it cannot transmit without causing another transmission to fail. Second, the centralized algorithm is sensitive to the traffic load at each terminal induced by the routing algorithm. If a terminal is required to forward packets for only a few source/destination pairs, then it refrains from transmitting when its queue is empty, allowing terminals with large traffic demands to access the channel more frequently. Lastly, the centralized algorithm described in this section does not schedule broadcast transmissions. Instead, the algorithm activates individual links at each step. This provides more flexibility in construction of the transmitter sets, and better performance for the unicast data traffic we consider.

Algorithm: Centralized Transmission Packing:

Input: Time slot *s*

Initial feasible transmitter set T_s created according to the fair use policy for slot *s*.

Initial receiver set $R_s = \{\varnothing\}$, Initial link matrix $\mathbf{L}_s = \{0\}$.

Output: Augmented set T_s listing terminals transmitting in slot *s*.

Receiver set R_s containing terminals receiving in slot *s*.

Link activation matrix $\mathbf{L}_s$ with entries $l_{i,j}^s = 1$ if link (*i*,*j*) is active in slot *s*, 0 otherwise.

//Create seed scenario for slot *s*

For each terminal $i \in T_s$

For each packet p enqueued at *i*

If (!*feasibleTransmission*(*p*)) next;

$j = p->next\_hop$;

$l_{i,j}^s = 1$; $R_s = R_s \bigcup j$; $i->Xmt(p)$;

End For

End For

// Pack transmissions according to feasibility

For each terminal $i \notin T_s \bigcup R_s$

For each packet p enqueued at *i*

If (!*feasibleTransmission*(*p*)) next;

$j = p->next\_hop$;

$l_{i,j}^s = 1$;

If (*feasibleLinkMatrix*($\mathbf{L}_s$))

$T_s = T_s \bigcup i$; $R_s = R_s \bigcup j$; $i->Xmt(p)$;

Else

$l_{i,j}^s = 0$;

End If

End For

End For

feasibleLinkMatrix($\mathbf{L}_s$)

For each element $l_{i,j}^s \neq 0$

If ($\xi_{i,j} \leq \beta$) **return** false;

End For

return true;

Figure 36. Centralized transmission scheduling algorithm used to estimate throughput capacity.

9.4 Estimating Throughput Capacity for Random Networks

In this section, we use the centralized algorithm in Figure 36 to estimate the throughput capacity of the random networks used in previous simulations, as well as to study the extent to which MPR affects throughput capacity. Results are presented from three scenarios. For the base scenario, only single-packet reception is allowed (MPR_MAX is 1) and terminals use maximum spreading factor, $N_{\max}$, for all transmissions. In another scenario, identified with the 'MPR' label, MPR_MAX is 4 so that terminals are able to receive from up to four transmitters in each slot. In the final scenario, MPR_MAX is 4 and terminals use SINR estimates to adapt the spreading factor for each transmission in the manner described in Section 5.4. The 'MPR, ATP' label is applied to these scenarios.

Apart from the centralized algorithm, which constructs the set of transmitters in each slot, the simulations in this section are identical to the simulations in previous sections. The simulations use the same channel and receiver model as in previous simulations. Traffic generation and routing is also identical. Any neighbor within communications range is considered as a communicable neighbor for routing packets, but the routing metric applies high cost to links which consistently have low SINR. Terminals generate periodic control packets according to the rules for ACK packets used by the INS protocol; thus, control overhead is approximately the same. NBR_ACK packets do not have a specific destination, so it is not required that they be successfully received by any neighbor (requiring them to be received by all neighbors within the communications range would severely limit spatial reuse). Lastly, terminals in receive

mode are automatically configured to receive a packet which is transmitted to them; the SINR requirement must be satisfied for successful reception, but the centralized algorithm guarantees that this is always the case.

In Figure 37, the top graph shows the end-to-end packet completion rate when $\beta' = 0.25$ ($N_{\max} = 32$) and the communications range, *R*, is 200m or 250m. The bottom graph shows the same metric for dense network topologies with *R* set to 300m or 350m. When *R*=200m, the packet completion rate is always less than 1 since there are a few instances in which one or more terminals are isolated from the rest of the network; hence, the packets they generate for destinations for which there is no valid route are dropped. This does not occur when *R*>200m, and the performance improves. There is only a very slight advantage to using MPR in these networks, and there is no perceptible advantage to using MPR in networks with higher values of *R*.

