A Performance Modeling of Ad hoc Networks with Multi-cell Overlapping Interference

Xiaoyu Wang

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of Requirements
for the Degree of Master of Applied Science at
Concordia University
Montreal, Quebec, Canada

March 2005

Your file Votre référence
ISBN: 0-494-14284-7
Our file Notre référence
ISBN: 0-494-14284-7

Abstract

A Performance Modeling of Ad hoc Networks with Multi-cell Overlapping Interference

Mobile ad hoc networks are non-infrastructure-based wireless networks which are expected to be an important part of 4G architecture. In this type of network, nodes are both source of information as well as they participate in relaying the information to their final destinations. Due to the broadcast nature of wireless transmission, mobile ad hoc networks require an effective MAC protocol to share the medium. In recent years, many MAC protocols have been proposed in the literature for this purpose, such as Dual Busy Tone Multiple Access (DBTMA) [3] protocol. In this thesis, we first propose to modify DBTMA protocol through inserting a third busy tone, BTd (data transmit busy tone) to protect data transmissions and the modified protocol is referred to as Triple-BTMA (T-BTMA). Second, we show that the existing performance model of DBTMA fails to consider the effects of multi-cell overlapping interference. The main contribution of this thesis is development of a performance analysis of T-BTMA protocol that takes into account properly the multi-cell overlapping interference. We show that our model may also be adopted for most Ready-to-Send/Clear-to-Send (RTS/CTS) based MAC protocols. We develop a simulation environment to validate approximations in our mathematical model. We find that theoretical results are in agreement with those of simulations. Finally, we extend our model into power-controlled networks and it is shown that a proper power control can lead to an ideal channel throughput.

Acknowledgements

I would like to express my deepest gratitude to my thesis supervisor, Dr. M. Mehmet Ali, for immensely helpful comments and suggestion. His guidance, background knowledge, patience and editing skills contributed enormously to the completion of this thesis.

I would like to thank my parents Wenyi Zhang and Linquan Wang for giving me the encouragement and strength I needed to complete my goals.

Special recognition goes out to my boyfriend Yihan Gu for his support, generosity and his excellent cooking.

Finally, I would like to acknowledge my lovely cat, who spent long time sleeping on my keyboard.

Contents

List of Figures

List of Tables

List of Symbols

ACK	Acknowledges
BTd	Data Transmit busy tone
BTMA	Busy-tone Multiple Access
BTr	Receive Busy Tone
BTt	Transmit Busy Tone
CA	Collision Avoidance
CS	Carrier Sensing
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access-Collision Avoidance
CTS	Clear-to-Send
CW	Contention Window
DBTMA	Dual Busy Tone Multiple Access
DCF	Distributed Coordination Function
DS	Data Sending
DSSS	Direct Sequence Spread Spectrum
FAMA	Floor Acquisition Multiple Access
FAMA-NCS	Floor Acquisition Multiple Access (Non-persistent Carrier Sensing)
FAMA-NPS	Floor Acquisition Multiple Access (Non-persistent Packet Sensing)

FHSS	Frequency Hopping Spread Spectrum
GPS	Global Positioning System
IEEE	Institute of Electrical and Electronics Engineers
IR	Infrared
LAN	Local Area Network
MAC	Media Access Control
MACA	Multiple Access Collision Avoidance
MACAW	MACA for Wireless
MANET	Mobile ad hoc networks
OFDM	Orthogonal Frequency Division Multiplexing
PCF	Point Coordination Function
PCMA	Power Controlled Multiple Access
PDA	Personal Digital Assistants
PDF	Probability Density Function
PRNET	Packet Radio Networks
RTS	Ready-to-Send
Rx	reception
SURAN	Survivable Adaptive Radio Networks
T-BTMA	Triple-BTMA
TDMA	Time Division Multiple Access
Tx	transmission
UWB	Ultra Wideband

WLAN	Wireless Local Area Network
t_d	busy tone detection delay
τ	propagation and mechanical delays
γ	transmission time of an RTS packet
δ	transmission time of a data packet
t_{mw}	mandatory waiting time
i	node i, for example, A denotes node A
$\bar{i}$	denotes all the nodes that are resident in home cell of i, for example, $\overline{B}$
η	node density per unit area
Λ_η	mean packet generation rate per unit area (this rate includes both new packets as well as retransmissions)
S_i	coverage area of home cell of node i
Ω_i	mean packet generation rate in home cell of i, $\Omega_i = \Lambda_\eta \cdot S_i$
R	transmission radius of a node
ℓ	distance between source and destination nodes during a transmission
T_s	average duration of a busy period that carries a successful transmission
T_f	average duration of a busy period that carries an unsuccessful transmission
$\overline{B}_i$	mean busy period of home cell i during a cycle
$\bar{I}_i$	mean idle period of home cell i during a cycle

$\overline{T}_i$	mean busy period of home cell *i* that carries a successful transmission
P_i^S	probability of a successful transmission in home cell *i*
$P_i^S(\ell)$	probability of a successful transmission in home cell *i* in power controlled networks
$P_h(\ell)$	probability of harmless ongoing transmission
S^{Tx}	effective transmission area
ρ_i	throughput of a channel serving cell *i*
$\overline{\rho}_i$	average throughput of a channel serving cell *i*
S	channel throughput in [3]
λ	mean aggregate generation rate in [3]
Q_i	throughput of each simulation run
r	number of simulation runs
Q	network throughput of simulation results
$\overline{Q}$	channel throughput of simulation results

Chapter 1

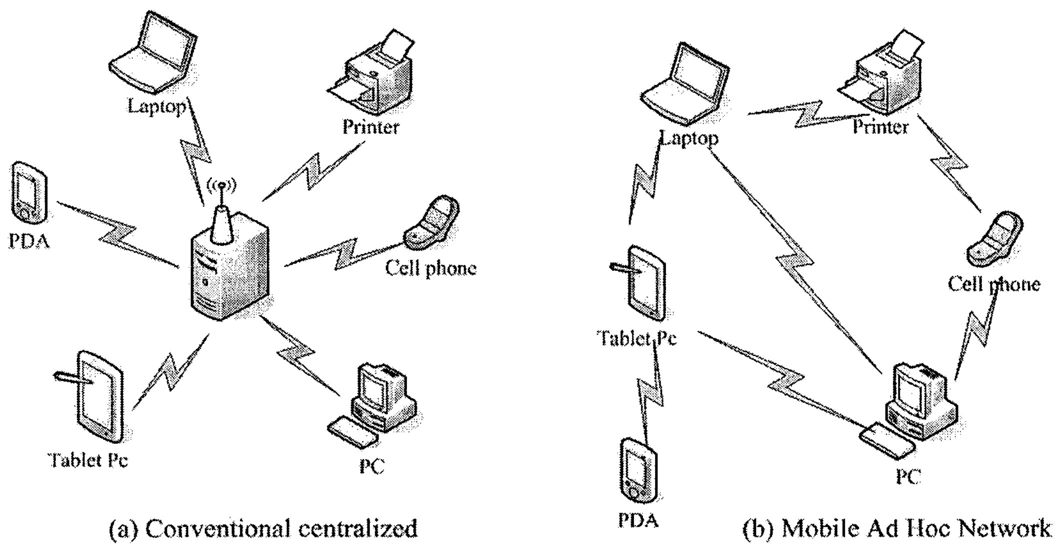
Introduction

It has been almost a decade that mobile ad hoc networks have drawn much attention in the area of wireless communications. Mobile ad hoc networks are non-infrastructure-based networks which are expected to be an important part of the 4G architecture. The transient and dynamic network architecture of mobile ad hoc networks makes them different from traditional wireless networks.

1.1 Traditional Wireless Networks vs. Mobile Ad Hoc Networks

In general, the traditional wireless networks primarily depend on fixed pre-constructed infrastructure and centralized administration to share information and resources. As shown in Figure 1.1-a, all nodes have direct access to base stations or access nodes. This figure illustrates a conventional centralized wireless local area network (WLAN) with base stations providing network infrastructure. The limitation of such wireless networks is that mobile users can receive network services only where infrastructures are available. However, in many situations, user-required infrastructures cannot be installed due to geography of areas, such as a desolate landscape, battlefields or a heritage site. And there are other situations that traditional wireless networks cannot be set up in a short period of time, for instance, in disaster areas or emergencies. Let's imagine a tragic scenario; a city is out of power because of an earthquake. There isn't any working base station or server; all communications are cut; no cell-phone signal is available at all. If cell-phones spontaneously set up networks without any servers, such an ad hoc network will greatly contribute to rescue operation. Not only can a rescue team contact people and save people's life, but also human beings can help one and another. Therefore, mobile ad hoc networks are highly desired for delivering wireless services in these situations.

Mobile ad hoc networks (MANET) take their name from the fact that they don't have pre-placed infrastructures, such as servers, routers and base stations, and they are formed in ad hoc manner. All mobile devices including cell phones, laptops and handheld digital devices can spontaneously set up networks anywhere at any time (see Figure 1.1-b). In an ad hoc network, a message transfer can occur either between two neighboring nodes or through some intermediate nodes which is relayed hop by hop. Therefore, ad hoc networks are often called wireless multiple-hop networks.



(a) Conventional centralized WLAN

(b) Mobile Ad Hoc Network MANET

Figure 1. 1 Centralized WLAN and MANET

1.2 The Short History of Ad Hoc Networks

The concept of ad hoc networks can be traced back to 1972, when the Packet Radio Networks (PRNET) project was launched for military purposes [16, ch.1]. The objective of the project was to exploit available radio technologies to provide reliable transmission of data packets in distributed wireless environments. PRNET is considered as the forerunner of today's mobile ad hoc networks. Following the termination of the PRNET project, the Survivable Adaptive Radio Networks (SURAN) program was developed in the 1980s. SURAN was intended to improve the radio adaptability, security and capacity of PRNET. As mobile computing exponentially grew in the 1990s, especially the emergence of masses of wireless devices such as cell phones, laptops and personal digital assistants (PDAs), the great commercial potential and advantages of these packet radio technologies became evident. At the same time, packet radio networks were given a new name as ad hoc networks, and the idea of commercial ad hoc networks merged. Later on, most of the ad hoc networking research outside the military domain was developed in the academic and commercial environments.

High demand for commercial deployment spurred the standardization of the technologies for wireless connectivity. By 1997, the first version of the Institute of Electrical and Electronics Engineers (IEEE) standard 802.11 had been released. Almost immediately in 1999, the second version of IEEE Std 802.11 was approved. "The purpose of this standard is to provide wireless connectivity to automatic machinery, equipment, or stations that require rapid deployment, which may be portable or hand-held, or which may be mounted on moving vehicles within a local area" [13]. Although technologies based on IEEE 802.11 are standardized for WLAN, the ad hoc mode is allowed in the infrastructure-less part of the specification. In fact, most of the ad hoc networking research is based on WLAN technologies.

1.3 Three Main Problems Specific to Ad Hoc Networks

As described in a previous section, mobile ad hoc networking is an attractive innovation in wireless networking technology; however, it inherits and complicates conventional problems of wireless and mobile communications. Specifically, contention for the medium in wireless transmission results in low network capacity. In ad hoc networks, all nodes rely upon the carrier sensing random access protocol to share the medium, and the presumption of these protocols is that each node is able to hear the transmission from every other node in the network. Thus, if this presumption is not realized in such a network, then using the carrier sensing protocol is inefficient due to the known hidden and exposed-terminal problems. In addition, since the broadcast nature of transmission causes high probability of interference, energy management can reduce such interference, as well as maximize channel reuse of the medium to increase network capacity. However, reduced transmission power may reduce the network connectivity, increase the number of hops, and introduce "Hidden Terminal Jamming" problem [25]. In the next subsection, hidden-terminal problem, exposed-terminal problem and power control problem are discussed in detail.

1.3.1 Hidden-terminal Problem

Figure 1.2-a illustrates an example of hidden-terminal problem. There are three nodes A, B and H in the given wireless network. We assume that node transmission ranges are identical. In this figure, circles centered at nodes A, B, H correspond to the transmission ranges of these nodes. As seen, B is within the transmission ranges of nodes A and H, which are outside of the transmission ranges of each other. Suppose that A is transmitting to B; thus, H is considered to be a hidden terminal with respect to this transmission. Since H is not aware of A's transmission, it may initiate its own transmission to one of the nodes within its range. Clearly, the transmission of H will result in interference at B destroying A's packet. Therefore, a station listening to the channel before transmitting isn't guaranteed interference free reception. In other words, the hidden-terminal problem degrades network throughput of MAC layer.

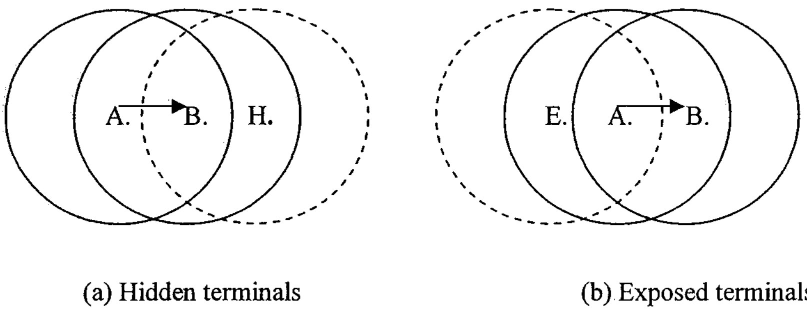


Figure 1. 2 Hidden and exposed terminals in a wireless network

1.3.2 Exposed-terminal Problem

Next we consider the exposed-terminal problem. As shown in Figure 1.2-b, nodes A and E are within transmission ranges of each other, while B is only within range of A. Suppose that A is transmitting to B, then E is considered to be an exposed terminal with respect to this transmission. Since E doesn't know whether or not the destination of A's transmission is outside of its range, it will defer its own transmission. Thus exposed-terminal problem also reduces channel throughput.

1.3.3 Power Control Problem

Finally, we talk about the power control problem. Mobile devices depend on battery power, and since device size and weight usually limit power capacity, efficient power management becomes a vital issue in designing protocols for mobile communications. In conventional infrastructure based wireless networks, although mobile devices also have constraints of battery power, various power management strategies are used such that resource-rich base stations consume more power while mobile devices less. In ad hoc networks, where there are no such fixed centralized infrastructures, every mobile device either transmits directly to a destination or forwards traffic as a router, which requires additional energy.

Moreover, although reduced transmission power can maximize the reuse of medium by limiting multi-user interference, it also increases the number of hops that it will take to deliver a packet to its final destination, as well as increases transmission delay. The extreme case is losing network connectivity; therefore, many researchers have aimed at optimal power control to achieve efficient channel reuse. A simple idea is to use ready-to-send/clear to send (RTS/CTS) exchange at the maximum power level during channel acquisition while minimizing the power during the transmission of a data packet. However, this doesn't increase channel reuse because it may not decrease the number of suppressed transmissions based on the fixed-power RTS and CTS [21]. For example, in Figure 1.3, C is located outside A's power-controlled transmission range (solid line circle) but inside the maximum transmission range (dotted line circle). Allowing nodes like C to use the channel is the intention of the power controlled MAC protocols with fixed power RTS/CTS. However, since C has already overheard an RTS-CTS exchange between A and B before A's data transmission, it is forced to wait till the end of this data transmission regardless of whether A's data transmission is sent at power level or not. Therefore, the performance of this power controlled MAC protocol with fixed power RTS/CTS is same as those without power controlled MAC protocols. In other words, the RTS-CTS exchange must be sent at flexible power levels to allow as many simultaneous transmissions as possible. The dilemma is that control packets with reduced power may introduce "Hidden Terminal Jamming" problem, which will also degrade throughput. As can be observed from Figure 1.4, the ongoing low power level transmission from A to B is corrupted by C's concurrent transmission to D.

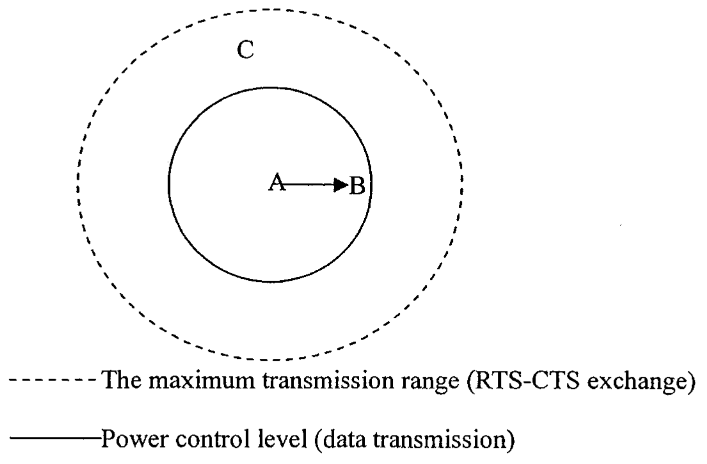

C
A
B
------The maximum transmission range (RTS-CTS exchange)
Power control level (data transmission)

Figure 1. 3 Suppression of transmission from C in power-fixed RTS/CTS power control protocol

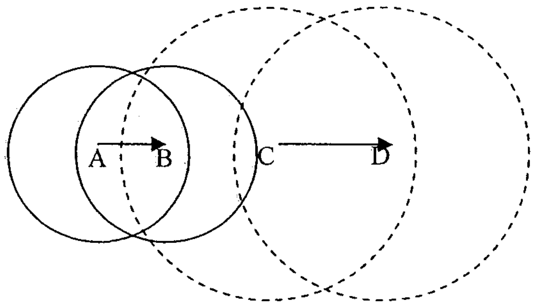


Figure 1. 4 Hidden terminal jamming problem

1.4 Literature Review

So far there have been numerous papers addressing the hidden-terminal and the power control problems. Most of these studies are based on simulation and test bed results, while few papers present analytical models. The first group of these papers is trying to solve the hidden-terminal problem, and the second is using power control methods to increase channel reuse and save energy. Next, we provide a brief summary of both groups of papers.

1.4.1 Review of Solutions for Hidden-terminal Problem

Early in 1975, Tobagi and Kleinrock in [6] considered the effect of hidden-terminal problem in MAC protocols and proved that the CSMA protocol suffers most from the existence of hidden terminals in packet radio networks with base stations. In their centralized network model, hidden terminals occur when two nodes, for example A and B in Figure 1.5, are within the range of the base station but out of range of each other or obstructed by walls. The authors group nodes which can hear the same subset of nodes in the population, and then first study the independent group case for both 1-persistent and non-persistent CSMA. After that, they considered an approximation model for dependent groups for non-persistent CSMA. Their analytical and simulation results show that the bigger is the proportion of hidden terminals, the smaller is the channel throughput. Based on this observation, Tobagi and Kleinrock provide a fairly good solution to the hidden-terminal problem through a so called Busy-tone Multiple Access (BTMA) mode in a single central-station environment. According to BTMA, a shared channel is divided into two sub-channels: a busy-tone channel and a message channel. Whenever the base station senses carrier on the message channel, it sets up a sine-wave busy tone signal on the busy-tone channel. Since the base station is in hearing range of all nodes, all nodes in the network obtain the status of the medium by sensing the busy-tone channel. The simulations show that BTMA results in better performance than CSMA on traditional single-central-station networks.

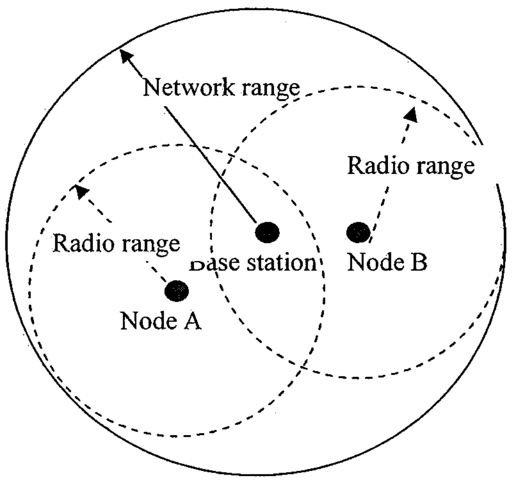


Figure 1. 5 Hidden terminals occur in a centralized network

As the most mature technology for infrastructure-based wireless LANs, the IEEE 802.11 standard defines both the physical layer and the Multiple Access Control (MAC) layer for wireless communications, including fixed and moving nodes. In the physical layer, the standard supports five types of transmission: infrared (IR), frequency hopping spread spectrum (FHSS), direct sequence spread spectrum (DSSS), orthogonal frequency division multiplexing (OFDM), and ultra wideband (UWB). Since nodes communicate wirelessly and share the same media, media contention and therefore packet collisions are inevitable. The MAC layer, which deals with congestion control and collision avoidance, becomes a crucial part of wireless communications. In this layer, IEEE 802.11 provides two different types of mechanisms to access the medium: Distributed Coordination Function (DCF) and Point Coordination Function (PCF). DCF is known as Carrier Sense Multiple Access—Collision Avoidance (CSMA/CA) protocol and specified to be used within both ad hoc networks and infrastructure network configuration, while PCF is considered unsuitable for ad hoc networks because of lack of centralized control [13].