The use of the adaptive transmission protocol is a detriment to network performance. The reason for this is that terminals use a reduced spreading factor on links with high SINR, regardless of whether or not they have multiple packets which can be transmitted. Using a reduced spreading factor effectively lowers the link SNR. As a result, the link can tolerate less MAI and it is more difficult for the centralized algorithm to activate additional transmitters. The INS protocol, on the other hand, produces a set of transmitters in each slot which generally remains the same from one transmission frame to the next; the ATP improves performance by increasing the transmission rate of certain transmitters in that set.

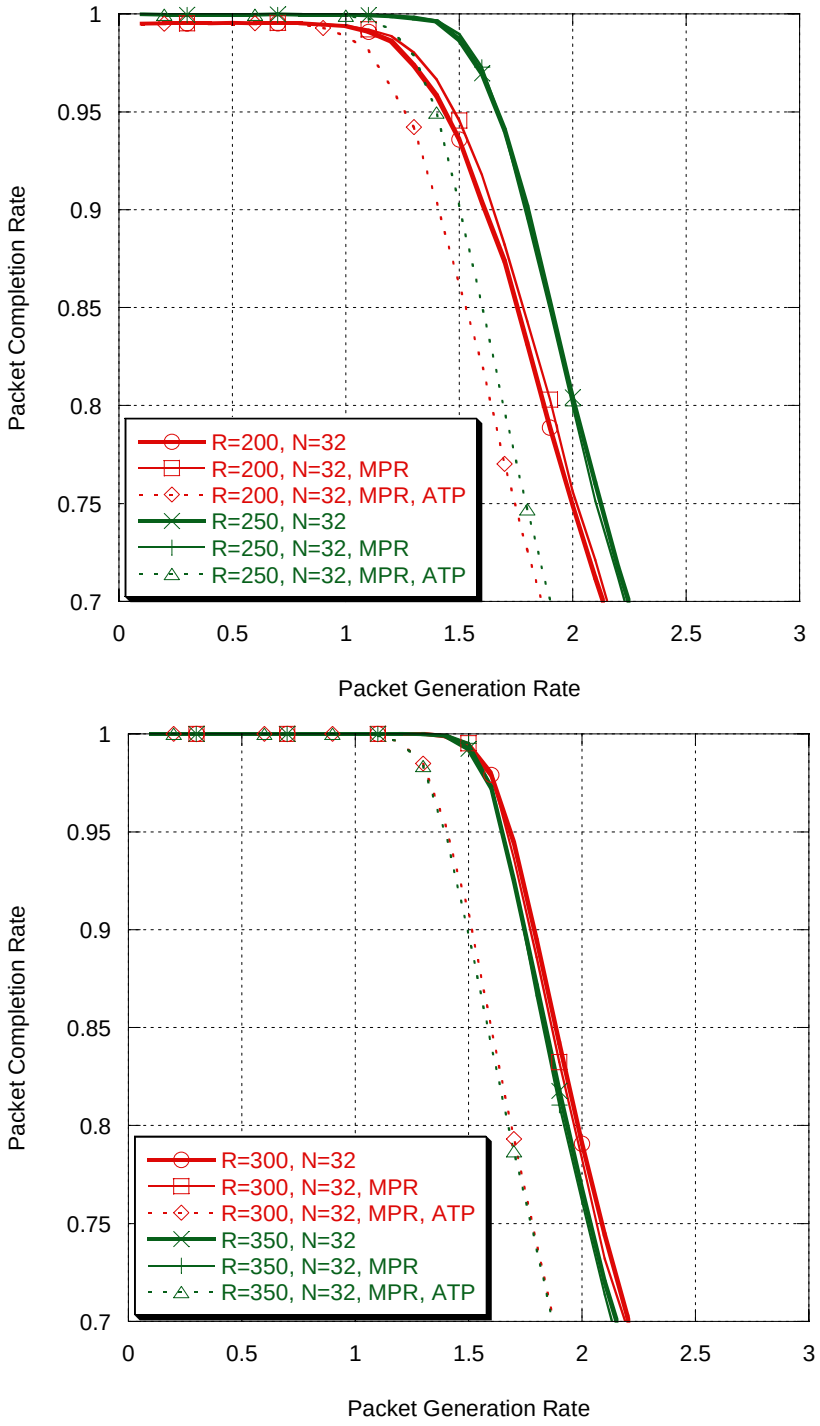


Figure 37. Packet completion rate for networks using a maximum spreading factor set to 32.

Figure 38 shows the end-to-end packet completion rate in networks for which $\beta' = 0.125$ ($N_{\max} = 64$). When R=200m and R=250m, there is a performance improvement of approximately 8% and 4.5%, respectively, at a 90% completion rate when terminals have MPR capability. The performance of dense networks (R=300m and R=350m) is not significantly improved with MPR capability even with a larger maximum spreading factor. This is likely due to the fact that terminals have more choices for forwarding packets when routes are created. In networks with sparse topology, a single terminal providing connectivity between two sections of the network is a frequent occurrence; in this situation, MPR capability allows the terminal to act essentially as two receivers. In a dense network topology, on the other hand, this type of situation is less likely.

Figure 39 shows the end-to-end packet completion rate in networks for which $\beta' = 0.08\overline{3}$ ($N_{\max} = 96$). For R=200m and R=250m, if a 90% packet completion rate is required then MPR capability improves performance by 12.5% and 10%, respectively. Results for R=300m and R=350m also show modest improvements in this case. From this we may conclude that the type of MPR capability we consider results in small increases in throughput capacity when the spreading factor is sufficiently large. Use of larger spreading factors makes this form of MPR more appealing, mirroring the results of the distributed INS-MPR protocol in Chapter 8.

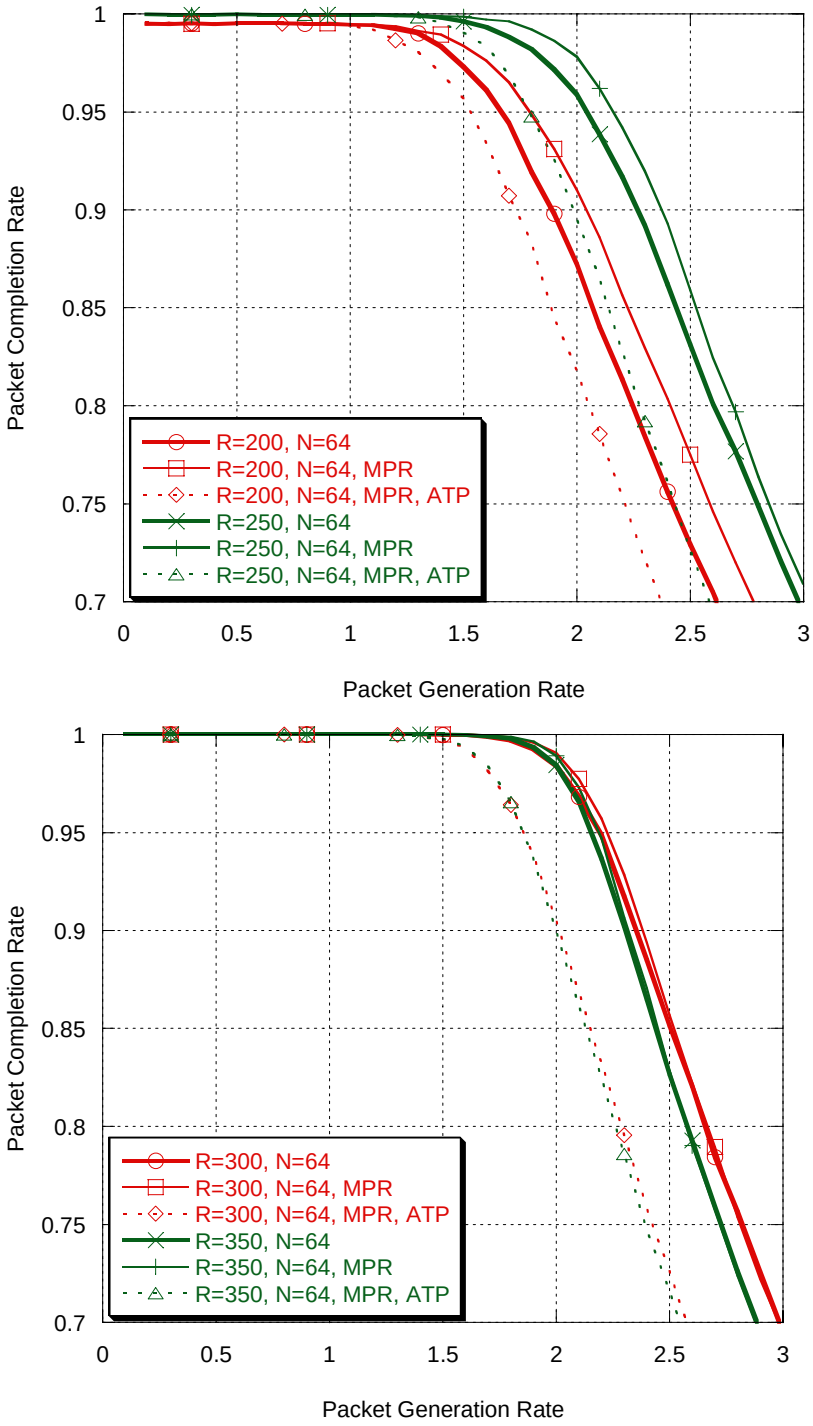


Figure 38. Packet completion rate for networks using a maximum spreading factor set to 64.