The CSMA/CA protocol is designed to reduce collisions when nodes access a shared medium. Every node senses the medium before transmitting. An exchange of RTS and CTS packets is used to reserve the medium before a data transmission actually takes place. And a random back-off procedure is used to resolve medium contention conflicts which occur at the end of the idle period. When a node has a data packet to send, it sends an RTS packet after sensing an idle medium. As the destination receives this packet, it consents to the impending communication by sending a CTS packet in reply. Upon receiving the CTS packet, the source starts its data transmission. At the end of the communication, the receiver acknowledges (ACK) the correct reception.

In [14], researchers study the effect of hidden terminals under different transmission rates and collision-free communications. They demonstrate that the hidden-terminal problem at the lower transmission rate more seriously damages the performance than at the higher transmission rate. In [15], a multiple channel CSMA protocol is proposed to reduce the impact of the hidden-terminal problem in ad hoc networks. The inspiration to this protocol comes from a multiple channel wired LAN, where each node selects one of idle sub-channels for transmission through carrier sensing. When the number of sub-channels is large, it means every node can reserve a channel for data transmission. This channel reservation strategy lowers the probability of contention for the medium as well as alleviates the hidden-terminal problem. However, a large number of channels may cause longer transmission delay [15].

Recently, a new generation of MAC protocols have been proposed to solve the hidden-terminal problem in ad hoc networks, such as MACA (multiple access collision avoidance), MACAW (MACA for Wireless), FAMA (floor acquisition multiple access), DBTMA (dual busy-tone multiple access) and so on...

In MACA [1], Karn points out that as long as hidden terminals or exposed terminals exist, lack or presence of carrier does not always indicate availability of a channel. Therefore, he eliminates the carrier sensing (CS) in CSMA/CA and extends the collision avoidance (CA) part to relieve both the hidden and exposed terminal problems. This is done through an RTS-CTS-DATA exchange. When a node has a data packet to send, it sends an RTS packet first without sensing the carrier. As soon as the destination successfully receives the RTS packet, it sends back a CTS packet to the source. Then, the source starts to transmit the data packet following the reception of the CTS packet. The RTS and CTS frames contain a duration value field that defines the period of time that the medium is to be reserved. Any neighbor overhearing them defers its attempt for long enough time to protect ongoing communications. Since there is no carrier sensing, the MACA attempts to sacrifice the less costly collisions between RTS packets to provide collision free data transmission.

Bharghavan [2] proposed an enhanced MACA protocol, called MACAW. The major difference between MACA and MACAW is the use of a data sending (DS) packet from a source and an acknowledgement (ACK) from a destination. An RTS-CTS-DS-DATA-ACK exchange is used for reliable data transmissions. The DS packet is sent prior to the DATA transmission, and used to notify all nodes in the range of the source regarding a successful RTS-CTS exchange and impending data transmission. Moreover, an ACK packet is transmitted immediately to complete data transmission. If the transmitter doesn't receive the ACK, it schedules a fast retransmission. The results from a test bed show significant improvement in performance. However, neither MACA nor MACAW completely solves the exposed-terminal problem. As may be seen in Figure 1.6-a, although the exposed terminal E is allowed to initiate an RTS, it still cannot receive any packet due to interference from the ongoing transmission from A to B. In addition, a node which does not overhear any RTS and CTS packet, for example of node C, in Figure 1.6-b, becomes a hidden terminal when it moves within the communication coverage range. This kind of hidden terminals introduced by nodal mobility may seriously destroy data transmissions. As may be seen in Figure 1.6-b, C's RTS packet destroys the ongoing data transmission from A to B, and this collision is inevitable. Clearly, the shortcomings of these protocols are due to absence of any protection during data transmissions.

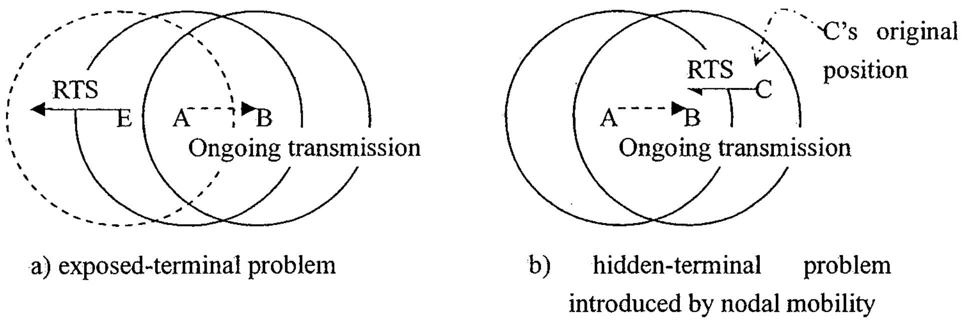

RTS
E
A
B
Ongoing transmission
a) exposed-terminal problem
C's original position
RTS
C
A
B
Ongoing transmission
b) hidden-terminal problem introduced by nodal mobility

Figure 1. 6 Unsolved problems in MACA or MACAW

In [8], FAMA protocols are proposed. According to the FAMA, a node uses the RTS/CTS exchange to acquire control of a channel (a floor) before sending a data packet or a data packet train to different destinations, and at any time, only one node can control the floor. The authors, Fullmer and Garcia-Luna-Aceves, further verify that FAMA protocols must use carrier sensing to support correct floor acquisition by comparing the performance of FAMA-NCS (non-persistent carrier sensing) with FAMA-NPS (non-persistent packet sensing). Moreover, combining non-persistent carrier sensing and the RTS-CTS exchange, FAMA-NCS uses longer CTS packets as a busy tone to ensure collision-free data transmission. Once a node has begun to send the longer CTS packet, its neighbors which are sending RTS packets will receive a portion of the longer CTS packet. Such neighboring nodes will backoff from accessing the channel, thereby allowing the consequent data packet to arrive at the destination free from collisions. FAMA protocols successfully solve the hidden-terminal problem [5,8] in single-channel packet-radio networks. However, the longer CTS packet may have negative effect in ad hoc networks with hidden terminals [5]. A node may mistake collided RTS packets for a longer CTS packet, and as a result, it may suppress its own data transmission. Thus, the channel capacity is wasted [5], especially when RTS collisions occur frequently under heavy traffic.

In [3,4,5], Dual Busy Tone Multiple Access (DBTMA) scheme is proposed to address the hidden-terminal problem and packet collisions. According to this protocol, channel is divided into a message channel and a control channel; special signals called BTt (transmit busy tone) and BTr (receive busy tone) are added to the channel at different frequencies. Each node in DBTMA detects the busy tones to determine the status of the channel before sending an RTS packet. If any busy tone is sensed, a node defers its transmission until a proper time. However, we find data transmissions are still at risk for collisions and interference. We leave details of this protocol to the next chapter because of its significance for our own work.

1.4.2 Review of Power-controlled MAC protocols

We begin this section with a brief overview of directional MAC protocols, and then review works on power-controlled MAC protocols. Both directional antennas and power control are implemented to increase spatial reuse, which provides as many simultaneous communications as possible [16]. Adjusting directional antennas towards an intended receiver not only reduces interference but also reduces energy.

For example, [17],[19] and [18] propose the directional antennas based on both with and without physical information, respectively. Their simulation results show improvement in the performance of ad hoc networks. However, they also point out that improper control of antenna beam may introduce new hidden-terminal problems, as may be seen in Figure 1.7. In this figure, the transmission coverage area from A to B is within the transmission coverage area from C to D. Assuming an ongoing transmission from A to B, then, C, D are hidden terminals with respect to this transmission. Since C has no knowledge of this ongoing transmission, it will initiate its own transmission immediately to D. Clearly, this new transmission will destroy the reception at B.

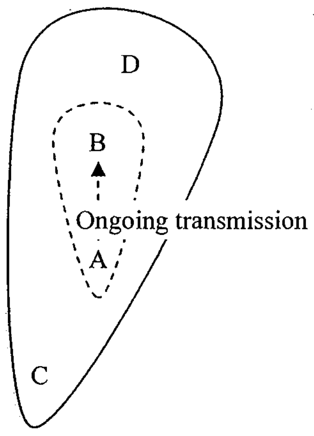


Figure 1. 7 Hidden-terminal problem when using directional antennas

The concept of controlling the transmission range in multi-hop packet radio networks is first introduced in [20]. They demonstrate the disadvantage of using a large transmission radius and develop a model to analyze the optimal transmission radius with a variety of MAC protocols, including slotted ALOHA and CSMA. In

[21], the authors further show that although a shorter transmission radius is favorable in terms of successful transmission due to reduction of interference, decreasing the power too much may leave the network unconnected. Therefore, control of the transmission range to achieve better performance has received great attention in research community.

There are many protocols that adjust the transmission power based on signal-to-interference-and-noise-ratio, for example, Power Controlled Multiple Access (PCMA) protocol [23] and the algorithm in [24]. [24] ties power control and time-division multiple-access (TDMA) scheduling in wireless ad hoc networks, and the simulation results show that distributed power control algorithms deployed for cellular networks can be used in wireless ad hoc networks.

Other protocols controlling the transmission power subject to the distance between source and destination pairs also have been proposed. For example, the scheme presented in [12] is proposed to combine the concept of power control with DBTMA(98). According to this scheme, a source node implements the RTS/CTS exchange to estimate its distance to the destination node; thus, the source transmits its data packet at an appropriate power level to increase the channel reuse. In this new MAC protocol with power control, a source transmits an RTS packet at power-controlled level which is determined based on how strong the BTr signals are around it. Data packets and BTt signals are transmitted at the lowest power level to reach the destination, while CTS packets and BTr signals maintain the largest transmission power in order to combat the hidden-terminal problem. However, the new protocol may become unfeasible to determine the transmission power level when there are several neighboring BTr signals on the air. This work presents a complete model for the analysis of the throughput in ad hoc networks. Their analytical and simulation results show that this new MAC protocol results in higher channel utilization and lower energy consumption. However, their model fails to take into account the effect of multi-cell overlapping interference.

1.5 Major Contributions of the Thesis

This thesis makes contributions to the performance modeling of ad hoc networks. The major contributions of this thesis may be summarized as follows:

1) DBTMA is one of the MAC protocols under serious consideration for ad hoc networks. We explain the main weaknesses of DBTMA protocol and propose the Triple-BTMA (T-BTMA) protocol which inserts a third busy tone, BTd (data transmit busy tone), to protect data transmission.

2) We show that the existing performance models of DBTMA fail to take into consideration the effects of multi-cell overlapping interference. The main objective of this thesis is the development of a performance analysis of T-BTMA protocol that takes into account the multi-cell overlapping interference.

3) We develop a discrete event-driven simulation environment for the T-BTMA protocol, particularly, of interest to overlapping-cell scenario. The theoretical results and the simulation results have a good match; therefore, the simulation results validate the approximations made in the analysis.

4) Comparing our analytic results to other researchers' simulation results, we show that our model may be adopted to variant MAC schemes with RTS/CTS mechanism.

5) Further, we study the power-controlled T-BTMA protocol combined with the RTS and the busy-tones, and extend our analytical model to this type of networks.

1.6 Thesis Outline

This thesis is concerned with the performance evaluation of T-BTMA protocol in ad hoc networks with or without power control. This thesis consists of four chapters as follows:

Chapter 2: Performance Analysis. In this chapter, two versions of DBTMA protocol in ad hoc networks are reviewed. We address weaknesses of the protocol and propose a modified protocol called as T-BTMA. Afterward, we detail the system model. Then, we propose a mathematical model to measure performance in terms of channel throughput. The multi-cell interference from ongoing transmissions is taken into account. Finally, we present more realistic simulations to validate our analysis.

Chapter 3: Analysis of a Power Controlled T-BTMA Protocol. This chapter focuses on how to bring the concept of power control into T-BTMA. A mathematical model is presented and the multi-cell interference from ongoing transmissions is taken into consideration in power-controlled networks. The numerical results show that the proper power control can lead to ideal channel throughput under heavy traffic loads.

Chapter 4: Conclusion. We conclude this thesis with a summary of the main contributions and future works.

Chapter 2

Performance Analysis

2.1 Introduction

As we have explained in the previous chapter, DBTMA is one of the protocols considered for ad hoc networks. In this chapter, we describe this protocol in detail and discuss its shortcomings including the deficiencies of the corresponding analytical model. Then, we propose a modification to this protocol and this new version is referred to as Triple-BTMA (T-BTMA). Following that, we provide a performance analysis of T-BTMA and present some numerical results.

2.2 The DBTMA protocol

In this section, we will introduce the DBTMA protocol. We will describe both the early and late versions to be referred to as DBTMA(98) and DBTMA(02), respectively.

First we explain the DBTMA(98) protocol. In DBTMA(98) version, the total available bandwidth is split into two sub-channels, as a data channel for transmissions of data packets and a control channel for transmissions of RTS and CTS packets. Furthermore, two sine-wave busy-tone signals, BTt, BTr, are inserted to the channel at different frequencies with enough spectral separation (Figure 2.1). Figure 2.2-a shows the operation of this protocol. Whenever a node has a data packet to send, it senses the channel for presence of BTr signals. The absence of the signals indicates that there are no nodes receiving data packets within the range of the source node. If this is true, then, the source sends an RTS packet to the destination but it does not set up any busy tone signal at this time. When the destination successfully receives the RTS packet, it senses for any BTt signals. The absence of the BTt signals indicates that there are no nodes transmitting data packets within the range of the destination. If this is true, the destination acknowledges RTS with a CTS packet and sets up a BTr signal simultaneously to show that it is receiving a data packet. After receiving the CTS packet successfully, the source starts the data transmission and sets up a BTt signal to show that it is transmitting a data packet. BTt and BTr signals protect the transmission. The source and destination turn off their BTt and BTr signals at the end of reception to announce the completion of the transmission.

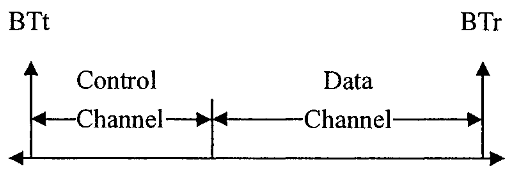


Figure 2. 1 Frequency chart of DBTMA protocol (refer to [4],[12])

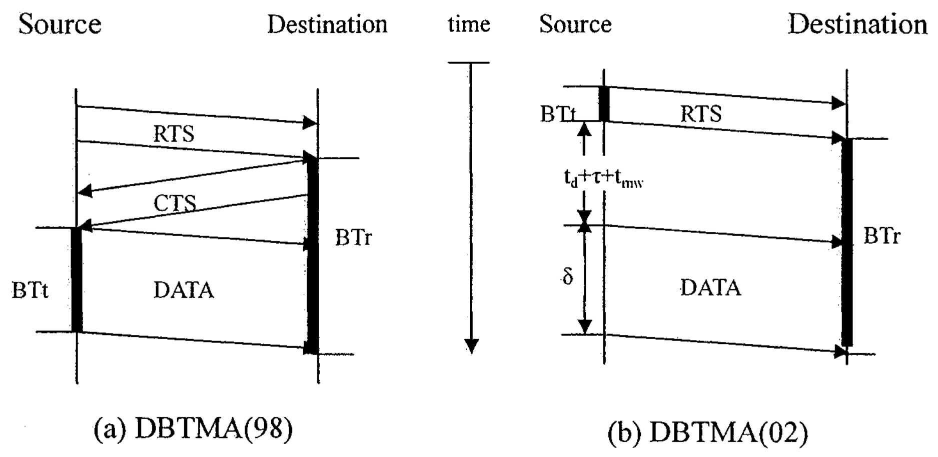


Figure 2. 2 The time lines of DBTMA(98) and DBTMA(02) protocols

Next, we describe the DBTMA(02) protocol, which is the modified version of DBTMA(98). The time line of this protocol is shown in Figure 2.2-b, with the following parameter definitions:

t_d: busy tone detection delay.

τ: propagation and mechanical delays.

γ: transmission time of an RTS packet.

δ: transmission time of a data packet.

t_{mw}: mandatory waiting time,2τ.

As may be seen, first, the source sets up and turns off its BTt signal at the beginning and the end of RTS transmission respectively. The presence of the BTt signal is to increase the probability of successful transmission of an RTS packet. Second, the destination does not send a CTS packet in response to an RTS packet, and it only sets up its BTr signal to indicate that it is waiting for a data packet. Finally, a mandatory waiting time (t_{mw}) is added into the process in order to abort all possible RTS requests in the range of the source. As may be seen, a major difference between DBTMA(98) and (02) versions is the absence of BTt signal during the transmission of data packet in (02) version. The objective of this change is to increase the probability of exposed terminal transmissions.

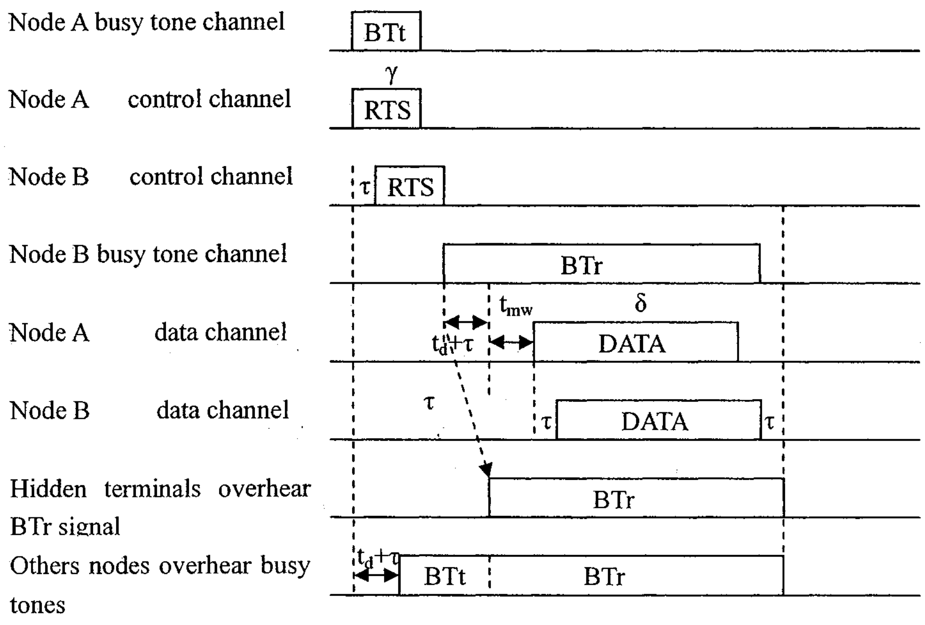


Figure 2. 3 The detailed time line of DBTMA(02)

Figure 2.3 presents more detailed time line of this protocol. As may be seen, the average duration of a busy period that carries a successful transmission is given by $T_s = \delta + \gamma + t_d + 6\tau$ and the average duration of a busy period that carries an unsuccessful transmission is given by $T_f = \gamma + \tau + \frac{t_d}{2}$ [3].

2.3 Main Weaknesses of DBTMA(02) protocol

In this section, we explain the main weaknesses of DBTMA(02) protocol as well as of its modeling. In the following, we assume that each node is served by the communication channel of its own home cell.

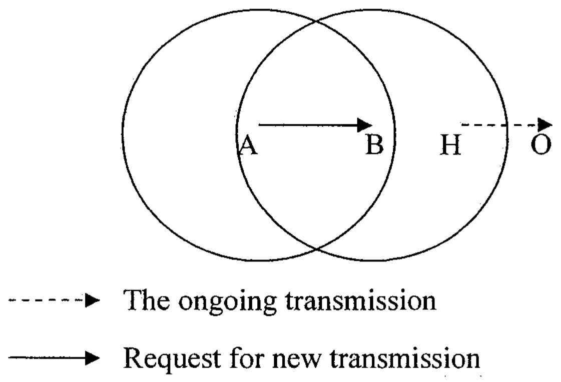


Figure 2. 4 Destruction of a new transmission by an ongoing transmission

First, we describe the weaknesses of DBTMA(02). Although DBTMA (02) protects RTS transmissions and data receptions, data transmissions suffer from interference. Figure 2.4 illustrates destruction of a data transmission by an ongoing transmission. The circles in the figure show the home cells of nodes A and B. We also assume an ongoing transmission from H to O at the time that A would like to initiate a transmission to B. H is inside the home cell of B but not of A, and O is outside of both home cells of A and B. During the ongoing data transmission from H to O, A and B will not detect any busy tone. If, at this time, A sends its RTS packet to B to request a channel, this channel request will be successful because RTS packets and data packets are transmitted on different channels. However, the consequent data transmission from A will be destroyed by H's data transmission at B. This collision or interference of data is inevitable. The cause of the collision is the removed BTt signal from DBTMA(98). Therefore, we propose to reinstate this busy tone to protect data transmission. To differ from BTt, we refer to the reinstated busy tone as BTd (data transmit busy tone). The modified DBTMA (02) is named as Triple-BTMA (T-BTMA). The following Figure 2.5 shows the time line of this proposed protocol.

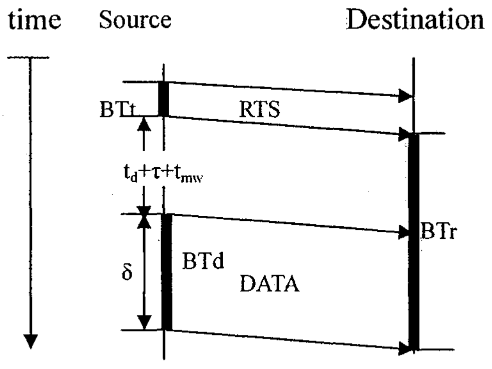


Figure 2. 5 The time lines of T-BTMA

An analytical model of DBTMA(02) has been presented in [3]. Next, we discuss the shortcomings of this model. First, the authors incorrectly borrow the network model of BTMA [6], which applies in centralized networks. In BTMA, the network model assumes that a base station is in the range of all terminals so that all terminals can hear the busy tone emitted by the base station, while hidden terminals occur when two nodes are within the range of the base station but out of range of each other (see Figure 1.5). In [3], the system model simply assumes that "all nodes are within the range of each other"; thus, all nodes can sense busy tone signed by any node in the network. However, this assumption implies that there are no hidden terminals in the network, which is the problem that they are trying to solve in the first place. Second, based on the improper network model, they inaccurately equate the probability of a successful data transmission to the probability of successful RTS transmission. This assumption can be valid only if there is no interference from ongoing transmissions in overlapping cells. However, this interference is the key factor in determining whether a data transmission in multi-cell networks will be successful. The objective of our work is to present an analytical model for the T-BTMA protocol which reinstates BTd signal to the DBTMA (02) version.

2.4 System Model

In this section, we introduce the system model for the T-BTMA protocol. It is assumed that the mobiles in the network have low mobility, such as pedestrians; hence, we do not consider the effect of node's mobility in the system.

We assume that each node is at the center of an imaginary cell defined by its transmission range. Therefore, the network will consists of many overlapping cells. Each channel has three states: transmission (Tx), reception (Rx) and idle (Idle). We assume that the wireless network consists of an infinite number of nodes and all nodes are identical. Though the wireless network has infinite number of channels, not all the channels may be active simultaneously due to overlapping of the channels. It is assumed that the total traffic generated by the nodes in cell i follows a Poisson distribution with aggregate rate of Ω_i,

$$P(k)=e^{-\Omega_i t}\frac{(\Omega_i t)^k}{k!} \qquad (1).$$

We note that the above result includes both new packets as well as retransmissions. The nodes do not implement any power control and therefore the radius of transmission range is R. The node locations are randomly distributed according to the uniform distribution with node density η nodes per unit area. The destination nodes for transmissions are also randomly chosen according to a uniform distribution. Since the number of nodes in a region is proportional to its area, the probability distribution function of the distance between the source and destination is given by

$F(\ell)$=Prob(distance between source and destination is less than ℓ)

$$=\frac{\pi\ell^2\eta}{\pi R^2\eta}=\frac{\ell^2}{R^2} \qquad (0\le\ell\le R) \qquad (2).$$

Taking the derivative of $F(\ell)$ with respect to ℓ, we have the probability density function (pdf) of the distance,

$$f(\ell)=2\ell/R^2 \qquad (0\le\ell\le R) \qquad (3).$$

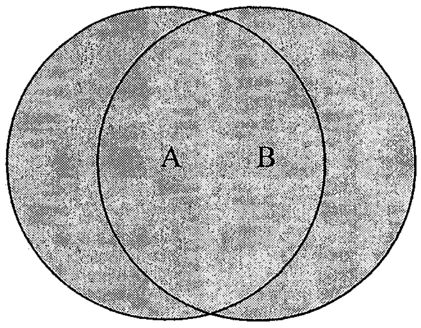


Figure 2. 6 The effective transmission area of a transmit-receive pair

We define effective transmission area as the combined region covered by the home cells of transmit and receive nodes, see Figure 2.6. The effective transmission area is given by

$$S^{Tx} = \pi R^2 + [Lens(R, -\frac{\ell}{2}) - Lens(R, \frac{\ell}{2})] \qquad (0 \le \ell \le R)$$

where the expression in "[]" is given by Eq. A2 in Appendix A. The minimum area of a Tx is "1" cell area when a destination and a source are very close to each other. And the maximum area of a Tx occurs when a destination is located at the edge of the transmission range of the source. The average effective Tx area approximates to1.41 times area of one cell.

A collision as defined by IEEE 802.11 occurs when two or more nodes within the transmission range of a source send RTS packets at the same time. Thus collisions may only occur during Contention Window (CW). Therefore, our model captures the collisions at source nodes.

Next, we introduce the following notation to be used in the later analysis:

i: node i, for example, A denotes node A.

$\bar{i}$: denotes all the nodes that are resident in home cell of i, for example, $\overline{B}$.

η: node density per unit area.

Λ_η: mean packet generation rate per unit area (this rate includes both new packets as well as retransmissions).

S_i: coverage area of home cell of node i.

Ω_i: mean packet generation rate in home cell of i, $\Omega_i = \Lambda_\eta \cdot S_i$.

R: transmission radius of a node.

ℓ: distance between source and destination nodes during a transmission.

T_s: average duration of a busy period that carries a successful transmission.

T_f: average duration of a busy period that carries an unsuccessful transmission.

2.5 Performance Analysis of T-BTMA

In this section, we present a performance analysis of T-BTMA that takes into account multi-cell overlapping interference.

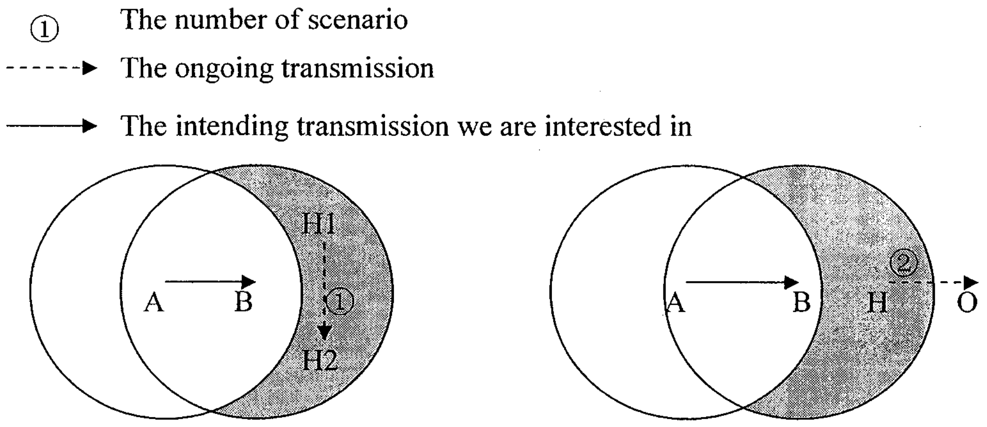
We assume that A would like to transmit a data packet to B (see Figure 2.7), where the two circles correspond to the home cells of A and B. A listens to the channel to make sure that there is neither BTt nor BTr signal. However, there may be factors preventing channel from being initialized successfully. We identify the following two scenarios of multi-cell overlapping interference from ongoing transmissions.

Scenario I, ongoing transmission in the shaded area

As shown in Figure 2.7-a, assume that there is an ongoing data transmission from node H1 to H2. Since this transmission is hidden from A, it transmits an RTS to B. B is able to receive this RTS packet because RTS packets and data packets are transmitted on different channels. However, B will not turn on its BTr signal due to detection of H2's BTr signal or H1's BTd signal. Therefore, channel initialization fails.

Scenario II, outgoing transmission in the shaded area

As shown in Figure 2.7-b, assume that node H is currently sending a packet to a destination O, outside of the shaded area. Again B will not be able to turn on its BTr signal due to hearing the H's BTd signal.


① The number of scenario
------► The ongoing transmission
———► The intending transmission we are interested in
H1
A
B
①
H2
A
B
②
H
O

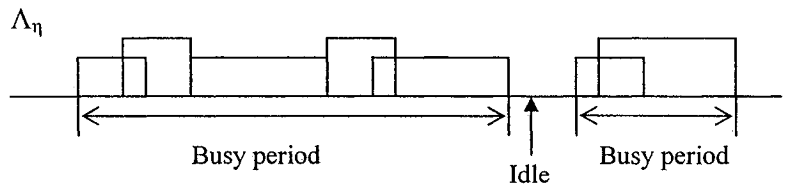
(a) The ongoing transmission in shaded area. The source and the destination of the ongoing transmission are both inside shaded area.

(b) The outgoing transmission in shaded area. Only the source is inside shaded area.

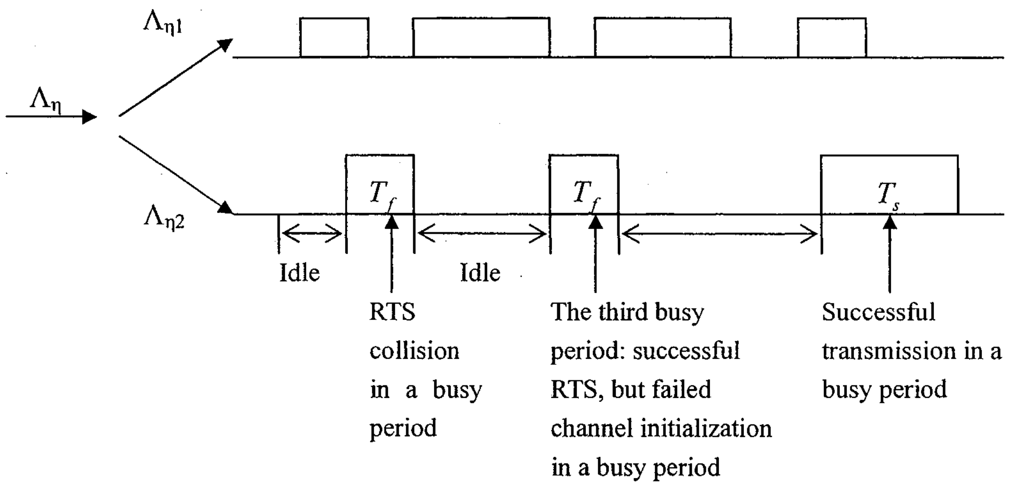
Figure 2. 7 Multi-cell overlapping interference

In most of the research works in this area, the above multi-cell overlapping interference is neglected in the performance evaluation of MAC protocols for ad hoc networks. These works use a signal-channel mode in order to avoid the complexity introduced by this kind of interference. However, our discussion of the above two scenarios shows clearly that such multi-cell overlapping interference affects the channel initialization. Although an RTS packet can be received successfully, it does not mean the channel is initialized successfully. The analysis to be presented later on will take this effect into account.

In single-channel networks, the status of channel alternates between busy and idle periods. A busy period is initiated by an arrival to the channel when the channel is sensed idle. The busy period assumes two forms, a busy period that carries a successful transmission or a collision. In multi-cell ad hoc networks, there can be several simultaneous transmissions in the network or several consecutive transmissions; therefore, the busy period of such network may be formed by several single-channel busy periods, as shown in Figure 2.8-a. To solve this complexity, we split Poisson arrival processes into a number of independent streams. We note that each stream itself is a Poisson process. In Figure 2.8-b, we present an example of splitting the Poisson arrival process into two independent streams with average arrival rates $\Lambda_{\eta1}$ and $\Lambda_{\eta2}$. From this consideration, the communication channel served by a node is assumed to be a dynamic channel. This assumption is in agreement with the transient and dynamic feature of ad hoc networks. These dynamic channels can be divided into several independent groups with different arrival rates. When a packet arrives, the possible actions are to assign it a free channel or to block it if no channels are available [28]. When a packet departs, the system is allowed to reassign the channels in use to a cell that has an attempt [28]. In this way, the dynamic channels are allocated through the network range. Over a long period of time, the status of a channel alternates between busy and idle periods.



(a) Channel status in ad hoc networks


$\Lambda_{\eta 1}$
Λ_η
$\Lambda_{\eta 2}$
T_f
T_f
T_s
Idle
Idle
RTS collision in a busy period
The third busy period: successful RTS, but failed channel initialization in a busy period
Successful transmission in a busy period

(b) Random splitting and a third busy period in T-BTMA

Figure 2. 8 Busy and idle periods

Within the T-BTMA protocol, the status of a dynamic channel has a third form of busy period. As discussed in the above two scenarios, the third busy period consists of a successful RTS transmission time, (see Figure 2.8-b), but this RTS does not successfully initialize the channel. Although this busy period does not contain conflicting RTS messages, the effect on the channel is considered as same, and numerically, it is similar to T_f. Moreover, this busy period is also followed by an interval in which the channel is sensed idle. Therefore, we can refer the consecutive idle and busy periods in a dynamic channel as a cycle. We let

$\overline{B}_i$ = mean busy period of home cell i during a cycle;

$\overline{I}_i$ = mean idle period of home cell i during a cycle.

$\overline{T}_i$ =mean busy period of home cell i that carries a successful transmission.

P_i^S = the probability of a successful transmission in home cell i.

Next, we will derive the throughput seen by a node. During this derivation, we will study a transmission with source, and destination nodes as A, B respectively. As shown in Figure 2.9, A and B are separated by a distance of ℓ.

From renewal theory, probability that cell i is busy is given by,

$$\sigma_i = \frac{\overline{B}_i}{\overline{B}_i + \overline{I}_i} \tag{4}$$

Each busy period may carry a single successful transmission or a collision. Thus,

$$\overline{T}_i = \delta \cdot P_i^S,$$

where δ is transmission time of a data packet.

Then, throughput of a channel serving cell i is given by,

$$\rho_i = \frac{\overline{T}_i}{\overline{B}_i + \overline{I}_i} = \frac{\delta P_i^S}{\overline{B}_i + \overline{I}_i} \tag{5}$$

Now, we apply Eq. (5) to cell A, thus,

$$\rho_A = \frac{\delta P_A^S}{\overline{B}_A + \overline{I}_A} \tag{6}$$

Next, we will determine each of the quantities in the above. First, we determine the mean idle period of cell A during a cycle. Since packet arrivals are according to a Poisson process with Ω_A, the packet inter-arrival time is exponentially distributed with the same parameter value. From the memoryless property of exponential distribution, mean idle period equals to the mean inter-arrival time,

$$\bar{I}_A = \frac{1}{\Omega_A} \qquad (7).$$

The mean busy period is given by

$$\bar{B}_A = P_A^S T_s + (1 - P_A^S) T_f . \qquad (8).$$

Second, we determine P_A^S, the probability of a successful data transmission in home cell of node A. In the T-BTMA, since RTS and data packets are transmitted on their own channels, ongoing data transmissions will not destroy RTS transmissions. Thus, a successful RTS transmission can be treated as a successful RTS reception. Then, the destination will turn on its BTr signal when it both has a successful RTS reception and there is no multi-cell overlapping interference. The latter happens when nodes in the shaded area are idle (Figure 2.9). Let B1 denote the shaded area in Figure 2.9, which corresponds to $\overline{B1} = \bar{B} - (\bar{A} \cap \bar{B})$. Once the BTr signal is turned on, the consequent data transmission is guaranteed to be collision free. Therefore,

$$P_A^S = \text{Pr(successful channel initialization).}$$

In detail, the channel initialization is successful if the following two independent conditions are satisfied, C1: No collision in cell A during RTS transmission.

C2: No multi-cell overlapping interference in cell B.

Therefore we have,

$$P_A^S = \Pr\{C1\}\Pr\{C2\} \quad (9).$$

Since packet arrivals are according to a Poisson process, and the vulnerable period for RTS packets are $t_d + \tau$ seconds,

$$\Pr\{C1\} = e^{-\Omega_A(td+\tau)}. \quad (10).$$

We note that in [3], the probability of a successful data transmission is inaccurately equated to the probability of successful RTS transmission, $P_A^S = \Pr\{C1\}$. The effects of multi-cell overlapping interference are neglected in their throughput performance model. Now, we take into account the multi-cell overlapping interference in our analysis. Let us define P_{B1}^I=Prob(nodes in B1 are idle given that the distance between A and B is ℓ). Thus,

$$\Pr\{C2\} = P_{B1}^I \quad (11)$$

Substituting Eq.(10) and (11) into (9), we have

$$P_A^S = e^{-\Omega_A(td+\tau)} \cdot P_{B1}^I \quad (12)$$

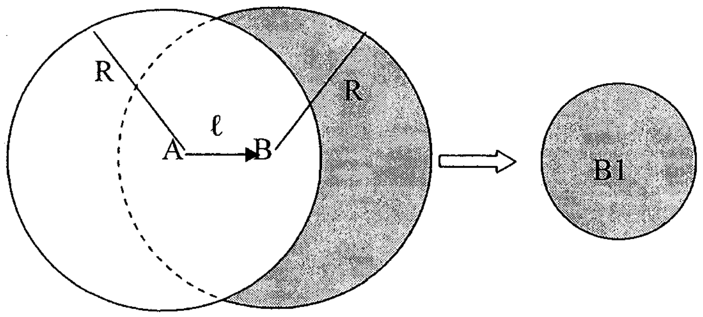


Figure 2. 9 Analysis of the probability of successful transmission in home cell of A

The probability P_{B1}^{I} is akey to this performance analysis because it takes into account the effect of multi-cell overlapping interference. In most theoretical analyses, this probability is never included due to those models being built on single-channel models. In this work, we will determine P_{B1}^{I} through a simplification. We replace the shaded region B1 with a circular cell of equal area, see Figure 2.9. This may be regarded as a virtual cell without hidden terminals, thus,

$$P_{B1}^{I} = 1 - \sigma_{B1} \qquad (13),$$

where σ_{B1}=Prob(cell B1 is busy)

Next applying Eq. (4), we have,

$$\sigma_{B1} = \frac{\overline{B}_{B1}}{\overline{B}_{B1} + \overline{I}_{B1}} \qquad (14).$$

Substituting Eq.(14) into (13),

$$P_{B1}^{I} = 1 - \frac{\overline{B}_{B1}}{\overline{B}_{B1} + \overline{I}_{B1}} = \frac{\overline{I}_{B1}}{\overline{B}_{B1} + \overline{I}_{B1}} \qquad (15).$$

The average idle period within virtual cell B1 is given by

$$\overline{I}_{B1} = \frac{1}{\Omega_{B1}} \qquad (16).$$

We are unable to determine the exact expression for $\overline{B}_{B1}$; therefore, we approximate it as follows

$$\overline{B}_{B1} = P_{B1}^{S} T_s + [1 - P_{B1}^{S}] T_f \qquad (17),$$

where

$$P_{B1}^{S} = e^{-\Omega_{B1}(td+\tau)} \qquad (18)$$

The above approximation will be justified through simulation.

Next, we determine the area of the shaded region, S$_{B1}$, which is needed in determining Ω_{B1} since $\Omega_{B1} = \Lambda_\eta S_{B1}$. Assume the two cells are identical with the same R in the non-power-controlled network. Thus

$$S_{B1} = Lens(R, -\frac{\ell}{2}) - Lens(R, \frac{\ell}{2}) \qquad (19).$$

The function $Lens(\bullet)$ is given in Appendix A. Substituting Eqs. (16)- (19) into Eq.(15), we obtain

$$\begin{aligned} P_{B1}^{I} &= \frac{\frac{1}{\Omega_{B1}}}{e^{-\Omega_{B1}\cdot(td+\tau)}(T_s - T_f) + T_f + \frac{1}{\Omega_{B1}}} \\ &= \frac{1}{\Lambda_\eta S_{B1}[e^{-\Lambda_\eta S_{B1}\cdot(td+\tau)}(T_s - T_f) + T_f] + 1} \end{aligned} \qquad (20).$$

Now, we complete the derivation of the probability of a successful data packet transmission by substituting Eq. (20) into Eq. (12), we have,

$$\begin{aligned} P_A^S &= e^{-\Lambda_\eta \pi R^2(td+\tau)} \cdot P_{B1}^I \\ &= \frac{e^{-\Lambda_\eta \pi R^2(td+\tau)}}{\Lambda_\eta S_{B1}[e^{-\Lambda_\eta S_{B1}\cdot(td+\tau)}(T_s - T_f) + T_f] + 1} \end{aligned} \qquad (21).$$

Finally, we obtain the channel throughput of the T-BTMA protocol by substituting the Eqs. (7), (8) and (21) into Eq. (6), which yields

$$\rho_A = \frac{\delta P_A^s}{P_A^s T_s + (1 - P_A^s)T_f + \bar{I}_A}$$

$$= \frac{\delta e^{-\Lambda_\eta \pi R^2 (td+\tau)}}{(T_s - T_f) e^{-\Lambda_\eta \pi R^2 (td+\tau)} + (T_f + \frac{1}{\Lambda_\eta \pi R^2})\{\Lambda_\eta S_{B1}[(T_s - T_f) e^{-\Lambda_\eta S_{B1}(td+\tau)} + T_f] + 1\}} \quad (22).$$

where S_{B1} is given by Eq. (19): $S_{B1} = Lens(R, -\frac{\ell}{2}) - Lens(R, \frac{\ell}{2})$.

The basic equation for the throughput ρ_A is expressed in terms of δ, T_s, T_f, $\bar{I}$, td, τ, R, Λ_η and ℓ. As it has been discussed in the previous section, ℓ is a random variable and its pdf is given by Eq.(3). Thus, the average throughput is obtained by unconditioning ρ_A with pdf of ℓ,

$$\bar{\rho}_A = \int_0^R f(\ell) \rho_A d\ell \quad (23).$$

2.6 Numerical and Simulation Results

In this section, we present some numerical as well as simulation results to demonstrate validity of our approximations.

2.6.1 Theoretical Results

First, we give numerical results regarding the network throughput. In order to compare the results with those of [3], we assume the same network parameter values as in [3].

- Channel data rate = 1Mb/s
- Data packet length = 4096 b, δ = 4.096ms

- RTS packet length = 200 b, γ = 0.2ms
- Busy tone detection delay td=0.01ms
- Maximum one-way propagation delay τ=0.12μs.