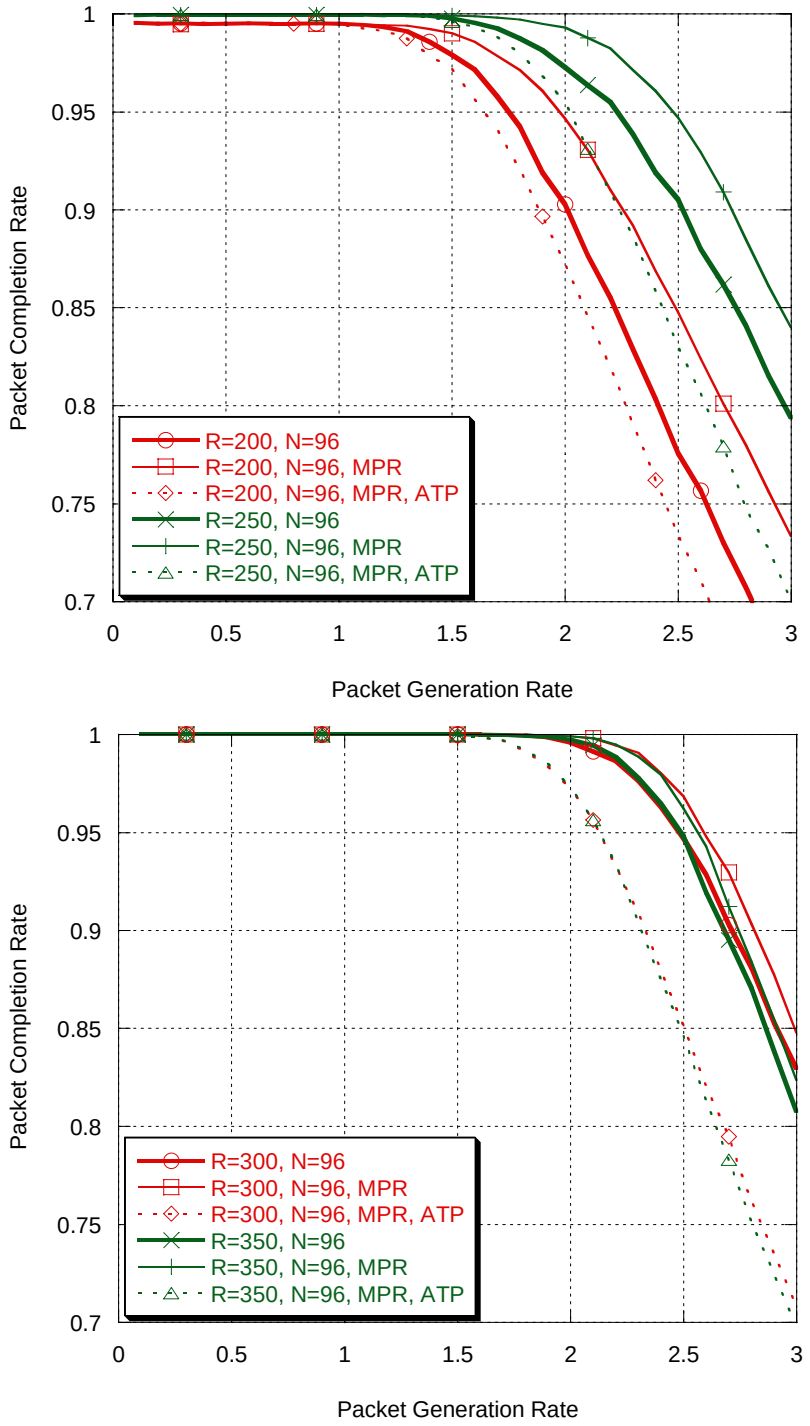


Figure 39. Packet completion rate for networks using a maximum spreading factor set to 96.

CHAPTER TEN

CONCLUSIONS

The primary contribution of this work is a new approach for designing transmission scheduling protocols in systems which use DSSS modulation. We have shown that scheduling approaches designed under the assumptions of a narrowband channel, such as the approach used for the BTS in simulations, fail to capitalize on the higher MAI tolerance which is possible in a DSSS system. The INS protocol, on the other hand, uses a more aggressive approach to schedule transmissions, sacrificing a small degree of connectivity for greatly improved spatial reuse of the channel. This results in significantly better performance in networks with robust connectivity.