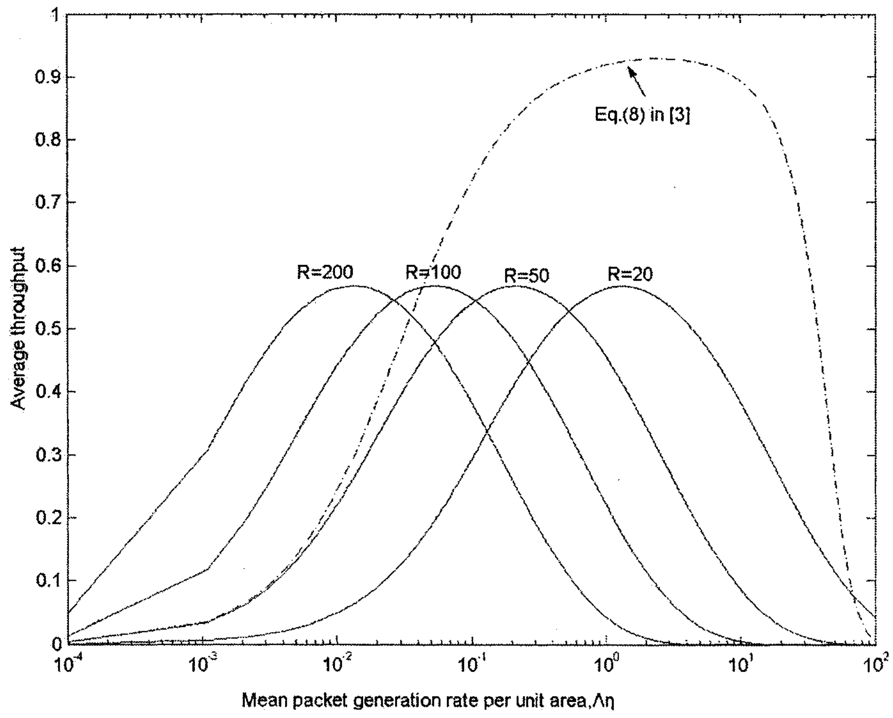


Figure 2. 10 The average throughput with different transmission ranges

Figure 2.10 presents the average throughput $\overline{\rho}_A$ versus the offered traffic Λ_η for different transmission ranges R. We note that the maximum throughputs of variant transmission ranges are same. As the transmission range increases; the curve shifts to the left. The dash curve corresponds to the throughput results in [3] (equation 8),

$$S = \frac{e^{-\lambda(td+\tau)}}{e^{-\lambda(td+\tau)}(T_s - T_f) + T_f + 1/\lambda},$$

where S denotes as channel throughput, λ denotes the mean aggregate generation rate, and the remaining parameters share the same denotations in our model. As may be seen, their model assumes that there are no hidden terminals in the networks, and therefore their model fails to take into account multi-cell interference. In [3], the numerical and simulation results coincide with each other, but simulations are also performed without hidden terminals. In their simulation, the system had 20 nodes with 35 meters of transmission range in a network that has spreaded to an area of 50 x 50 m^2, which implies that there are no hidden terminals. This is the reason why the throughput results in [3] is higher than ours.

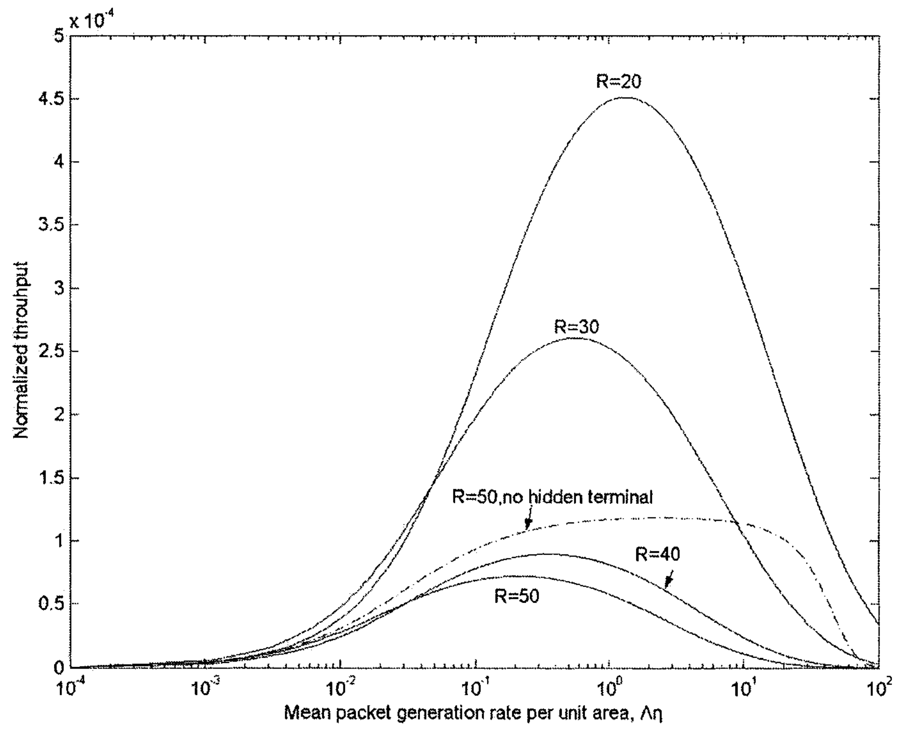


Figure 2. 11 Normalized throughput with different transmission ranges

In Figure 2.11, we plot the normalized throughput versus the offered traffic Λ_η with different transmission ranges to further observe its impact on the throughput.

We define the normalized throughput as,

Normalized throughput= Average throughput/ Area of a cell

$$= \frac{\overline{\rho}_A}{\pi R^2}.$$

From these results, it is all evident that the reduced transmission range improves the throughput by lowering the probability of collision and interference. However, reducing transmission range too much will also induce problems, such as increasing the number of hops to reach the destination; therefore, increasing the packet delay, and may even result in loss of network connectivity.

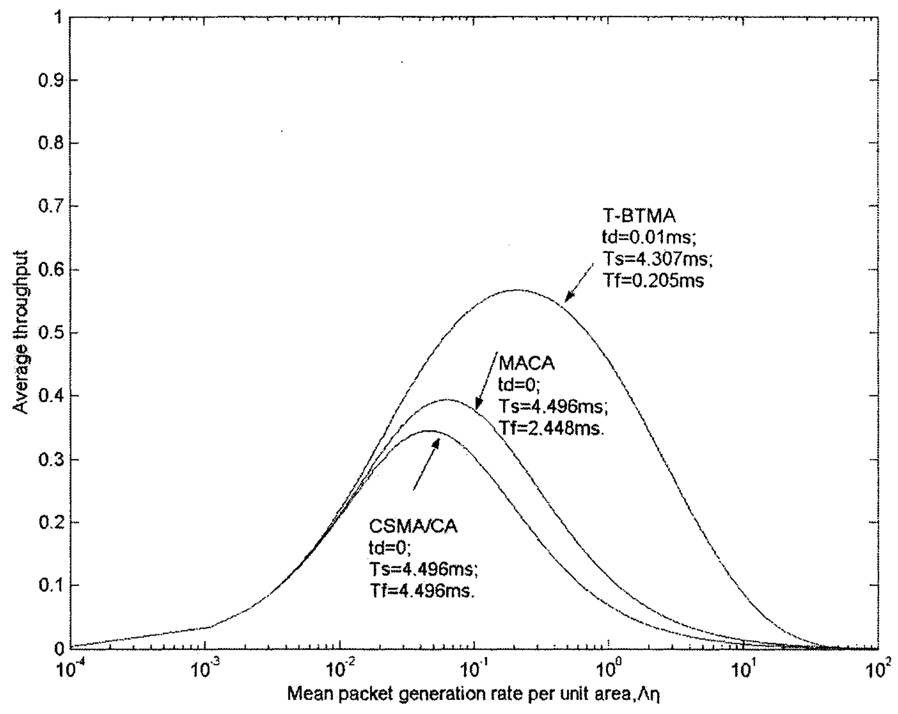


Figure 2. 12 The average throughput versus mean packet generation per unit area.

We may adopt our analysis to model other MAC protocols for ad hoc networks discussed in the introduction. Figure 2.12 shows the average throughput

$\bar{\rho}_A$ of T-BTMA, MACA and CSMA/CA protocols by substituting their own system parameters, such as td, T_s and T_f, into our mathematical model. According to studies of the size of collision windows, we approximate that td=0, Ts=4.496ms and T_f=4.496ms as the system parameters for the CSMA/CA protocol. The maximum throughput of numerical result reaches around 0.35, which matches the maximum throughput of simulation result provided by Tobagi and Kleinrock [6]. Similarly, we approximate that td=0, T_s=4.496ms and T_f=2.448ms for the MACA protocol. The maximum throughput of numerical result reaches around 0.4, which is also validated by existing simulation results in [2]. As desired, our methodology may be employed to evaluate the throughput for various RTS/CTS mechanism MAC protocols. In addition, as expected, the higher the ratio of Ts to Tf, the better is the performance. This ratio is important to design an efficient MAC protocol.

2.6.2 Simulation Model

Next, we describe the event-driven simulation program that we have built to validate the analytical results. The model simulates T-BTMA protocol by using VC/C++ language.

We assume a network area of a disk with radius of 200 meters. We assume a node population of 100, which are randomly placed over the disk. All the nodes have the same radio transmission range radius of 50 meters, which is the optimal maximum expected hop length as shown in [7]. The one-way propagation delay is assumed to be $0.01\,\mu s$ and the transmission speed is 1Mb/s. We made sure that each node has at least one neighbor; none of them is isolated. For each transmission, a source node is randomly chosen and then the destination is selected also randomly among its neighbors. In the simulation model, the new packets are generated independently from a Poisson distribution. This is achieved by generating packets with exponentially distributed interarival times. Collision and random retransmission are accounted for without further assumptions [27]. The backoff algorithms are used after every collision and re-sensing. Backoff times are exponentially distributed. Each of the simulation results represents 10 random runs. Each simulation run corresponds to 1800 packets. It has been observed that longer runs do not change the results. In each run, we collected statistics only from packet 200 through 1500 in order to eliminate transient portion of the simulation.

The throughput of each run is determined as,

$$Q_i = \frac{average\_transmission\_time\_of\_a\_datapacket \times number\_of\_packets}{total\_duration\_of\_a\_run}$$

The throughput of the system has been determined as,

$$Q = \frac{1}{r}\sum_{i=1}^{r} Q_i$$

where r is the number of runs.

A throughput value Q is said to be feasible or achievable if every node can send at an offered rate to its destination. Our simulation environment allows multiple data transmission simultaneously; therefore, the obtained results are the aggregated utilization of all channels. We let n denote the average number of concurrent transmissions that network may handle, and then we conservatively approximate the maximum number of concurrent transmissions by

$$n = (\text{Network area})/(\pi R^2).$$

Finally, the channel throughput result of the simulation is given by

$$\overline{Q} = \frac{Q}{n}.$$

2.6.3 Simulation Procedure of T-BTMA

Next, we describe the simulation process in more detail. Figure 2.13 presents the flowchart that outlines the operation of the simulation. First, we place randomly100 nodes according to the uniform distribution over the circular network area. Then we find the neighbors of each node. If one of these nodes has zero number of neighbors, we generate a new set of random nodes till each node has at least one neighbor. We need to generate exponentially distributed random variable for packet interarrival times and backoff times. Assuming that a random variable t is exponentially distributed, then, its probability distribution function (pdf) is given by,

$$F(t) = 1 - e^{-\lambda t}.$$

Solving for t,

$$t = -\frac{1}{\lambda}\ln[1 - F(t)]$$

We note that $1 - F(t)$ is uniformly distributed over the interval over $[0,1]$. Letting

$U = 1 - F(t)$, then,

$$t = -\frac{\ln U}{\lambda}.$$

Thus, we may generate an exponentially distributed random variable according to the above equation by generating uniformly distributed random variable. Next, for each value of the packet arrival rate, the simulation program is run 10 times. Figure 2.14 shows that the simulation has four event types: 1.new arrival, 2.departure, 3.sensing/re-sensing, 4.RTS transmission. This is a discrete event simulation and future events are kept in a system schedule table. This table is always kept sorted in ascending order according to the future event times. The simulation time advances from one event to the next event time. Simulation program always processes the event at the head of the schedule and the present time corresponds to this event time. After an event has been processed, it is removed from the head of the schedule. Then the simulation continues with the processing of the next event in the schedule. At the start of simulation, all the nodes are idle and the simulation begins with the scheduling of the arrival of first packet. A simulation run terminates after a fixed number of packets have gone through the system. And each event has its own operation and interaction of scheduling, as shown through Figures 2.15-2.18.

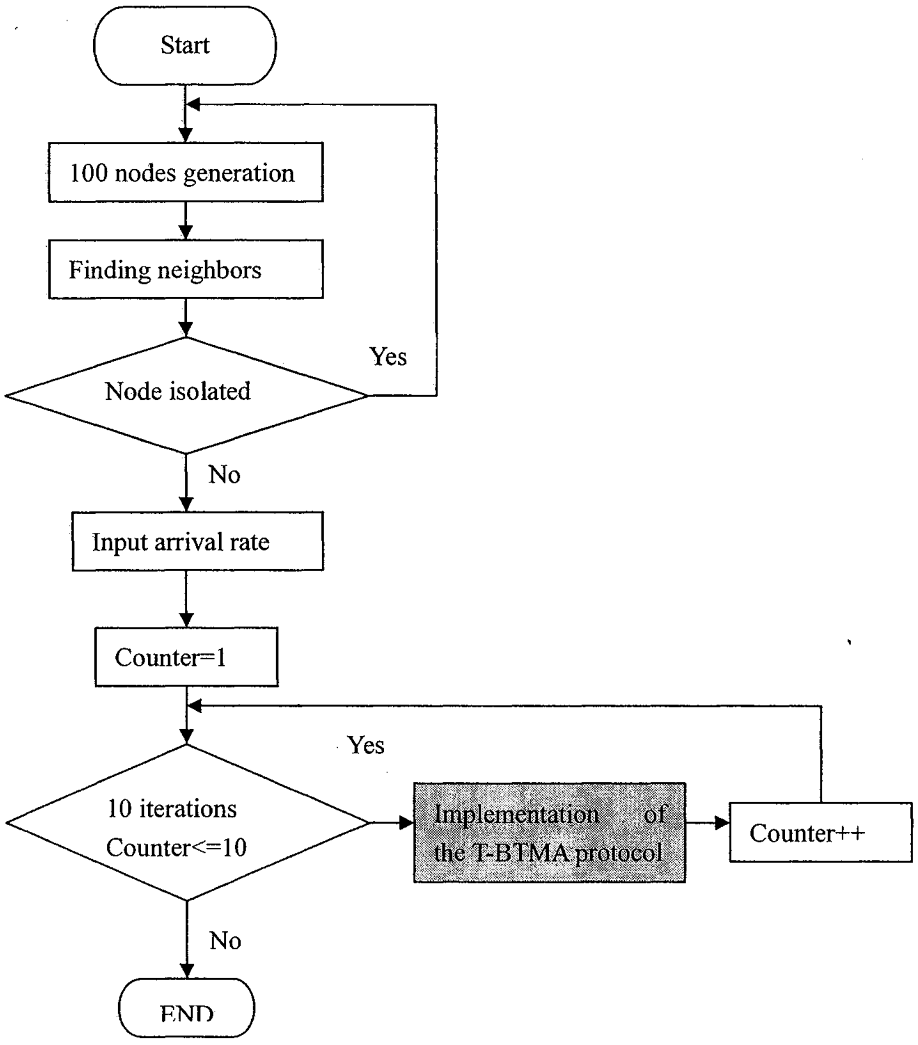


Figure 2. 13 Flowchart of the simulation

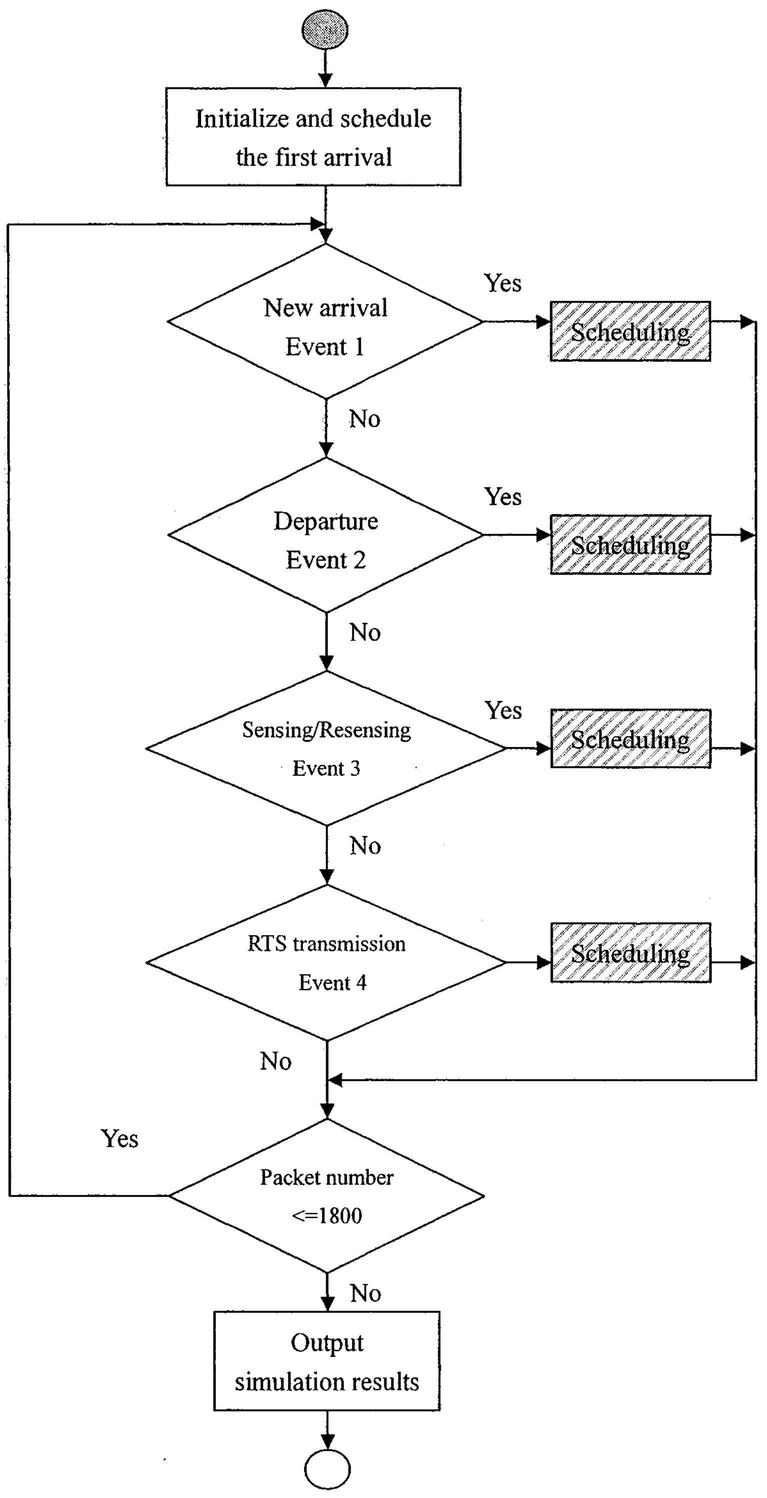


Figure 2. 14 Flowchart of the four event types of the simulation

Next, we describe how the simulation handles occurrence of each type of event.

I. Processing of an Arrival Event.

Each simulation run begins with scheduling of the first arrival. From there on, a new arrival can be only scheduled during the processing of new arrival event. When a new packet arrives, the arrival of the next packet is scheduled. There are two steps in processing the new arrival event. Step 1, when an arrival occurs, the RTS packet for the new arrival may be scheduled (see Fiugre2.15). This is done by checking the event types of the source's neighbor in the system schedule. If none of the neighbors have RTS transmission or data reception scheduled; then, the system schedules an RTS transmission for the source and ends the first step. If there are RTS transmissions among neighbors during the vulnerable time, $t_d + \tau$, the system will still schedule the RTS transmission but will flag a collision and finish the first step. If neither of the above two sub events is possible, then, system schedules sensing/re-sensing with an exponential backoff algorithm for the new arrival and then records the number of re-sensings. The system starts its Step 2 with scheduling a new arrival event according to an exponential distribution new packet arrival rate λ.

II. Processing of an RTS Packet Transmission Event.

When the system processes an event of RTS Transmission, (see Figure 2.16), it first checks the collision flag. If there is a collision flag, the system will schedule a re-sensing time with the exponential backoff algorithm and increment the re-sensing number. Then the processing of this event terminates. If the number of re-sensing reaches 50, the maximum number of re-sensing, the packet will be discarded. If there is no RTS collision, the system will schedule the data transmission (the departure event) after ensuring that no ongoing transmission can prevent the channel initialization. If all the above conditions are not met, the system will schedule a re-sensing time and increment the number of re-sensings. Following that, the processing of this event type terminates.

III. Processing of a Sensing/Resensing Event.

The processing of the sensing/re-sensing event, (see Figure 2.17), is same as the first step of processing the new arrival event, but ends without scheduling the next new arrival.

IV. Processing of a Departure Event.

When processing departure event, which means a successful data transmission, (see, Figure 2.18), the system places the source node back to the No-packet array, and calculates the total delay of the packet.

An array named "No-packet array" contains all the idle nodes which have no packets to transmit. Once a node has a packet, its ID will be removed from this array, and once the node finishes its transmission, its ID will be inserted back to the array. When the No-packet array is empty, it means that every node in the network has a data packet that waiting for transmission. If the above condition is satisfied, the system may be saturated, and the simulation terminates.

Following table shows the system scheduling table captured at the 0.36156485695th second. The first column of Table 2.1 contains the packet number. The second column contains the event type of each packet, "1" denotes the event type of new arrival, "2" denotes the event type of departure, "3" denotes the event type of sensing/re-sensing, and "4" denotes the event type of RTS transmission. And there is only one "1" in this table at any moment. The third column contains the scheduled time of each event, and the system scheduling is sorted by this column. The fourth and fifth column contain the IDs of source nodes and destination nodes, respectively. The sixth column contains the starting time of each event, and they are used to schedule the next event. The seventh column contains the collision flags of each packet, "0" is defined as collision free and "1" is defined a collision. In this table there is no collision at this moment. The last column contains the retransmission number of each packet; This value determines whether a packet is allowed further resensing. Each row corresponds to the record of a different packet. For example in the first row, at the 0.36156485695th second, the 347th packet which is sent by node 89 to node 60 is processed. The system first checks the event type and chooses the process of sensing/re-sensing to schedule this packet. If this packet is scheduled to resense the channel, the retransmission number "7" will be used to calculate the backoff time.

1	2	3	4	5	6	7	8
Packet Number	Event Type	Scheduled Time	Source Node	Destination Node	Starting time	Collision flag	Retrans-mission Number
347	3	0.36156485695	89	60	0.36094449695	0	7
353	4	0.36169960707	8	50	0.36148948707	0	0
354	1	0.36202015194	61	38	0.36202015194	0	0
332	3	0.36223065346	39	85	0.33998162447	0	7
349	3	0.36254640745	44	2	0.35943565001	0	5
348	2	0.36320780804	47	63	0.35889108804	0	0
350	2	0.36424521001	88	35	0.35992849001	0	0
351	2	0.36486044601	46	45	0.36054372601	0	0
352	2	0.36544971086	25	19	0.36113299086	0	0
341	3	0.36623072352	42	45	0.34933314535	0	6
269	3	0.60837228471	83	87	0.28720454238	0	11

Table 2. 1 The system scheduling table

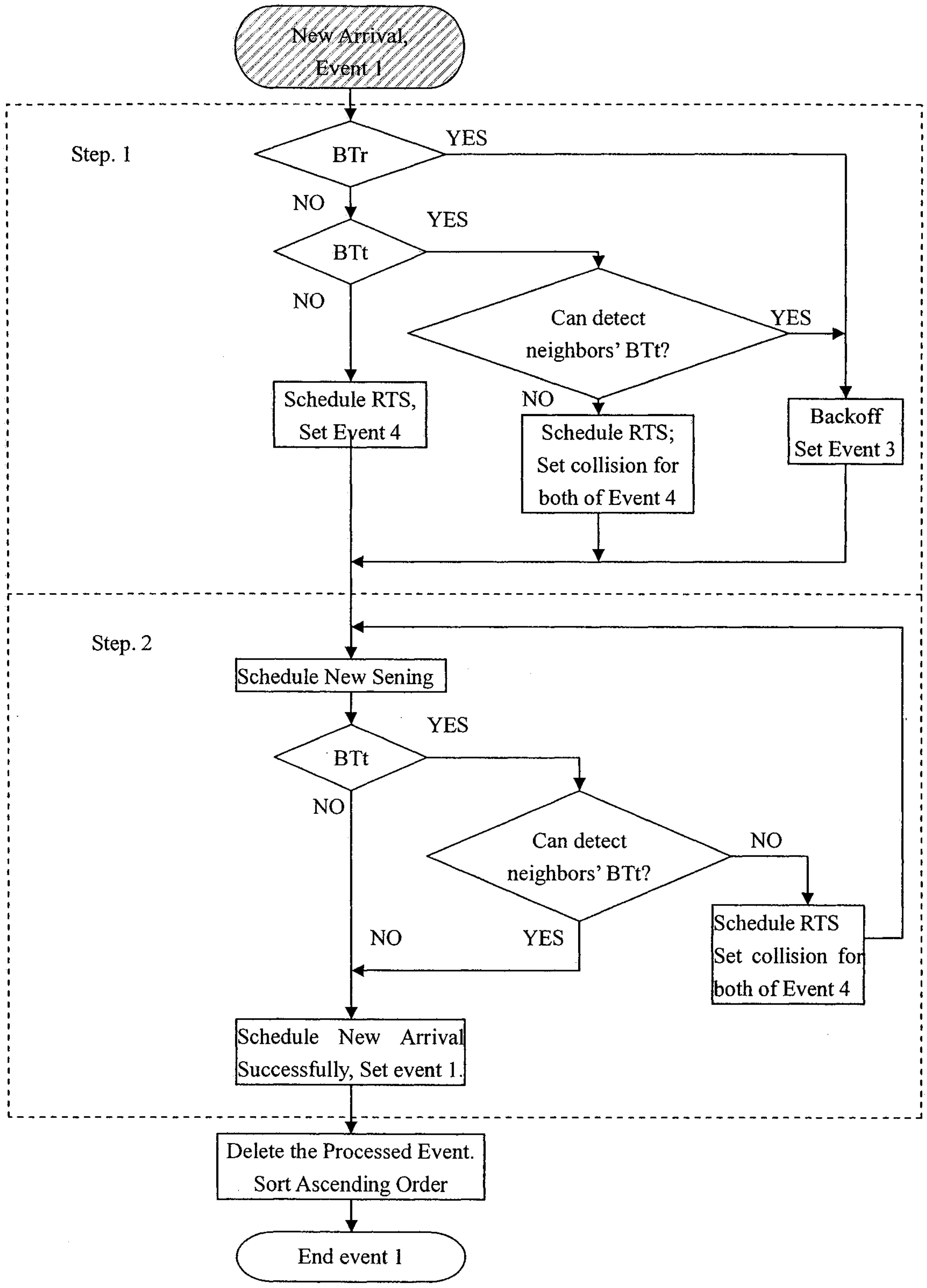

New Arrival, Event 1
Step. 1
BTr
YES
NO
BTt
YES
NO
Can detect neighbors' BTt?
YES
NO
Schedule RTS, Set Event 4
Schedule RTS; Set collision for both of Event 4
Backoff Set Event 3
Step. 2
Schedule New Sening
BTt
YES
NO
Can detect neighbors' BTt?
NO
YES
NO
Schedule RTS Set collision for both of Event 4
Schedule New Arrival Successfully, Set event 1.
Delete the Processed Event. Sort Ascending Order
End event 1

Figure 2. 15 Processing of a new arrival event

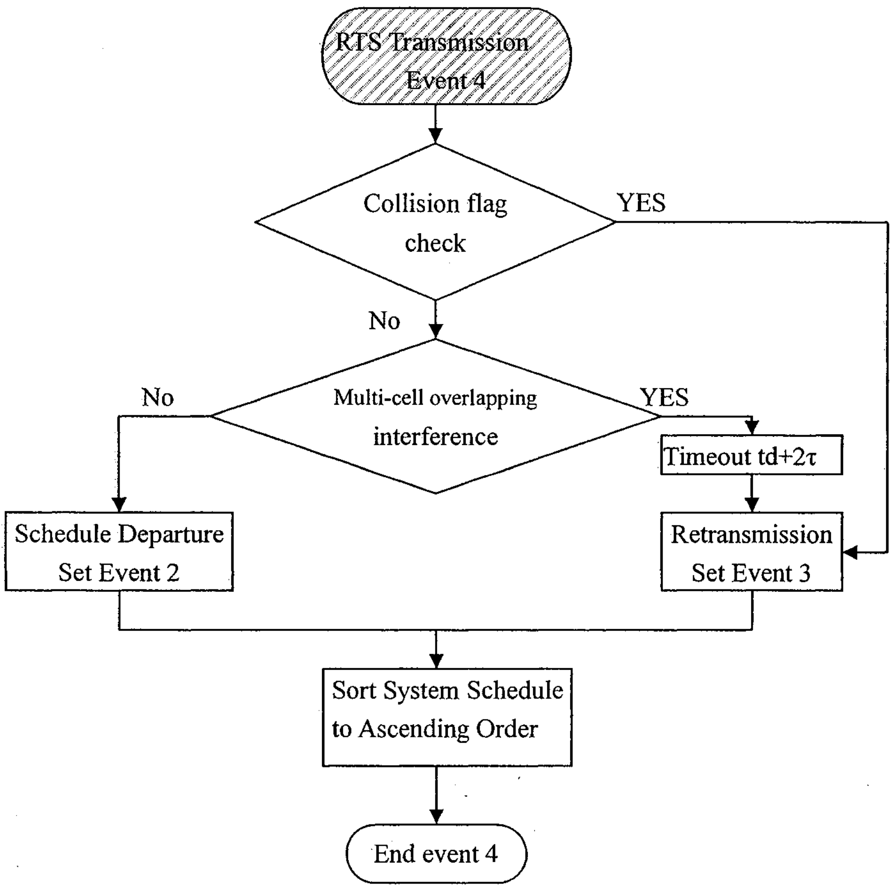

RTS Transmission Event 4
Collision flag check
YES
No
Multi-cell overlapping interference
No
YES
Timeout td+2τ
Schedule Departure Set Event 2
Retransmission Set Event 3
Sort System Schedule to Ascending Order
End event 4

Figure 2. 16 Processing of a RTS packet transmission event

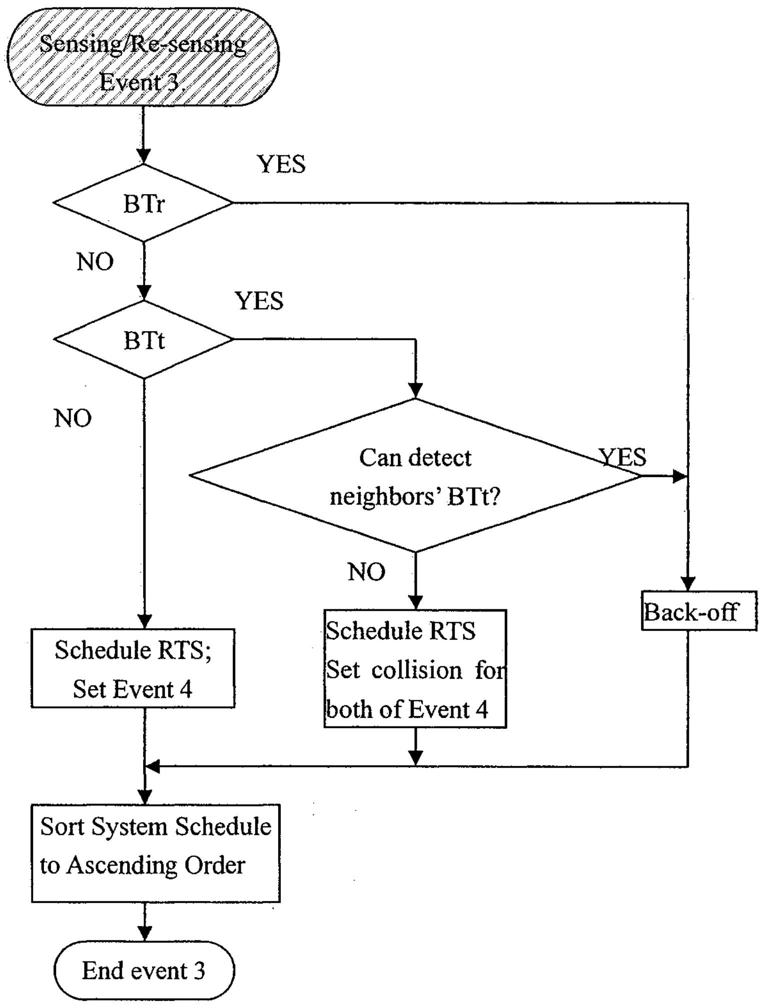

Sensing/Re-sensing Event 3
BTr
YES
NO
BTt
YES
NO
Can detect neighbors' BTt?
YES
NO
Schedule RTS; Set Event 4
Schedule RTS Set collision for both of Event 4
Back-off
Sort System Schedule to Ascending Order
End event 3

Figure 2. 17 Processing of a sensing/re-sensing event

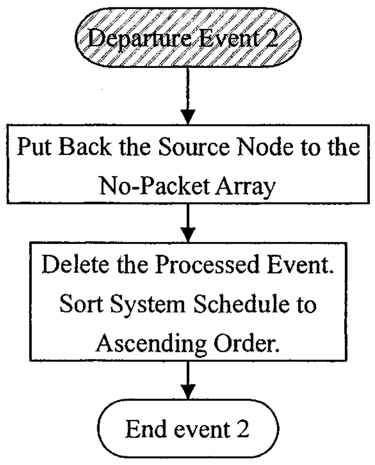

Departure Event 2
Put Back the Source Node to the No-Packet Array
Delete the Processed Event. Sort System Schedule to Ascending Order.
End event 2

Figure 2. 18 Processing of a departure event

2.6.4 Simulation Results

Figures 2.19 and 2.20 show the average throughput versus offered traffic load for the T-BTMA protocol. We find that the protocol saturates at an offered load of about load = 547 packets/s for a given achievable throughput $\overline{Q}$. The figure also presents the corresponding analytical results and as may be seen these two have a good match, and this validates the approximations made in the analysis.

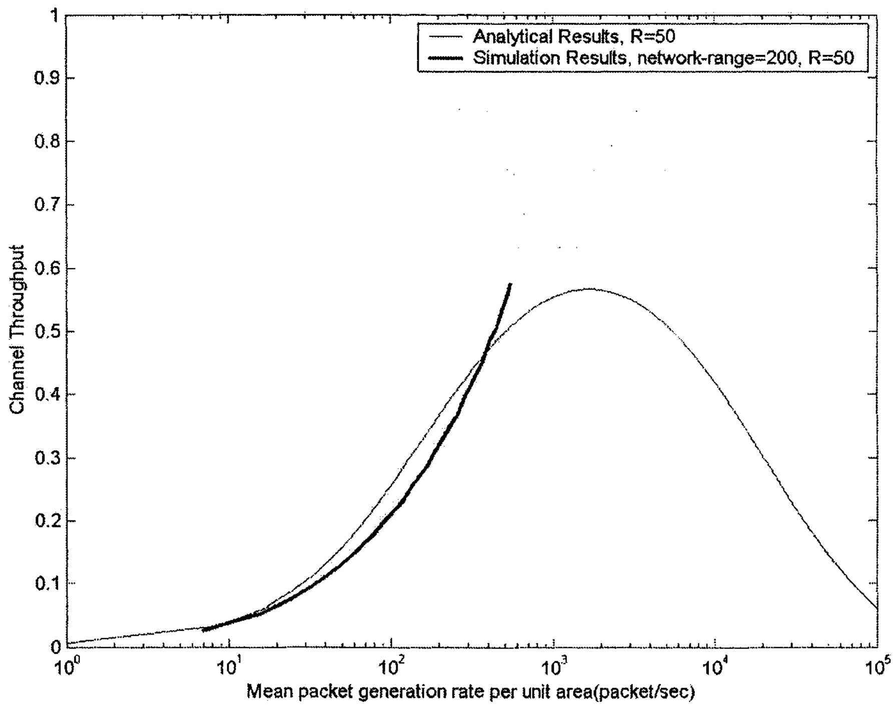


Figure 2. 19 Simulation and analytical results for throughput versus offered traffic (with logarithmic horizontal scale).

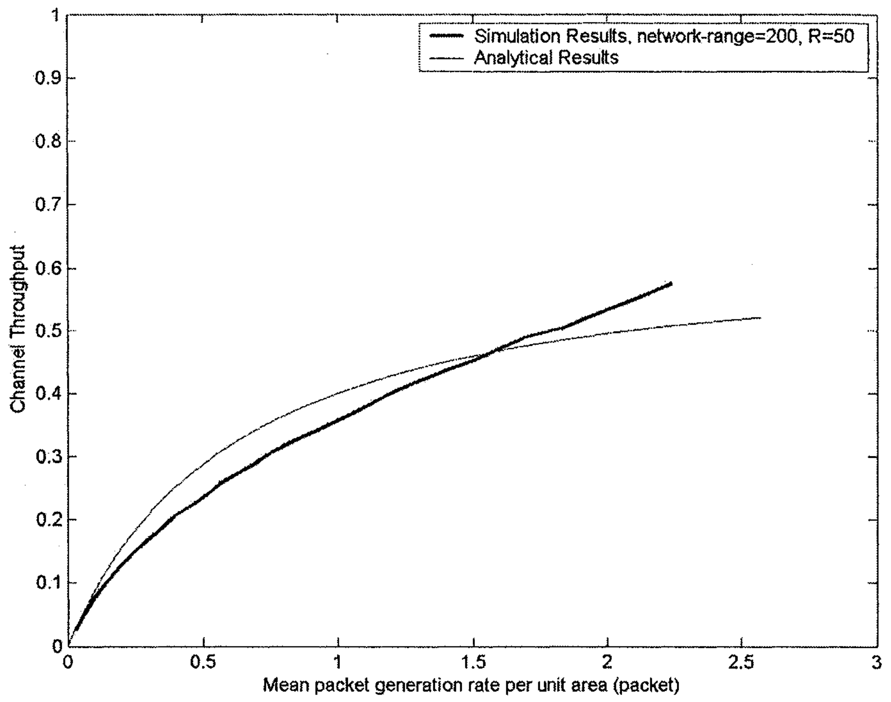


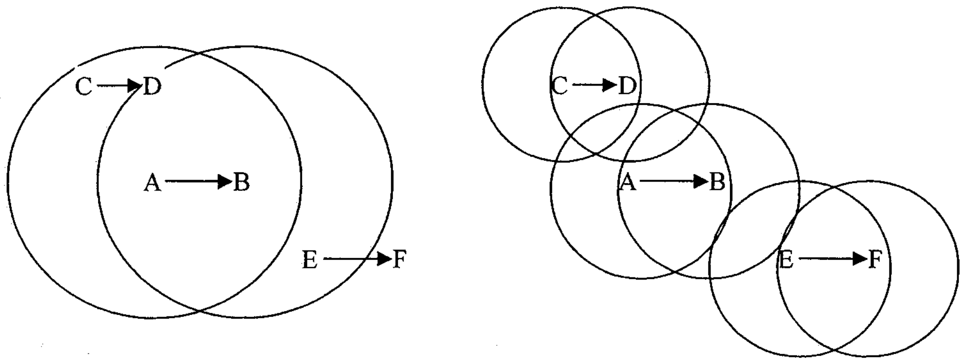
Figure 2. 20 The simulation and analytical results for throughput versus offered traffic (with linear horizontal scale).

Chapter 3

Analysis of Power Controlled T-BTMA Protocol

3.1 Introduction

The main objective of this chapter is to investigate the performance of power-controlled T-BTMA in ad hoc networks. As well known, power is a precious resource in the wireless environment. Specifically, in ad hoc networks, this energy problem is further aggravated due to the limitation associated with small, portable devices. Therefore, it is important to bring the concept of power control into the media access problem in ad hoc networks. In addition, power control is an effective way to maximize the reuse of the channel by reducing the transmission power to minimize the interference with other neighbors. As the power gets low, only a few neighbors are within the transmission range and the interference zones correspondingly shrink. Such short-range transmission is favorable in terms of successful transmission due to the reduced possibility of collision [10]. For example, in Figure 3.1, we illustrate the environments without and with power control, respectively. As may be seen in Figure 3.1-a, in no power controlled networks, the transmission from C to D cannot be successful due to the interference from the ongoing transmission from A to B. Also, the transmission from E to F is suppressed since E senses B's BTr signal on the channel. As may be seen in Figure 3.1-b, all the transmissions can exist concurrently without any interference if we properly apply the power control. Therefore, the approach of power management helps in saving the system power as well as leads to the improvement in the throughput.



(a) Non-power-controlled network

(b) Power controlled network (the sizes of circles may not be same)

Figure 3. 1 Environments without and with power control

In the previous chapter, the mathematical model of the non-power-controlled T-BTMA in ad hoc networks was discussed. Here, we extend the analysis to the case of power-controlled T-BTMA, which combines the busy-tones and RTS with the concept of power control. The numerical results are presented afterwards. Finally, we draw conclusions.

3.2 Power-Controlled T-BTMA

The concept of controlling the transmission power in DBTMA(98) was first introduced in [12]. In this section, we show how this power controlled protocol works. Then, we propose the power- controlled T-BTMA.

The following outlines four essential principles of the power controlled DBTMA(98) scheme:

(1) A source node uses the RTS/CTS dialogue to determine the distance between itself and the destination, then adjusts the transmission power.