The design of the INS protocol is particularly advantageous in mobile ad hoc networks for two reasons. First, terminals operating under the INS protocol do not have to coordinate across multiple hops to make changes to the transmission schedule in response to mobility. Second, the INS protocol allows terminals to operate with a shorter transmission frame than a scheduling protocol which uses a larger neighborhood to schedule transmissions. As a result, terminals can more quickly establish bidirectional communications links with neighbors in a mobile network. These features also make network initialization easier.

We evaluate the performance of the INS protocol using unicast data traffic. The INS protocol is envisioned for systems using distributed routing protocols, such as OLSR, as well as systems which handle multicast and broadcast traffic. The INS protocol is well-suited for these systems since it constructs a broadcast transmission

schedule for each terminal. Not all neighboring terminals are reachable in every slot, but through the transmit matrix, a terminal generally has knowledge of which neighbors are reachable in a particular slot.

The INS protocol accounts for the diversity of communications links, via link SINR estimates, which are available to a terminal. Multiple packets are scheduled for transmission in a slot when link quality allows it by using a reduced spreading factor. Unicast data packets are only forwarded to neighbors for which a bidirectional communications link has been established, and these packets are only transmitted in slots for which the neighbor has indicated reception is possible.

The link cost metric used in routing incorporates link SINR estimates and slot utilization estimates reported through the exchange of NBR_ACK control packets. The link cost metric is designed to emphasize use of reliable links with high SINR estimates, as well as links that may be activated at high transmission rates, and that are associated with neighbors reporting low slot utilization and greater effective transmission rate.

The INS protocol uses Lyui's transmission scheduling algorithm to assign transmission slots to terminals based on the local neighborhood determined through reception of FLAG packets. We demonstrated that Lyui's algorithm provides better spatial reuse than two other well-known slot assignment algorithms. There are two reasons for this. First, Lyui's algorithm uses a variable frame length which depends on local terminal density; thus, terminals in sparse areas of the network to transmit more frequently and priority starvation is reduced. Second, Lyui's algorithm uses colors instead of terminal ID to arbitrate transmission priority. This effectively reduces the

number of contending entities viewed by a terminal, since some neighbors can be assigned the same color. As a result, instances of priority chaining occur less often.

We designed two protocol variants designed to enhance the connectivity of sparse networks operating with the INS protocol. One approach, selective collision elimination, introduces additional overhead in the form of BLOCK packets. These packets allow a terminal to separate two neighbors transmitting in the same slot when they interfere excessively with one another, or when both neighbors are important for forwarding packets. Another approach leverages multi-packet reception capability (MPR) to improve network performance. MPR allows terminals to establish more communications links without requiring additional control packet overhead. In sparse networks, establishing more communications links improves the network connectivity, leading to better network performance. In dense networks with large traffic demands, the protocol variation which leverages MPR is preferable to the approach of selective collision elimination because it does not introduce additional communication overhead.

As a final step, we analyzed the throughput capacity of networks using a centralized transmission scheduling algorithm. Our results suggest that MPR, in the form outlined in this work, has limited potential for improving throughput capacity of networks. This potential for improvement is greater in networks for which $\beta' << 1$, since more scenarios for MPR are feasible. Other strategies, such as activating transmitters at a higher rate or distributing network traffic more evenly through routing, may be more effective means for increasing throughput capacity. Other forms of MPR, such as MPR

based upon multi-user detection and successive interference cancellation, may also be more effective means for increasing throughput capacity.

REFERENCES

[1] N. Abramson, “The ALOHA system – another alternative for computer communications,” *AFIPS Conference Proceedings*, vol. 37, pp.281-285, November, 1970.

[2] L. Kleinrock and F.A. Tobagi, "Packet switching in radio channels: part I - carrier sense multiple access modes and their throughput delay characteristics", *IEEE Transactions on Communications*, vol. com-23, no. 12, pp. 1400- 1416, 1975.

[3] P. Karn, “MACA - a new channel access method for packet radio,” *ARRI/CRRI Amateur Radio 9th Computer Networking Conference*, pp. 134-140, Sept. 1990.

[4] C. Perkins, E. Belding-Royer, S. Das, “Ad hoc on-demand distance vector (AODV) routing,” *RFC Editor*, RFC3561, 2003

[5] T. Clausen, P. Jacquet, “Optimized link state routing protocol (OLSR),” *RFC Editor*, RFC3626, 2003.

[6] B. Sklar, *Digital Communications*, Prentice-Hall, Englewood Cliffs, NJ, 3rd ed., 1988.