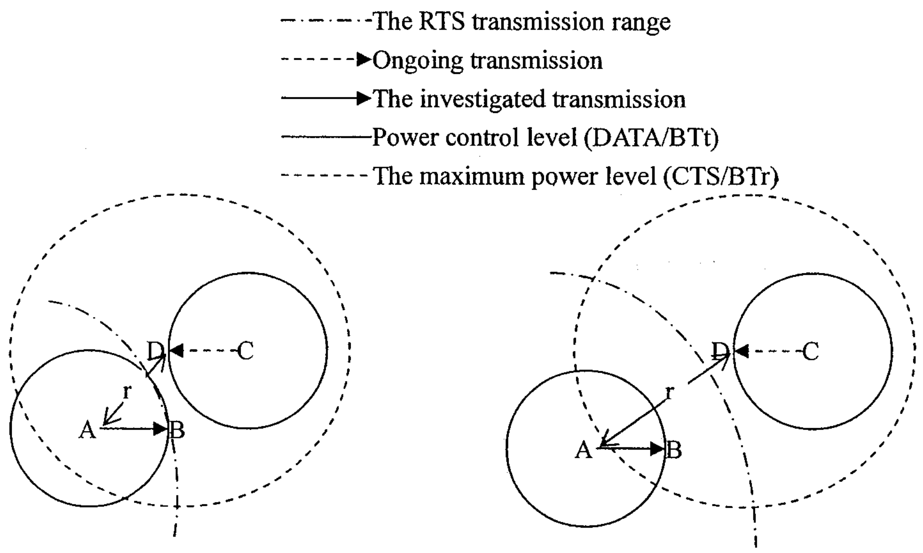
(2) Data packets and BTt signals are sent at the determined power level.

(3) CTS packets and BTr signals are sent at the highest power level.

(4) RTS packets are sent at the power levels, which are determined by the strength of neighboring BTr signals.

Figure 3.2 illustrates two examples of the operations of the power controlled DBTMA(98). There are four nodes A, B, C and D in an ad hoc network. Let R denote the radius of maximum transmission range of each node. The transmission from C to D is ongoing, so during this time the BTt signal at the power level and BTr singal at the maximum power are both on the air. In Figure 3.2-a, A is located at any place at a distance of r from D, r< R. When A wants to communicate with B, it senses the BTr signals and then estimates how far away it is from the ongoing reception at D. After estimating the distance $\overline{AD}$, A tries to send the RTS request to

B at a certain power level which will not reach to D. As B receives the RTS packet, it senses that there is no BTt signal; consequently, B acknowledges the RTS with a CTS packet at the maximum power. Receiving the CTS packet, A estimates the distance of B through the principle (1) above. Then, A transmits its data packet at the power level that can only reach B. On the other hand, if A is outside of D's maximum transmission area, r> R, A cannot detect D's BTr signal, as shown in Figure 3.2-b. Then, A sends the RTS packet at the maximum power level.

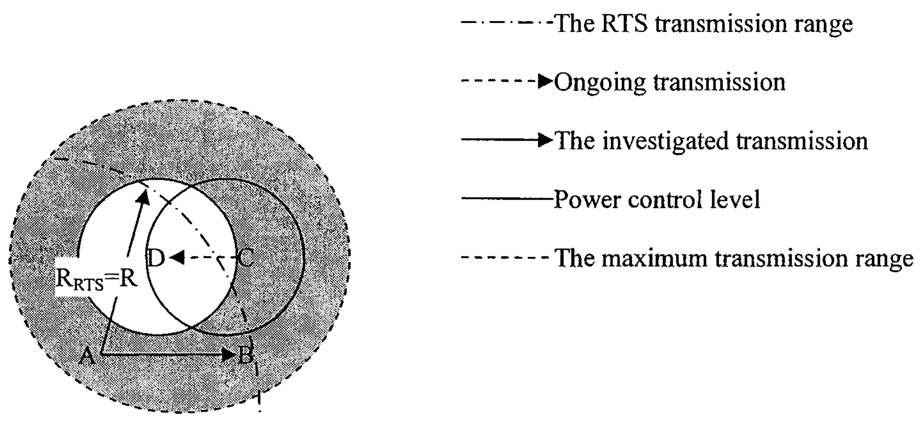

The RTS transmission range
Ongoing transmission
The investigated transmission
Power control level (DATA/BTt)
The maximum power level (CTS/BTr)
D
C
r
A
B
D
C
r
A
B

(a) Node A inside of D's maximum transmission range

(b) Node A outside of D's maximum transmission range

Figure 3. 2 Two examples of the operation of the power controlled DBTMA(98)

The above scenarios differ from the ideal scenario we suggested in Figure 3.1-b, where the communication pair use the same transmission power. A question arises: why the BTr signal can not be sent at the optimal power level? To answer this question, we first investigate the scenario that the BTr signal is transmitted with power control, see Figure 3.3. Suppose that A is located inside of D's maximum transmission range (the dashed circle) but outside of its power controlled transmission range (the solid circle). In this circumstance, A transmits the RTS packet at the maximum power. Although A's RTS transmission does not corrupt the ongoing data transmission, the consequent data transmission from A will corrupt this ongoing data transmission if $\overline{AB} \geq \overline{AD}$. In other words, if we don't keep BTr at the MAXIMUM power, the hidden terminal problem will be more serious due to the absence of the BTr signal in the shaded area where the increased number of hidden-terminals are located. This is called as the "Hidden Terminal Jamming" problem. In conclusion, the BTr signal must be sent at the maximum power level to alleviate the hidden terminal problem in power controlled networks.



$R_{RTS}=R$
D
C
A
B
The RTS transmission range
Ongoing transmission
The investigated transmission
Power control level
The maximum transmission range

Figure 3. 3 The scenario of the BTr signal being sent at a power control level

However, there is a problem in this power controlled scheme. If there are more than one BTr signal on the air under a heavy load, and then, selecting one busy tone to determine the power of an attempted RTS transmission becomes a main difficulty. We suggest a way to alleviate the problem for power-controlled T-BTMA applying above power control scheme. We adopt the Global Positioning System (GPS) [16] to locate neighbors for outdoor purposes. There are two reasons for this choice. One is that there is no RTS/CTS handshake that can be used to determine the proper power level in T-BTMA. The other is that GPS devices are very economical nowadays. For indoor applications, we can rely on the upper layer protocol or alternative technologies to obtain location information. How to locate other nodes in ad hoc networks is beyond the scope of this thesis. Therefore, we assume that a source knows the location of its destination.

The power controlled T-BTMA works as follow: when a node attempts to send its data packet to its destination, it senses the busy-tone channels first. There are three possibilities in initiating an RTS transmission:

1) if there is any BTt signal present, the source suppresses its RTS transmission.

2) if there are only BTr signals present, the source estimates the proportion of its power level to the total power of BTr signals. If the proportion is bigger than a threshold, the source can send its RTS packet at this power level to reach the destination and turns on its BTt signal at the power level; otherwise, the source depresses its RTS transmission.

3) if there are no BTr or BTt signal present; then, the source sends its RTS packet and simultaneously turns on its BTt signal both at power levels.

At the end of RTS transmission, the source turns off the BTt. If there are collisions during the vulnerable period, this channel request fails. If the RTS is successfully transmitted, then, a successful request of the channel depends on whether there are multi-cell interferences from ongoing transmissions. If ongoing transmissions can destroy the new data transmission, the destination will not turn on its BTr to reply the RTS; thus channel initiation fails. If ongoing transmissions can not destroy the new data transmission, the destination will successfully reply by setting its BTr at the MAXIMUM power to indicate a reception. (We will detail how to determine whether an ongoing transmission is harmless in the next section). After detecting a BTr signal, the source turns on its BTd at the power level and simultaneously sends the data packet at the power level. When the data transmission and reception is finished, both BTr and BTd are turned off.

3.3 Performance Analysis of the Power Controlled T-BTMA

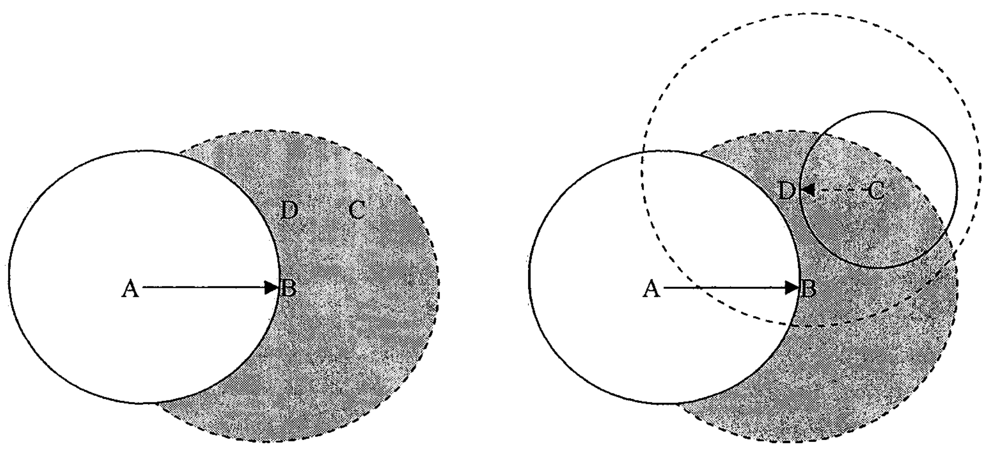
3.3.1 System Model

The basic assumptions and the parameters for the model we study here are same as in Section 2.4, except for the assumption that all nodes are aware of geographical location of their neighbors. Moreover, we assume that the power level is the minimum transmission power that can reach to the destination.

3.3.2 Performance Analysis

In this section, we determine the channel throughput of power controlled T-BTMA in ad hoc networks. The analysis consists of two parts: the first part will derive the probability of successful transmission; and the second part will determine the probability of harmless multi-cell overlapping interference.

To begin, we state the conditions for successful transmission of a packet in home cell of node i and then determine its probability, denoted by $P_i^S(\ell)$, ℓ indicates the transmission distance. As we emphasized in the analysis of non-power-controlled networks, the success of channel initialization depends on whether there are ongoing transmissions in the transmission range of the receiver. The analysis of power controlled networks is complicated. As shown in Figure 3.4, there are two successful channel initialization scenarios. We study a transmission with source, destination nodes as A, B receptively. In Figure 3.4-a, the channel initialization of A is successful since nodes C and D are idle in the shaded area. Figure 3.4-b illustrates successful channel initialization of A with the harmless ongoing transmission from C to D. Here, the harmless ongoing transmission is considered as a transmission originated by a hidden terminal C and $\overline{CD} < \overline{BC}$, which indicates the ongoing transmission from C to D does not interfere on the intended transmission from A to B. In other words, B does not detect the BTd signal of C.



(a) E1: the nodes C and D in shaded area are idle

(b) E2: the nodes C and D in shaded area are active, and $\overline{CD} < \overline{BC}$, therefore such a transmission is harmless

- - - - - ▶Ongoing transmission

———▶The intended transmission

———Power control level(DATA/RTS/BTd)

- - - - - - -The maximum transmission range (BTr)

Figure 3. 4 Two successful RTS transmission scenarios

In brief, besides a successful RTS transmission, either of the following two conditions must be satisfied to initialize the channel successfully in power-controlled networks.

E1: nodes in the shaded area, denoted as B2, are idle.

E2: there are ongoing transmission but they are harmless.

Since events E1 and E2 are mutually exclusive, the probability of successful channel initialization of A is given by,

$$P_A^S(\ell) = \Pr(\text{successful RTS transmission})[\Pr\{E1\}+\Pr\{E2\}]$$

$$= e^{-\Omega_A(td+\tau)} \cdot [P_{B2}^I(\ell) + P_{B2}^B(\ell) \cdot \text{Prob(harmless ongoing transmissions)}] \quad (24),$$

where $P_{B2}^{I}(\ell)$=Prob(nodes in B2 are idle), and $P_{B2}^{B}(\ell)$=Prob(nodes in B2 are busy).

Next, we will determine each of the quantities in Eq. (24). First, we determine the $P_{B2}^{I}(\ell)$ by applying Eq. (15), we have

$$P_{B2}^{I}(\ell)=\frac{\bar{I}_{B2}}{\bar{B}_{B2}+\bar{I}_{B2}}, \qquad (25),$$

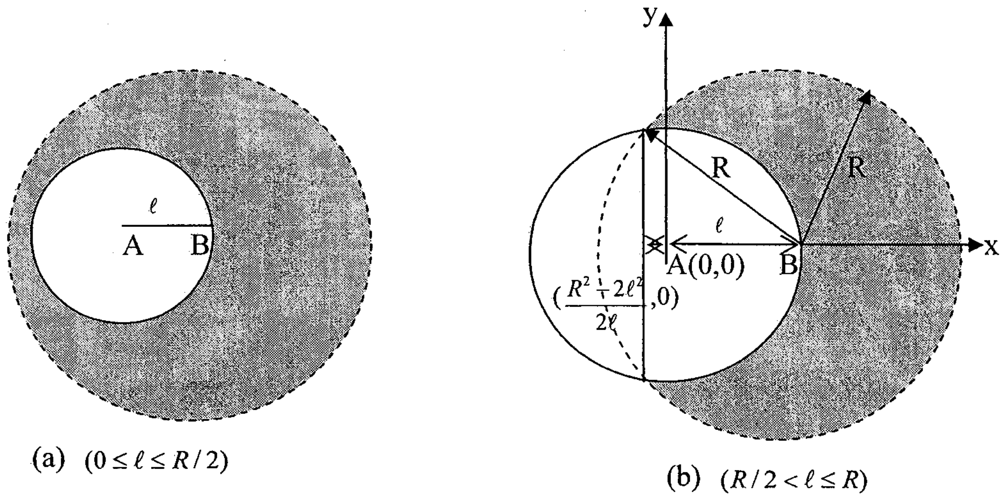
where

$$\bar{I}_{B2}=\frac{1}{\Omega_{B2}}=\frac{1}{\Lambda_{\eta}S_{B2}} \qquad (26).$$

Again we approximate $\bar{B}_{B2}$ as

$$\bar{B}_{B2}=P_{B2}^{S}(\ell)T_{s}+[1-P_{B2}^{S}(\ell)]T_{f} \qquad (27),$$

where, $P_{B2}^{S}(\ell)=e^{-\Omega_{B2}(td+\tau)}$. We make this approximation here because it has alrealy been justified through the simulation of non-power-controlled networks in the previous chapter. Note that the coverage of the shaded area B2 is not fixed as in the previous study for non-power-controlled networks, see Figure 3.5. Since the transmission distance depends on the random location of source and destination, we identify two possibilities.



ℓ
A
B
(a) $(0 \le \ell \le R/2)$
y
x
R
R
ℓ
A(0,0)
B
$(\frac{R^2-2\ell^2}{2\ell},0)$
(b) $(R/2 < \ell \le R)$
——Power controlled transmission range
-----Maximum transmission range

Figure 3. 5 Two possible situations for the shaded area

Situation I: $\ell \le R/2$. Therefore, the coverage of the shaded area B2 is given by

$$S_{B2} = \pi R^2 - \pi \ell^2 \qquad (0 \le \ell \le \tfrac{1}{2}R)$$

Situation II: $R/2 < \ell \le R$. Assuming that a coordinate system is centered at node A. Let x and y denote the magnitudes of the coordinates of the intersection of the two circles. Then, we have

$$\begin{cases} x^2 + y^2 = \ell^2 \\ (x+\ell)^2 + y^2 = R^2 \end{cases} \Rightarrow x = \frac{R^2 - 2\ell^2}{2\ell}$$

Therefore, the coverage of the shaded area B2 is given by

$$S_{B2} = Lens(R, -\frac{R^2}{2\ell}) - Lens(\ell, \frac{2\ell^2 - R^2}{2\ell}) \qquad (R/2 < \ell \le R),$$

see Eq. A3 in Appendix A.

Now, we have,

$$S_{B2} = \begin{cases} \pi R^2 - \pi \ell^2 & (0 \le \ell \le \frac{1}{2}R) \\ Lens(R, -\frac{R^2}{2\ell}) - Lens(\ell, \frac{2\ell^2 - R^2}{2\ell}) & (R/2 < \ell \le R) \end{cases} \quad (28)$$

Substituting Eqs. (26)-(28) into (25), we have

$$P_{B2}^{I}(\ell) = \frac{1}{\Lambda_\eta S_{B2} \cdot [e^{-\Lambda_\eta S_{B2} \cdot (td+\tau)}(T_s - T_f) + T_f] + 1} \quad (29).$$

Next, we will determine $P_{B2}^{B}(\ell)$. Since $P_{B2}^{B}(\ell)$ is the probability of the event that nodes in B2 are busy,

$$P_{B2}^{B}(\ell) = \sigma_{B2} = 1 - P_{B2}^{I}(\ell) \quad (30).$$

Combing Eqs. (24)-(30), we have following set of formulas,

$$\begin{cases} P_A^S(\ell) = e^{-\Lambda_\eta \pi \ell^2 (td+\tau)} \cdot \{P_{B2}^{I}(\ell) + [1 - P_{B2}^{I}(\ell)] \cdot \text{Prob(harmless ongoing transmission)}\} \\ P_{B2}^{I}(\ell) = \frac{1}{\Lambda_\eta S_{B2} \cdot [e^{-\Lambda_\eta S_{B2} \cdot (td+\tau)}(T_s - T_f) + T_f] + 1} \\ S_{B2} = \begin{cases} \pi R^2 - \pi \ell^2 & (0 \le \ell \le \frac{1}{2}R) \\ Lens(R, -\frac{R^2}{2\ell}) - Lens(\ell, \frac{2\ell^2 - R^2}{2\ell}) & (R/2 < \ell \le R) \end{cases} \end{cases} \quad (31)$$

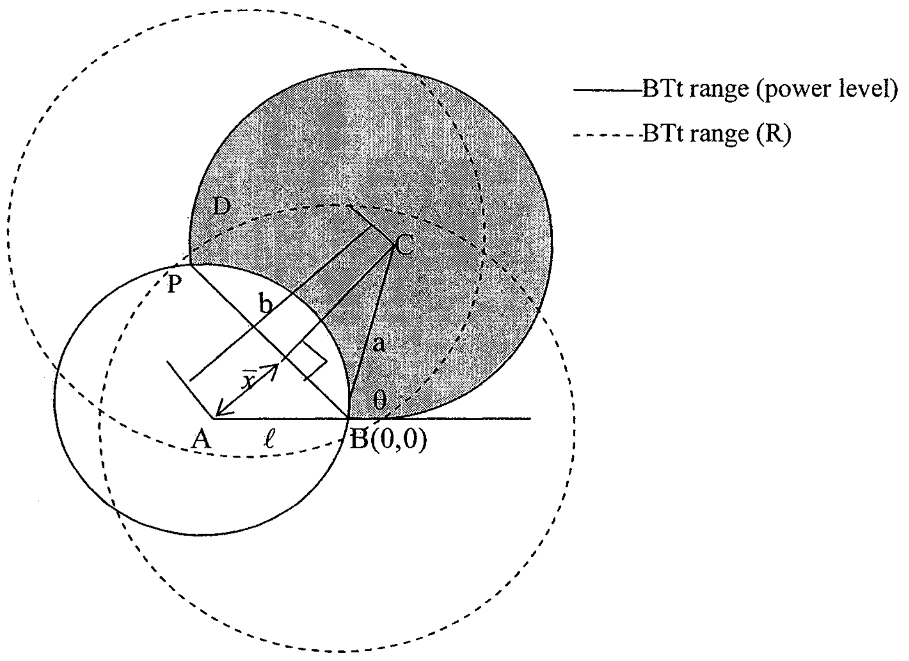


Figure 3. 6 Node D locates in the shaded area

Finally, we will find the Prob(harmless ongoing transmission) by using a geometrical approach. We let $P_h(\ell)$ denote the probability of this event. As shown in Figure 3.6, there is an ongoing transmission from C to D, while we intend to have a transmission from A to B. In this figure, ℓ denotes the transmission distance form A to B. Let "b" denote the distance between A and C, "a" denote the distance between C to B, "θ" denote the angle between $\overrightarrow{BC}$ and $\overrightarrow{AB}$, and "$\bar{x}$" denote the distance between A to $\overline{PB}$ (point "P" indicates another intersection of circles A and C). As may be seen, D can be located anywhere inside the maximum transmission range of C, except in A's transmission range. If $\overline{CD}$ is shorter than $\overline{BC}$, the ongoing transmission will be harmless,and the transmission from A to B can take place. In other words, D must be positioned in the shaded area, refer to Figure 3.6. The probability is

$$P_h(\ell)=\text{Prob}[(\overline{CD}<\overline{BC})\cup(D\notin\overline{A})]$$

=Prob(D is located in the shaded area in Figure 3.6)

$$=Area_{shaded} \,/\, Area_{\max\_transmission\_range}$$

$$=\frac{Lens(a,\overline{x}-b)-Lens(\ell,\overline{x})}{\pi R^2} \qquad (32),$$

where

$$b=\sqrt{\ell^2+a^2+2\ell a\cos\theta} \qquad (33).$$

And $\cos\angle CAB=\frac{\overline{x}}{\ell}=\frac{\ell+a\cdot\cos\theta}{b}$, so

$$\overline{x}=\frac{\ell^2+\ell a\cdot\cos\theta}{b} \qquad (34).$$

Substitute Eqs. (33) and (34) into Eq(32), we obtain the double conditional probability: given location of node C, C(a, θ); given transmission distance ℓ. C is uniformly distributed with respect to B; thus, we have $f(a)=\frac{2a}{R^2}$, and $f(\theta)=\frac{1}{2\pi}$. First, we remove the condition on C(a, θ) to obtain the probability of harmless ongoing transmission for a given transmission distance,

$$P_h(\ell)=\int_0^R\frac{2a}{R^2}\cdot da\int_0^{2\pi}\frac{1}{\pi}\frac{Lens(a,\overline{x}-b)-Lens(\ell,\overline{x})}{\pi R^2}\cdot d\theta \qquad (35)$$