[7] M. B. Pursley, “Performance evaluation for phase-coded spread-spectrum multiple-access communication: Part I. System Analysis,” *IEEE Transactions on Communications*, vol. com-25, no. 8, pp. 795-799, Aug. 1977.

[8] J. Grönkvist, and A. Hansson, “Comparison between graph-based and interference-based STDMA scheduling,” *ACM MobiHoc*, pp. 255-258, Oct. 2001.

[9] A. Behzad, and I. Rubin, "On the performance of graph-based scheduling algorithms for packet radio networks," *IEEE GLOBECOM*, vol.6, pp. 3432- 3436, Dec. 2003.

[10] H. Zhu, Y. Tang, I. Chlamtac, "Unified collision-free coordinated distributed scheduling (CF-CDS) in IEEE 802.16 mesh networks," *IEEE Transactions on Wireless Communications* , vol.7, no.10, pp.3889-3903, Oct. 2008.

[11] Y. Shi, Y. T. Hou, J. Liu, and S. Kompella, “How to correctly use the protocol interference model for multi-hop wireless networks,” *ACM MobiHoc*, pp. 239-247, May 2009.

[12] L.C. Pond and V.O.K. Li "A distributed time-slot assignment protocol for mobile multi-hop broadcast packet radio networks," *Proc. IEEE Miltary Communications Conf. (MILCOM).*, vol. 1, pp.70-74, Oct. 1989.

[13] S. Ramanathan, E. L. Lloyd, “Scheduling algorithms for multihop radio networks,” *IEEE/ACM Transactions on Networking*, vol. 1, no. 2, pp. 166-177, April 1993.

[14] P. Djukic and S. Valaee, "Link scheduling for minimum delay in spatial re-use TDMA," *Proc. IEEE INFOCOM*, pp.28-36, May 2007.

[15] D. D. Vergados, D. J. Vergados, C. Douligeris, S. L. Tombros, “QoS-aware TDMA for end-to-end traffic scheduling in ad-hoc networks”, *IEEE Wireless Communications*, Vol. 13, No. 5, pp. 68-74, Oct. 2006.

[16] J. Grönkvist, “Distributed scheduling for mobile ad hoc networks – a novel approach,” *2004 IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC 2004)*, vol. 2, pp. 964-968, Sept. 2004.

[17] B.J. Wolf, J. L. Hammond, H. B. Russell, “A distributed load-based transmission scheduling protocol for wireless ad hoc networks,” *2006 International Conference on Wireless Communications and Mobile Computing (ACM IWCMC)*. pp. 437-442. July 2006.

[18] W. Wang, X. Li, O. Frieder, Y. Wang, and W. Song, “Efficient interference-aware TDMA link scheduling for static wireless networks,” *ACM MobiCom*, pp. 262-273, Sept. 2006.

[19] K. Papadaki, and V. Friderikos, “Robust scheduling in spatial reuse TDMA wireless networks,” *IEEE Transactions on Wireless Communications*, vol. 7, no. 12, pp. 4767-4771, December 2008.

[20] A. Ephremides and T. V. Truong, “Scheduling broadcasts in multihop radio networks,” *IEEE Transactions on Communications,* vol. 38, pp. 456-460, Apr. 1990.

[21] G. Wang and N. Ansari, "Optimal broadcast scheduling in packet radio networks using mean field annealing,” *IEEE Journal on Selected Areas in Communications,* vol. 15, no. 2, pp. 250-260, Feb. 1997.

[22] N. Funabiki and Y. Takefuji, “A parallel algorithm for broadcast scheduling problems in packet radio networks,” *IEEE Transactions on Communications*, vol. 41, pp. 828–831, June 1993.

[23] G. Chakraborty, “Genetic algorithm to solve optimum TDMA transmission schedule in broadcast packet radio networks,” *IEEE Transactions on Communications*, vol. 52, no. 5, May 2004.

[24] S. Salcedo-Sanz, C. Bousoño-Calzón, and A.R. Figueiras-Vidal, “A mixed neural-genetic algorithm for the broadcast scheduling problem,” *IEEE Transactions on Wireless Communications*, vol. 2, no. 2, pp. 277-283, March 2003.

[25] B. J. Wolf, J. L. Hammond, and H. B. Russell, “Designing transmission schedules for wireless ad hoc networks to maximize network throughput,” *Proc. IEEE Military Communications Conf.(MILCOM)*, vol. 4, pp. 2012-2018, October 2005.

[26] D. Jungnickel, *Graphs, Networks and Algorithms*, Springer-Verlag, Berlin, Germany, 1999.

[27] P. Bjorklund, P. Varbrand, and D. Yuan, “A column generation method for spatial TDMA scheduling in ad hoc networks,” *Ad Hoc Networks*, vol. 2, no. 4, pp. 405-418, 2004.

[28] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows*, Prentice-Hall, Upper Saddle River, New Jersey, 1993.

[29] S. Kompella, J. E. Wieselthier, and A. Ephremides, "A cross-layer approach to optimal wireless link scheduling with SINR constraints," *Proc. IEEE Military Communications Conference (MILCOM)*, pp.1-7, Oct. 2007.

[30] A. Pantelidou, and A. Ephremides, "Minimum schedule lengths with rate control in wireless networks," *Proc. IEEE Military Communications Conference (MILCOM)*, Nov. 2008.

[31] T. ElBatt, and A. Ephremides, "Joint scheduling and power control for wireless ad hoc networks," *IEEE Transactions on Wireless Communications*, vol.3, no.1, pp. 74-85, Jan. 2004.

[32] S. Ramanathan, “A unified framework and algorithm for channel assignment in wireless networks,” *Wireless Networks*, vol. 5, no. 2, pp. 81-94, March 1999.

[33] W.-P. Lyui, “Design of a new operational structure for mobile radio networks,” Ph.D. dissertation, Clemson University, August 1991.

[34] J. L. Hammond and H. B. Russell, “Properties of a transmission assignment algorithm for multiple-hop packet radio networks,” *IEEE Transactions on Wireless Communications*, vol. 3, no. 4, pp. 1048-1052, July 2004.

[35] X. Ma and E. L. Lloyd, “Evaluation of a distributed broadcast scheduling protocol for multihop radio networks,” *Proc. IEEE Military Communications Conf.(MILCOM)*, vol. 2, pp. 998-1002, 2001.

[36] C. D. Young, “USAP multiple access: dynamic resource allocation for mobile multihop multichannel wireless networking,” *Proc. IEEE Military Communications Conf.(MILCOM)*, pp. 271–275, October 1999.

[37] P. K. Appani, J. L. Hammond, D. L. Noneaker, and H. B. Russell, "An adaptive transmission-scheduling protocol for mobile ad hoc networks, " *Ad Hoc Networks*, vol. 5, no. 2, pp. 254-271, March 2007.

[38] B. J. Wolf, J. L. Hammond, D. L. Noneaker, and H. B. Russell, "A protocol for construction of broadcast transmission schedules in mobile ad hoc networks," *IEEE Transactions on Wireless Communications*, vol. 6, no. 1, pp. 74-79, Jan. 2007.

[39] C. Zhu, and M. S. Corson, “A five-phase reservation protocol (FPRP) for mobile ad hoc networks,” *Wireless Networks*, vol. 7, no. 4, pp. 371-384, July 2001.

[40] A. D. Myers, G. V. Záruba, and V. R. Syrotiuk, “An adaptive generalized transmission protocol for ad hoc networks,” *Mobile Networks and Applications*, vol. 7, no. 6, pp. 493-502, December 2002.

[41] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: distributed randomized TDMA scheduling for wireless ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 10, pp. 1384-1396, Oct. 2009.

[42] R. Rozovsky and P.R. Kumar, “SEEDEX: A MAC protocol for ad hoc networks,” *ACM MobiHoc*, pp. 67-75, Oct. 2001.

[43] L. Bao and J.J. Garcia-Luna-Aceves, “Distributed dynamic channel access scheduling for ad hoc networks,” *Journal of Parallel and Distributed Computing* vol. 63, no. 1, pp. 3-14, Jan. 2003.

[44] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu,” Impact of interference on multi-hop wireless network performance,” *ACM MobiCom*, pp. 66-80. Sept. 2003.

[45] Y. Yi, G. de Veciana, and S. Shakkottai, “On optimal MAC scheduling with physical interference,” *Proc. IEEE INFOCOM*, pp.294-302, May 2007.

[46] X. Yang, G. de Veciana, "Inducing multiscale clustering using multistage MAC contention in CDMA ad hoc networks," *IEEE/ACM Transactions on Networking*, vol.15, no.6, pp.1387-1400, Dec. 2007.

[47] L. Bao, "MALS: multiple access scheduling based on latin squares," *Proc. IEEE Military Communications Conf.(MILCOM)*, vol.1, pp. 315-321, 2004.

[48] RSA Data Security, Inc. MD5 Message Digest Algorithm, © 1991-1992, translated by M. Abzug for C++, v.1.02. Available at http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html.