Now we have determined all the quantities in Eq.(24). Next, we apply the renewal theory from Eq. (6) to obtain the channel throughput in power controlled networks for a given transmission distance as,

$$\rho_A=\frac{\overline{T}_A}{\overline{B}_A+\overline{I}_A}=\frac{\delta P_A^S(\ell)}{P_A^s(\ell)(T_s-T_f)+T_f+\frac{1}{\Lambda_\eta\pi\ell^2}} \qquad (36).$$

Combining Eqs.(31) and Eq.(36)

$$
\begin{cases}
\text{(i): } \rho_A = \dfrac{\overline{T}_A}{\overline{B}_A + \overline{I}_A} = \dfrac{\delta P_A^S(\ell)}{P_A^S(\ell)(T_s - T_f) + T_f + \dfrac{1}{\Lambda_\eta \pi \ell^2}} \\
\text{(ii): } P_A^S(\ell) = e^{-\Lambda_\eta \pi \ell^2 (td+\tau)} \cdot \{P_{B2}^I(\ell) + [1 - P_{B2}^I(\ell)] \cdot p_h(\ell)\} \\
\text{(iii): } P_{B2}^I(\ell) = \dfrac{1}{\Lambda_\eta S_{B2}[e^{-\Lambda_\eta S_{B2}}(T_s - T_f) + T_f] + 1} \\
\text{(iv): } P_h(\ell) = 2 \cdot \int_0^R \dfrac{2a}{R^2} \cdot da \int_0^\pi \dfrac{1}{\pi} \dfrac{Lens(a, \overline{x} - b) - Lens(\ell, \overline{x})}{\pi R^2} \cdot d\theta \\
\text{(v): } \\
\quad S_{B2} = \begin{cases} \pi R^2 - \pi \ell^2 & (0 \le \ell \le \frac{1}{2} R) \\ Lens(R, -\dfrac{R^2}{2\ell}) - Lens(\ell, \dfrac{2\ell^2 - R^2}{2\ell}) & (R/2 < \ell \le R) \end{cases}
\end{cases}
$$

All these equations are functions of transmission distance, ℓ. Finally, unconditioning with ℓ using Eq.(3), we obtain,

$$
\begin{aligned}
\overline{\rho}_A &= \int_0^R \frac{2\ell}{R^2} \rho_A \cdot d\ell \\
&= \int_0^{R/2} \frac{2\ell}{R^2} \rho_A \cdot d\ell + \int_{R/2}^R \frac{2\ell}{R^2} \rho_A \cdot d\ell
\end{aligned} \tag{37}
$$

3.4 Numerical Results

Next, we present numerical results for the power-controlled T-BTMA. Figure 3.7 shows average throughput as a function of mean packet generation rate with and with out power control. As may be seen, power controlled protocol outperforms the non-power controlled T-BTMA. Especially under heavy load, the former reaches the maximum average channel throughput at 0.98 when Λ_η is at10, while the average channel throughput of the latter declines steeply to 0.1.

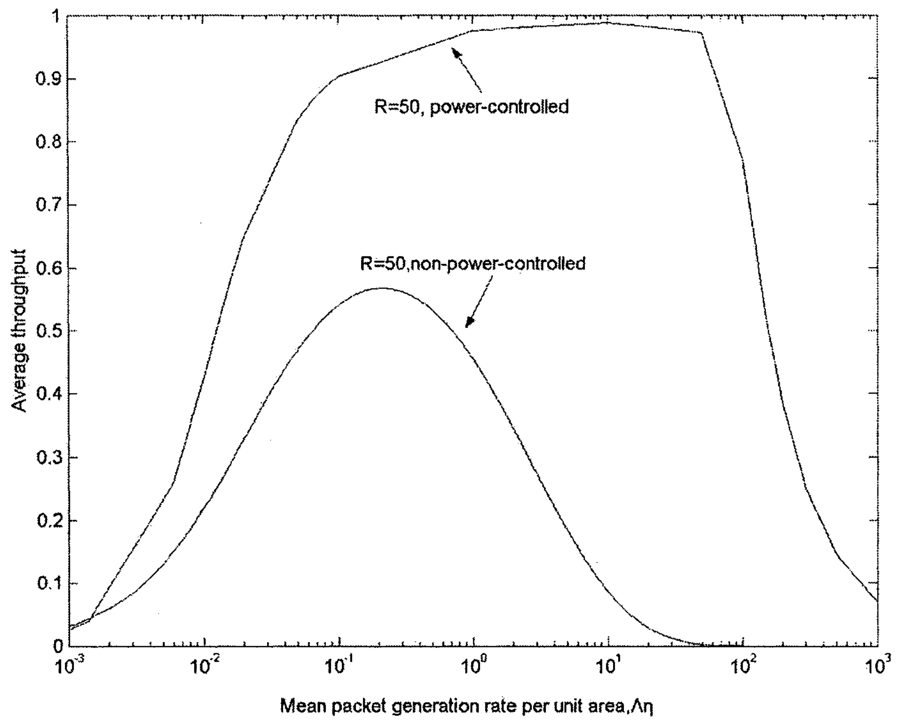


Figure 3. 7 The network throughput; power control versus non-power-control

Chapter 4

Conclusions and Future Work

The main objective of MAC protocols is to efficiently control access of a shared medium among multiple nodes. In mobile ad hoc networks, this is of more challenge since packet collisions, media contention and multi-cell overlapping interference, which are introduced by broadcasting nature and shared transmission medium, are difficult to eliminate. And these problems seriously degrade the performance of MAC protocols.

In this thesis, we propose to modify DBTMA protocol through inserting a busy tone signal for the protection of data transmissions. We refer to the modified version as T-BTMA protocol. The existing performance model of DBTMA protocol fails to take into account multi-cell overlapping interference. The main contribution of this thesis is development of a performance analysis of T-BTMA protocol that approximately takes into consideration multiple-cell overlapping interference. We determine throughput of the system as a function of packet arrival rate using renewal theory results. Then, we show that our model may be adopted for most RTS/CTS class of MAC protocols.

In order to verify our mathematical model, a simulation environment is developed. Our simulation is of particular interest to overlapping-cell scenario, instead of group performance. The results of this more realistic simulation validate our mathematical model and demonstrate that multi-cell overlapping interference degrade the channel throughput. Moreover, we observe that if we substitute proper system parameters, for example, t_d, Ts and T_f, into our mathematical model, the maximum throughput of numerical result matches the maximum throughput of the existent simulation result of CSMA/CA and MACA protocols in [6] and [2], respectively. Therefore, we conclude that our model can be used in most RTS/CTS class MAC protocols.

Fixed transmission power wastes limited battery energy in ad hoc networks. Therefore, in the later part of this thesis, we propose the power-controlled T-BTMA for ad hoc networks. Based on the knowledge of the location of a destination, a source transmits its packet at a power level to reach the destination. We emphasize that the BTr signal should be sent at the maximum power to alleviate the hidden-terminal and the hidden-terminal jamming problems. Finally, analyses show that channel throughput can be significantly increased by our protocol.

Next, we discuss possible future work. This research may be extended in number of directions. First, we would like to develop better MAC protocols to solve the hidden and exposed terminal problems. A possible second research direction concerns the power controlled MAC protocols. It has been shown that throughput performance of power-controlled T-BTMA is better than that of without power control. However, this comparison is based in signal hop communication. As the transmission range is reduced, it takes high number of hops to reach the final destination. Thus, we need to have more balanced comparison of the two approaches. Finally, development of an exact performance analysis of this class of MAC protocols remains as the ultimate goal.

References

[1] P. Karn, “MACA-A New Channel Access Method for Packet Radio,” in ARRL/CRRL Amateur Radio 9th Computer Networking Conference, pp. 134-40, ARRL,1990.

[2] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, “MACAW: A Media Access Protocol for Wireless LAN’s,” in Proc. Acm SIGCOMM ’94,1994, pp.212-225.

[3] J. Deng and Z.J. Hass, “Dual Busy Tone Multiple Access (DBTMA) --- A Multiple Access Control Scheme for Ad Hoc Networks,” IEEE Trans. Commun, vol.50, No.6, June. 2002.

[4] Z.J. Hass and J. Deng, “Dual Busy Tone Multiple Access (DBTMA) – Performance Evaluation,” in proc. 49th Annu. Int. Veh. Technol. Conf., Oct.1998.

[5] J. Deng and Z.J. Hass, “Dual Busy Tone Multiple Access (DBTMA): A New Medium Access Control for Packet Radio Networks,” in Proc. Int. Conf. Universal Personal Commun., Oct. 1998.

[6] F. A. Tobagi and L. Kleinrock, “Packet Switching in Radio Channels: Part II—The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy-Tone Solution,” IEEE Trans. Commun., vol.23, pp1417-1433,1975.

[7] H. Inaltekin and Z. J. Hass, “The Analysis of Power-controlled MAC Layer for Wireless Ad Hoc Networks,” 2003.

[8] C. L. Fuller and J.J.G, “Solutions to Hidden Terminal Problems in Wireless Networks,” in Pro. ACM SIGCOMM’97, 1997, pp.39-49.

[9] K.C. Huang and K. C. Chen, “Interference Analysis of Nonpersistent CSMA with Hiddern Terminals in Multicell Wireless Data Networks,” 1995.

[10] H. Takagi and L. Kleinrock, “Optimal Transmission Range for Randomly Distributed Packet Radio Terminals,” IEEE Trans. Commun, vol, com-32, No. 3. march, 1984.

[11] T. Hou and V. O. K. Li, “Transmission Range Control in Multihop Packet Radio Networks,” IEEE Trans. Commun, vol, com-34. No.1. Jan, 1986.

[12] S. Wu, Y. Tsend and J. Sheu, “Intelligent Medium Access for Mobile Ad Hoc Networks with Busy Tones and Power Control,” IEEE Trans. Commun, vol, 18, No.9, Sep, 2000.

[13] IEEE Std 802.11-1999: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications.

[14] Mauro Borgo, Andrea Zanella, Paola Bisaglia and Simone Merlin, “Analysis of the hidden terminal effect in multi-rate IEEE 802.11b networks,” WPMC04, sept.2004.

[15] Asis Nasipuri, Jun Zhuang, and Samir R. Das, “A Multichannel CSMA MAC Protocol for Multihop Wireless Networks,” IEEE, 1999.

[16] Stefano Basagni, Mobile Ad Hoc Networking, Wiley-Interscience,2004

[17] Y. B. Ko, V. Shankarkumar, and N. H. Vaidya, “Medium access control protocols using directional antennas in ad hoc networks”, Proc. of IEEE INFOCOM’2000,

Mar.2000.

[18] A. Nasipuri, S. Ye, et al. “A MAC protocol for Mobile Ad Hoc Networks Using Directional Antennas“, IEEE Wireless Communications and Networking Conferece, Sep 2000.

[19] D. Lal, R. Gupta and D. P. Agrawal, “Throughput Enhancement in Wireless Ad Hoc Networks with Spatial Channels – A MAC layer Perspective”, IEEE ISCC’02.

[20] H. Takagi and L. Kleinrock, “Optimal transmission ranges for randomly distributed Packet radio terminals,” IEEE Trans: Commun,. Vol. COM-32, No. 3, Mar.1984

[21] T. Hou and V. O. Li, “Transmission Range Control in Multihop Packet Radio Networks,” IEEE Trans: Commun., VOL. COM-34. No.1 January 1986.

[22] S. Wu, Y. Tsend and J. Sheu, “A Multi-channel MAC Protocol with Power Control for Multi-hop Mobile Ad Hoc Networks,” British Computer Society, the Computer Journal, Vol. 45, No.1, 2002.

[23] J. P. Monks, V. Bharghavan and W. W. Hwu, “ A Power Controlled Multiple Access Protocol for Wireless Packet Networks,” IEEE INFOCOM 2001.

[24] T. Elbatt and A. Ephremides, “Joint Scheduling and Power Control for Wireless Ad Hoc Networks,” IEEE Trans: Wireless Commun, Vol. 3, No. 1, January 2004.

[25] C. Ware, T. Wysocki, and J. Chicharo, “On the hiddern terminal jamming problem in IEEE 802.11 mobile ad hoc networks,” in IEEE ICC,2001

[26] S. Gobriel, R. Melhem and D. Mosse, “A Unified Interference/Collision Analysis

for Power-Aware Adhoc Networks," IEEE, INFOCOM, March, 2004

[27] Leonard Kleinrock, "Packet Switching in Radio Channels: Part I—Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characterisitcs," IEEE TRAN. Comm, Vol. Com-23, No.12, Dec,1975

[28] Richard S. Sutton and Andrew G. Barto, "Reinforcement Learning," Chapter.11, MIT Press, Cambridge, MA, 1998

[29] Leonard J. Cimini, Jr., Gerard J. Foschini and Larry A. Shepp, "Single-Channel User-Capacity Calculations for Self-Organizing Celluar Systems," IEEE TRAN. Comm, Vol. 42, No.12, Dec,1994

Appendix

A. Circle-Circle Intersection

To find the area of the asymmetric "lens" in which the circles intersect (see Figure A.1), we use the following formula for the circular segment of radius R and the scale d whose absolute value is the triangular height,

$$Lens(R,d) = R^2 \cos^{-1}(\frac{d}{R}) - d\sqrt{R^2 - d^2} \qquad \text{(A1).}$$

We note that the first term in the above equation gives the area of a pie with radius R and angle θ , while the second term gives the area of the isosceles triangle that has the height $|d|$. Clearly, the difference between the two terms corresponds to area of the lens.

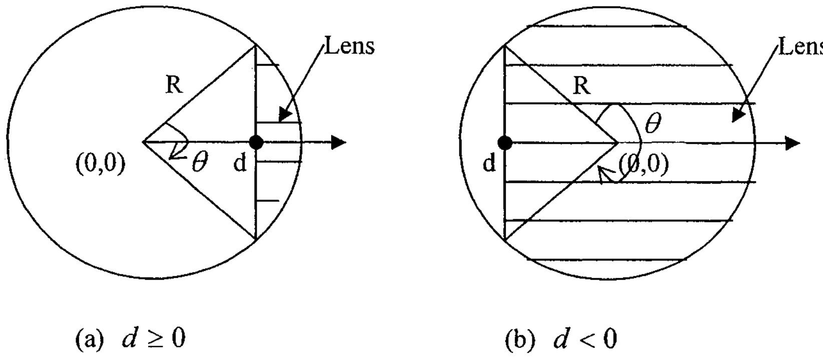


Figure A. 1 The area of lens

As illustrated in Figure A.2, two circles have identical radius R. The coverage area of the shadow is

$$S_{shaded} = Lens(R, -\frac{\ell}{2}) - Lens(R, \frac{\ell}{2}) \quad \text{(A2).}$$

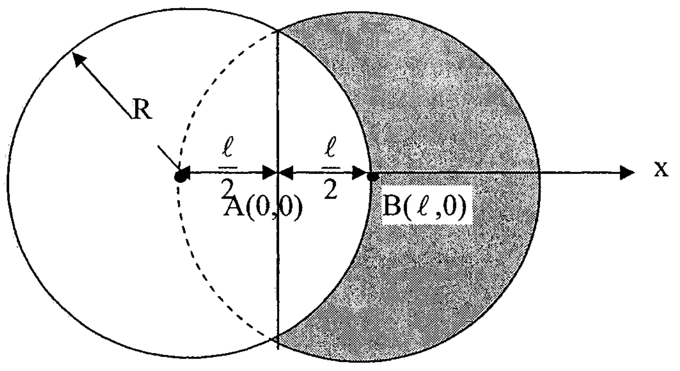


Figure A. 2 The coverage area of shadow with same radius R

As illustrated in Figure A.3, the two circles that have different radius ℓ and R. The coverage area of the shadow is given by