[49] A. Ebner, H. Rohling, M. Lott, and W. Halfmann, “Decentralized slot synchronization in highly dynamic ad hoc networks,” *Fifth International Symposium on Wireless Personal Multimedia Communications*, vol.2, pp. 494-498, Oct. 2002.

[50] B. J. Wolf, H. B. Russell, and K.-C. Wang, "Synchronizing transmission schedules of partitioned ad hoc networks," *Proc. IEEE Military Communications Conf. (MILCOM)*, pp. 1-7, Oct. 2007.

[51] M. B. Pursley, “The role of spread spectrum in packet radio networks,” *Proc. IEEE*, vol. 75, no. 1, pp. 116-134, Jan. 1987.

[52] C. Ware, J. Chicharo, and T. Wysocki, "Simulation of capture behaviour in IEEE 802.11 radio modems," *Proc. of IEEE Vehicular Technology Conference*, vol.3, pp.1393-1397, Oct. 2001.

[53] P. Gupta and P. R. Kumar, “The capacity of wireless networks,” *IEEE Transactions on Information Theory*, vol. 46, pp. 388–404, Mar. 2000.

[54] C. Bettstetter, G. Resta, and P. Santi., “The node distribution of the random waypoint mobility model for wireless ad hoc networks,” *IEEE Transactions on Mobile Computing*, vol. 2, no. 3, pp. 257-269, July-Sept. 2003.

[55] B. Hajek, A. Krishna, and R. O. LaMaire, “On the capture probability for a large number of stations,” *IEEE Transactions on Communications*, vol. 45, no. 2, pp. 254–260, Feb. 1997.

[56] G. D. Celik, G. Zussman, W. F. Khan, and E. Modiano, "MAC for networks with multipacket reception capability and spatially distributed nodes," *Proc. IEEE INFOCOM,* pp.1436-1444, April 2008.

[57] J.J. Garcia-Luna-Aceves, H. R. Sadjapour, and Z. Wang, “Challenges: towards truly scalable ad hoc networks,” *ACM MobiCom*, pp. 207-214, September 2007.

[58] P. Stuedi and G. Alonso, “Computing throughput capacity for realistic wireless multihop networks,” *Proc. MSWiM ’06*. ACMPress, 2006.

[59] P. Stuedi, and G. Alonso, "Log-normal shadowing meets SINR: a numerical study of capacity in wireless networks," *Proc. IEEE SECON*, pp.550-559, June 2007.

[60] R.L. Cruz and A.V. Santhanam, “Optimal Routing, Link Scheduling and Power Control in Multi-Hop Wireless Networks,” *Proc. IEEE INFOCOM,* vol. 1, pp. 702-711, Mar.-Apr. 2003.

[61] M. Franceschetti, O. Dousse, D. N. C. Tse, and P. Thiran, “Closing the gap in the capacity of wireless networks via percolation theory,” *IEEE Transactions on Information Theory*, vol. 53, no. 3, pp. 1009-1018, March 2007.

[62] S. Ghez, S. Verdu, and S. C. Schwartz, "Stability properties of slotted aloha with multipacket reception capability," *IEEE Transactions on Automatic Control*, vol.33, no.7, pp.640-649, July 1988.

[63] G. Mergen and L. Tong, “Receiver controlled medium access in multihop ad hoc networks with multipacket reception,” *Proc. IEEE Military Communications Conference (MILCOM)*, vol. 2, pp. 1014–1018, Oct. 2001.

[64] J. Zhang, Z. Dziong, F. Gagnon, and M. Kadoch, "Multiuser detection based MAC design for ad hoc networks," *International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness: Qshine 2007*, Aug. 2007.

[65] P. Patel and J. Holtzman, "Analysis of a simple successive interference cancellation scheme in a DS/CDMA system", *IEEE Journal on Selected Areas in Communications*, vol. 12, pp. 796 - 807, 1994.

[66] S.P. Weber, J.G. Andrews, X. Yang, and G. de Veciana, "Transmission capacity of wireless ad hoc networks with successive interference cancellation," *IEEE Transactions on Information Theory*, vol. 53, no. 8, pp. 2799-2814, Aug. 2007.

[67] R. Yim, N. B. Mehta, A. F. Molisch, and Jinyun Zhang, "Efficient multiple access using received signal strength and local channel information," *Proc. IEEE Wireless Communications and Networking Conference*, pp.1962-1967, April 2008.

[68] A. Muqattash and M. Krunz, “CDMA-based MAC protocol for wireless ad hoc networks,” *Proc. ACM MOBIHOC*, pp. 153-164, 2003.