$$S_{shaded} = Lens(R, x - \ell) - Lens(\ell, x) \quad \text{(A3).}$$

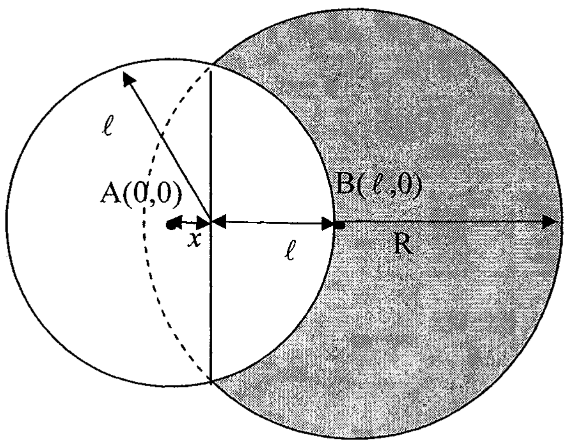


Figure A. 3 The coverage area of shadow with different radius

B. Simulation Source Code

```
//main function of the simulation of TBTMA
#include<iostream>
using std::cout;
using std::endl;
using std::cin;
using std::ios;

#include<iomanip>
using std::setw;
using std::setiosflags;

#include<cmath>

#include<fstream>
using std::ofstream;
ofstream prresults("outputc.dat",ios::out);

#include "Genode.h"
#include "Neighbor.h"
#include "Tbtma3_3.h"

double Average(double [10]);          //averages the value of array
double est_avr(double [10],double);   //estimates the confidence interval
void prout(double [40]);              //prints array
void controltime();                   //controls the interval of each simulation run

int main()
{
    int NetR,j=0;                     //defining NetR as network range.
    const int n=100,m=2;
    double node[100][2]={0};          //defining a "node" array to contain location information.
    int neighborID[100][100]={0};     //contains the neighbor ID.
    int neighbool[100][100]={0};      //contains boolean value for neighbors.
    double arrival_rate;              //defining arrival rate.
    double G=0, D=0,A=0;          //G-Total arrival rate; D-Average delay; A-average throughput
    double arrG[10]={0},arrD[10]={10},arrthr[10]={0};
    double avedel_var,AverG[40]={0},AverD[40]={0},throu[40]={0}; //contains average values.
    int runtime=10,ind;                                    //simulation times and index.
    double condelayl[40]={0},condelayr[40]={0};           //confidence interval
    double tan_1=2.262;                                  //95%, value of t distribution.
    int inj,ini;

simubegin:
    cout<<"\nEnter Network Range:   ";
    cin>>NetR;                         //user input transmission range on screen.

    //Simulation begins
    Genode generate(node,NetR); //generating random nodes.
    //generate.printnode(node);     //printing nodes.

    for(ini=0;ini<100;ini++)          //initializing neighborID and neighbool
    {
        for(inj=0;inj<100;inj++)
        {
            neighborID[ini][inj]=0;
            neighbool[ini][inj]=0;
        }
    }

    Neighbor findnei(node,neighborID,neighbool);   //Finding neighbors

    //printing neighbor ID, first column is the total number of neighbors.
    findnei.printneiID(neighborID);

    //check each node has at least one neighbor
    for(ind=0;ind<100;ind++)
        if (neighborID[ind][0]==0)
            goto simubegin;

    cout<<"\nEnter Begin of Arrival Rate:   ";
    cin>>arrival_rate;                          //user input arrival rate on screen.

    for(ini=0;ini<40;ini++)                     //40 values arrival_rates, from input arrival rate
    {
        for(ind=0;ind<runtime;ind++)
        {
            G=D=A=0;

            //Process the protocol.
            Tbtma3_3 process(neighborID,neighbool,arrival_rate,&G,&D,&A);
            arrG[ind]=G;
            arrD[ind]=D;
```

```
                arrthr[ind]=A;

                cout<<"...";
                controltime();          // controls the interval of each simulation run

                if (arrG[ind]==0)       //system saturated
                    goto abortsimul;
            }
            AverG[ini]=Average(arrG);       //recording average throughput
            AverD[ini]=Average(arrD);       //recording average delay
            throu[ini]=0.004096*Average(arrthr)*2500/pow(NetR,2);     //converting to signal cell
            avedel_var=est_avr(arrD,AverD[ini]);                       //variance of delay

            //left value of confidence interval of delay
            condelayl[ini]=AverD[ini]-tan_1*avedel_var/pow(runtime,0.5);
            //right value of confidence interval of delay
            condelayr[ini]=AverD[ini]+tan_1*avedel_var/pow(runtime,0.5);

            cout<<arrival_rate<<endl;
            arrival_rate+=100;              //adding 100 on arrival rate of each simulation run
        }

abortsimul:
        prresults<<"Aggregate Arrival Rate:"<<endl;
        prout(AverG);
        prresults<<"Throughput:"<<endl;
        prout(throu);
        prresults<<"Average Delay:"<<endl;
        prout(AverD);
        prresults<<"Left bound of Confidence Interval:"<<endl;
        prout(condelayl);
        prresults<<"Right bound of Confidence Interval:"<<endl;
        prout(condelayr);

        return 0;
}

//averages the value of array
double Average(double a[10])
{
        double Sum=0;
        for(int i=0; i<10; i++)
        {
            if (a[i]==0)
        {
            Sum=0;
            goto returnsum;
        }
        Sum+=a[i];
    }
returnsum:
    return Sum/10;
}

//estimates the confidence interval
double est_avr(double c[10],double expect)
{
    double sum1=0;
    for(int i=0;i<10;i++)
        sum1+=pow((c[i]-expect),2);
    return pow((sum1/9),0.5);
}

//controls the interval of each simulation run
void controltime()
{
    for(int i=0;i<200000000;i++);
}

//prints out array
void prout(double b[40])
{
    prresults<<"[";
    for(int i=0;i<40;i++)
    {
        prresults<<b[i]<<" ";
    }
    prresults<<"]"<<endl;
    prresults<<endl;
}


//Declaration of Genode class.
//Member functions are defined in Genode.cpp.
#ifndef GENODE_H
#define GENODE_H

class Genode
```

```
{
public:
    Genode(double [][2],int);

    bool compare(double,double,double); //checking whether node outside the network range.
    void printnode(double [][2]);          //printing the randomly generated node

private:
    int n,m;                              //define array size.

};

#endif


//Declaration of the Neighbor class.
//Member functions are defined in Neighbor.cpp
#ifndef NEIGHBOR_H
#define NEIGHBOR_H
#include "Genode.h"

class Neighbor
{
public:
    Neighbor(double [][2],int [][100],int [][100]);
    bool compare(double,double,double); //compares the distance of two nodes with radio radius.
    void printneiID(int [][100]);          //prints the number of neighbors and their IDs.
    //prints the bool value of neighbors,1--neighbor,0--not eighbor.
    void printneitable(int [][100]);

private:
    double TrRange;                       //define network range.
    int n;                                //nodes number.
};

#endif

//Declaration of Tbtma3_3 class.
//Member functions are defined in Tbtma3_3.cpp.
#ifndef TBTMA3_3_H
#define TBTMA3_3_H

class Tbtma3_3
{
```

```
public:
    //schedules processing.
    Tbtma3_3(int [][100],int [][100], double, double *,double *,double *);
    void prinitime();

private:
    double csmasched[101][8];     //system schedule.
    int nopacket[100];            //contains the nodes which haven't packet.
    double ini_time[2000];        //records arrival time of each packet
    double td,pdelay,RTStime,datatime;    //simulation parameters.
    double Sumdelay,simultime;
    double BTttime,BTrtime,COLLtime;
    double rate;
    int count;                    //counts the number of idle nodes
    int Sumnewarr,Sumoldarr;      //counts the arrivals

    double backoff();             //gets random backoff time
    int get_node(int);            //radomly gets index of node
    double gettime();             //gets random time with exponential distribution
    void bubbleSort();            //ascendes order in csmasched[][2]
    void upmove();                                  //deletes the first row of array.
    int TRcheck(double,int,int[][100]);            //checks any ongoing transmission
    int REcheck(double,int [][100]);               //checks any ongoing receiving
    void schenewsense(int,int,int,int [][100]);    //gets new gerneration
    void prnopacket();                              //prints the No-packet array
    void prsche();                                  //prints system schedule
    int getarraysize();                             //gets size of array
    int countbackoff(int);                          //counts backoff times of each packet
};
#endif


//Member functions for class Genode.
//Used to randomly generate 100 nodes.
#include<iostream>
using std::cout;
using std::endl;
using std::cin;
using std::ios;

#include<iomanip>
using std::setw;
using std::setiosflags;
```

```
#include<cmath>
#include<ctime>
#include<cstdlib>

#include<fstream>
using std::ofstream;

ofstream printgenode("genode.dat",ios::out);

#include "Genode.h"

Genode::Genode(double generatenode[][2],int R)
{
    n=100;                          //nodes number.
    m=2;

    srand(time(0));             //reading system clock to obtain the value of the seed.

    //randomly generate 100 nodes on a disk with radius R.
    for(int i=0;i<n;i++)
    {
        for (int j=0;j<m;j++)
        {
            //randomly place the node
             generatenode[i][j]=(static_cast<double>(rand())/RAND_MAX-0.5)*2*R;
             cout<<setiosflags(ios::fixed)<<setw(15)<<generatenode[i][j];
        }

        //checking the random node if it is on the disk with radius R.
        if (!compare(generatenode[i][0],generatenode[i][1],R))
            i--;                                    //if not, random generating a new node again.
        cout<<"\n";
    }
    cout<<"\n";
}

//checks the random node with the network range, return boolean value.
bool Genode::compare(double x,double y,double distance)
{
    return pow(x,2)+pow(y,2)<pow(distance,2);   //if x*x+y*y<d*d, return bool value 1.
}

//prints the randomly generated nodes.
void Genode::printnode(double node[][2])
```

```
{
    printgenode<<"node ID"
        <<setw(12)<<'x'<<setw(12)<<'y'<<endl;
    for (int i=0;i<100;i++)
    {
        printgenode<<setw(7)<<i;
        for (int j=0;j<2;j++)
            printgenode<<setw(12)<<node[i][j];
        printgenode<<endl;
    }
}


//Member functions for class Neighbor
//Used to find all nodes' neighbors
#include<iostream>
using std::cout;
using std::endl;
using std::cin;
using std::ios;
using std::cerr;

#include<iomanip>
using std::setw;
using std::setiosflags;

#include<cmath>
#include<ctime>
#include<cstdlib>

#include<fstream>
using std::ofstream;

ofstream printneid("neighborID.dat",ios::out);

#include "Neighbor.h"

//Neighbor constructor initializes each data member.
//In neighbor constructor, "orinode" is used to receive a copy of generated "node",
//"IDtable" is used to contain number of neighbors and IDs.
//"neighbortable" is used to contain the boolean value, 1-neighbor,0-not neighbor.
Neighbor::Neighbor(double orinode[][2],int IDtable[][100],int neighbortable[][100] )
{
    TrRange=50;    //radio radius.
```

```
        n=100;              //number of nodes.
        double dx,dy;       //distance between two nodes, used to compare with the RR.

        //finds the neighbors of each nodes, using boolean value.
        for(int i=0;i<n;i++)
        {
            neighbortable[i][i]=1;  //setting all themselves as their neighbors.
            for (int j=i+1;j<n;j++)
            {
                dx=orinode[j][0]-orinode[i][0];         //calculating the x-distance of the next nodes.
                dy=orinode[j][1]-orinode[i][1];         //calculating the y-distance of the next nodes.
                //calling function "compare",setting boolean value to the array symmetrically.
                neighbortable[i][j]=neighbortable[j][i]=compare(dx,dy,TrRange);
            }
        }

        int count;                  //counts number of neighbor of each node.
        for(i=0;i<n;i++)
        {
            count=0;                //each repetition of the row, set counter to be 0.
            for(int j=0;j<n;j++)
            {
                if ((neighbortable[i][j]==1)&&(i!=j))   //obtain neighbor ID
                {
                    IDtable[i][++count]=j; //put ID to array IDtable.
                    IDtable[i][0]=count;    //the fist column contain the number of neighbors.
                }
            }
        }
        //printneiID(IDtable);
    }

    //compares distance of nodes with radio radius, return boolean value.
    bool Neighbor::compare(double x,double y,double distance)
    {
        return pow(x,2)+pow(y,2)<pow(distance,2);          //if x*x+y*y<d*d, return bool value 1.
    }

    //prints neighbor ID.
    void Neighbor::printneiID(int a[][100])
    {

        printneid<<"nodeID"<<setw(6)<<"Total"
            <<setw(12)<<"Neighbor ID"<<endl;
    for (int i=0;i<100;i++)
    {
        int j=1;
        printneid<<setw(6)<<i
            <<setw(5)<<a[i][0]<<" ";
        while(a[i][j]!=0||a[i][j+1]!=0)
            //only print the useful data.not print the sequent "0" at the end of each row.
            printneid<<setw(3)<<a[i][j++];
        printneid<<'\n';
    }
}

//prints the boolean value table of neighbors.
void Neighbor::printneitable(int b[][100])
{
    for (int i=0;i<100;i++)
    {
        for (int j=0;j<100;j++)
            printneid<<setw(3)<<b[i][j];
        printneid<<'\n';
    }
}

//Member function for Tbtma3_3 class.
//used to process the protocol
//backoff uniformly [0,0.5d],double after every collsision and resensing.
//using exponential backoff algorithm
//only sense three busy tones, not the channel
//final edition

#include<iostream>
using std::cout;
using std::endl;
using std::cin;
using std::ios;

#include<iomanip>
using std::setw;
using std::setiosflags;
using std::setprecision;

#include<ctime>
#include<cmath>
```

```
#include<fstream>
using std::ofstream;

ofstream prcsmasche("csmaschedule.dat",ios::out); //print system schedule.
ofstream printpacket("nopacket.dat",ios::out);     //print dynamic changes of the nodes which
                                                    haven't pakcets after each event processing
ofstream printinitime("initime.dat",ios::out);      //print arrival time of each packet.
ofstream proutput("results.dat",ios::out);          //print simulation results.

#include "Tbtma3_3.h"

//The process of TBTMA protocol
//Calling the neighbor ID array, the corresponding boolean array from main.cpp;
//receiving addresses of two argument, G and D, aggregate rate and delay.
Tbtma3_3::Tbtma3_3(int nei_ID [][100],int neibool [][100],double arrivalrate,double *g,double
*d,double *a)
{
    srand(time(0));              //read system clock to obtain the value of the seed.

    //simulation parameters.
    td=0.00001;                  //the busy tone detection delay
    pdelay=0.00000012;           //propagation and mechanical delay
    RTStime=0.0002;              //transmission time of RTS
    datatime=0.004096;           //transmission time of DATA

    //initializing arguments contain simulation statistics
    Sumdelay=0;                  //sumof delay
    count=100;                   //counter, counting the number of node which has nopacket
    Sumnewarr=0;                 //sum of new arrival
    Sumoldarr=0;                 //sum of sensing and resensing.
    rate=arrivalrate;            //calling arrival rate from main.cpp

    BTttime=td+pdelay+RTStime;       //setting the duration of BTt
    BTrtime=td+5*pdelay+datatime;    //setting the duration of BTr
    COLLtime=td+pdelay;              //setting collision window

    proutput<<"arrivalrate = "<<rate<<"."<<endl;         //printing out offered arrival rate

    int i,j;
    int timeout=50;         //setting timeout of collsisions and resensings

    //initializing the array that contains arrival time of each packet.
    for(i=0;i<2000;i++)
        ini_time[i]=0;
```

```
//initializing the array that all these nodes haven't packet at the beginning of each simulation.
for(i=0;i<100;i++)
    nopacket[i]=i;

//initializing the system schedule.
for(i=0;i<101;i++)
{
    for(j=0;j<8;j++)
        csmasched[i][j]=0;
}

int packet_n=1;          //the packet number
int checkid,sourceid,Ndepack,T2,T4,decheckid,destid,tbo;   //transient arguments
int n_dep_packet=0;           //number of depatures
double simul0=9999,tnow;    //initializing the start time of simulation.

//initializing the first row of system schedule
schenewsense(0,1,100,nei_ID);             //initializing the first arrival
ini_time[packet_n]=csmasched[0][5];    //recording the arrival time of the first packet
csmasched[0][1]=1;                         //initializing the event of the first arrival to 1.
packet_n++;                                //increment of arrivals
count--;                                   //decrement of nodes which haven't packets

//check point ERROR0 for isolated nodes.
if (csmasched[0][4]==99999)
{
    proutput<<"ERROR0:A NODE IS ISOLATED!!!!!!!!"<<endl;
    goto endsimul0;                        //abort simulation
}

//***********************************//
//*********Processing Begins************//
//generates 1800 arrivals, collects statistics from 200-1500th arrival
//processes the first 1800 arrivals//
while(packet_n<1800)
{
    //prcsmasche<<"tnow="<<csmasched[0][2]<<endl;

    //**************** New arrival, event 1*********************//
    if(csmasched[0][1]==1)
    {
        tnow=csmasched[0][2];
        if(csmasched[0][0]==1500) //check the number of arrivals, if it is the 1500th packet
        goto endsimul;                    //simulation ends

//*****************The First Step**********************//
//**********Schedules the transmission of RTS*************//
i=getarraysize();                   //getting size of system schedule

//check point ERROR1 for finding schedule size bigger than the number of nodes.
//prints message, and aborts simulation.
if(i>100)
{
        proutput<<"ERROR1:Program stopped at getting size!!!!"<<endl;
        proutput<<"Packet Number is "<<packet_n<<endl;
        *d=0;                       //returning average delay as 0
        *g=0;                       //returning total arrival rate as 0
        goto endsimul0;             //aborting simulation
}

csmasched[i][5]=csmasched[0][5];     //insert arrival time at the end.

//sense BTr and BTt
if(REcheck(csmasched[0][3],neibool)==0)           //sensing BTr,0=no-BTr; 1=BTr
{   //no BTr

        if(TRcheck(csmasched[0][3],4,neibool)==1) //sensing BTt,0=no-BTt; 1=BTt
        {     // no BTr, but BTt is on.

                checkid=csmasched[0][3];        //recording source node of the pakcet
                decheckid=csmasched[0][4];  //recording destination node of the pakcet
                                            //deckeckid used to check hidden terminal

                //check whether all the BTts belongs to its neighbor
                for(int j=1;j<i;j++)
                {
                        if(csmasched[j][1]==4)          //check event 4, BTt signal on.
                        {
                                //recording source node of checked packets
                                sourceid=csmasched[j][3];
                                //recording destination node of checked packets
                                destid=csmasched[j][4];

                                if(neibool[checkid][sourceid]==1)
                                {     //neighbor's BTt is on.

                                        //can detect the BTt?
```

```
//arrival or sensing time after vulnerable time, pdelay+td
if(fabs(csmasched[i][5]-csmasched[j][5])>COLLtime)
{
    //node can sense ongoing RTS successfully
    if(csmasched[i][6]!=1)  //skip the colliding packets
    {
        tbo=countbackoff(i)-1; //get backoff times
        if(tbo>timeout)         //timeout
        {
            //insert source node at the end of
            //No-packet array
            nopacket[count]=csmasched[0][3];
            //increasing the number of nodes which is
            //idle.
            count++;
            csmasched[i][0]=0;
            csmasched[i][1]=0;
            csmasched[i][2]=0;
            csmasched[i][3]=0;
            csmasched[i][4]=0;
            csmasched[i][5]=0;
            csmasched[i][6]=0;
            i--;
          //discarding the timeout packet, and ending
          //the first step.
            goto newsensing;
        }

        //BTt signal is sensed
        //sechdules sensing/resensing, event 3
        csmasched[i][1]=3;
        //setting backoff time
        csmasched[i][2]=tnow+pow(2,tbo)*backoff();
        //ending the first step
        goto endevent1RTS;
    }
}

//cannot detect neighbot's BTt, a collision occurs
//flag a collision for both packets
else
{
    csmasched[i][2]=tnow+BTttime; //scheduling RTS
    csmasched[i][5]=tnow;   //recording RTS start time
```

```
                              csmasched[i][1]=4;     //setting event 4, RTS
                              csmasched[i][6]=1;       //setting 1--collision
                              csmasched[j][6]=1;       //setting 1--collision
                              //ending the first step
                         }
                    }
               }
          }
     }

     //no BTt or BTr
     //schedules RTS,event 4
     else
     {
          csmasched[i][2]=tnow+BTttime;        //schedule RTS
          csmasched[i][5]=tnow;                //record RTS start time
          csmasched[i][1]=4;                   //set event 4.
          goto endevent1RTS;
     }
}

//BTr is on.
//schedules sensing/resensing, event 3,
else
{
     tbo=countbackoff(0)-1;     //get backoff times
     if(tbo>timeout)            //check timeout
     {
          //timeout, discards the packet
          //inserting source node at the end of nopacket array
          nopacket[count]=csmasched[0][3];
          count++;                    // increasing the number of nodes which is idle.
          csmasched[i][0]=0;
          csmasched[i][1]=0;
          csmasched[i][2]=0;
          csmasched[i][3]=0;
          csmasched[i][4]=0;
          csmasched[i][5]=0;
          csmasched[i][6]=0;
          i--;
          //discarding the timeout packet, and ending the first step.
          goto newsensing;
     }
```

```
                    //no time out, schedules sensing/resensing, event 3
                    csmasched[i][2]=tnow+pow(2,tbo)*backoff();     //setting backoff
                    csmasched[i][1]=3;                             //setting event 3.
                }

//ends the first step, inserts the new scheduling at the end
endevent1RTS:
                csmasched[i][3]=csmasched[0][3];
                csmasched[i][4]=csmasched[0][4];
                csmasched[i][0]=csmasched[0][0];

                //*********The second Step**************//
                //*******Schedules next new arrival*******//
                //if a new generation is scheduled within the vulnerable time of its neighbors' RTS
                //transmission,
                //the system schedules immediately the RTS transmission for the new generation,
                //and continues generating new arrival till successfully setting event 1 to a new
                //generation
                //if setting event 1 to a new generation is successful, ends the second step.
                //newsensing:
                //check point ERROR for system is blocked.
                if (count==0)     //all nodes have packets, system is blocked, abort simulation.
                {

                    proutput<<"ERROR: The system is blocked. Simulation is ended!"<<endl;
                    proutput<<"Packet Number is "<<packet_n<<endl;
                    *d=0;                       //returning average delay as 0
                    *g=0;                       //returning total arrival rate as 0
                    goto endsimul0;             //abort simulation
                }

                i++;                            //used to insert at the end of schedule
                schenewsense(i,packet_n,count,nei_ID);   //scheduling new generation.
                ini_time[packet_n]=csmasched[i][5];      //recording the packet initial time.

                //check point ERROR2 for isolated nodes.
                if (csmasched[i][4]==99999)
                {
                    proutput<<"ERROR2:A NODE IS ISOLATED!!!!!!!!"<<endl;
                    goto endsimul;                  // abort simulation
                }

                //sets starting time of simulation at the 200th arrival.
                if(csmasched[i][0]==200)
```

```
     simul0=csmasched[i][2];       //recording the start time.

//counts new arrivals during simulation period
if (csmasched[0][2]>=simul0)
     Sumnewarr++;

//gets next generating successfully
packet_n++;
count--;

//check BTt signal.
//No BTt signal is on, no ongoing RTS transimission.
//schedul new arrival successfully, abort sensing.
if(TRcheck(csmasched[i][3],4,neibool)==0)   //0=no BTt; 1=BTt
{
     //no BTt, sets event 1 for the new generation, ends the whole processing.
     csmasched[i][1]=1;
     goto endsening;
}

//BTt is on.
else
{
     checkid=csmasched[i][3];      //recording source of processed packet
     decheckid=csmasched[i][4];   //recoreing destination of processed packet

     for(int j=1;j<i;j++)
     {
          if(csmasched[j][1]==4)
          {
               sourceid=csmasched[j][3]; //source of checked packets
               destid=csmasched[j][4];      //destination of checked packets

               //check whether all the BTts belongs to its neighbor
               if(neibool[checkid][sourceid]==1)      //1=neighbor
               {
                    //neighbors' BTt signals are on
                    //can detect the BTt? Yes, no collision; No, collision
                    if(fabs(csmasched[i][5]-csmasched[j][5])>COLLtime)
                    {
                         //NO collision.
                         //arrival  or  sensing  time  is  after  vulnerable  period
                          //pdelay+td
                         //node can sense ongoing RTS successfully
```

```
                                        csmasched[i][1]=1;        //setting new arrival, event 1
                                        goto endsening; //ending the whole processing of event 1
                                    }
                                    else
                                    {
                                        //collision, sets collision to both packets
                                        //continues to schedule a new generation
                                        //scheduling RTS
                                        csmasched[i][2]=csmasched[i][5]+BTttime;
                                        csmasched[i][1]=4;
                                        csmasched[i][6]=1;                    //setting 1--collision
                                        csmasched[j][6]=1;                    //setting 1--collision
                                    }
                                }
                            }
                        }
                        goto newsensing;              //continuing to schedule a new generation
                    }

endsening:      //end of processing event 1
                upmove();        //deleting the processed schedule
                bubbleSort();  //sorting system schedule by bubble sorting of csmasched[][2].

                //if (csmasched[0][2]>=simul0)
                //prsche();          //print out system schedule after the 200th packet
                //prnopacket();  //print out the No-packet array.
            }


            //***************Departure, Event 2********************//
            //inserts source node back to nopacket array,
            //calculates the total delay of the packet
            //summates the number of departures
            else if(csmasched[0][1]==2)
            {
                tnow=csmasched[0][2];

                //inserting source node at the end of No-packet array
                nopacket[count]=csmasched[0][3];
                count++;                        //increasing the number of nodes without holding packet.
                //recording the number of the departing packet, used to find its arrival time
                Ndepack=csmasched[0][0];

                //summates delay of all depart packets.
```

```
    if(csmasched[0][2]>=simul0)
    {
            Sumdelay+=tnow-ini_time[Ndepack];    //total delay
            n_dep_packet++;                       //the number of departures.
    }

    //end of processing event 2
    upmove();           //deleting the processed schedule
    //if (csmasched[0][2]>=simul0)
    //prsche();       //print out system schedule after the 200th packet
    //prnopacket(); //print out the No-packet array
}

//*************Sensing/Resensing, Event 3*******************//
//same as the first step of processing the new arrival event 1,
//but ends without scheduling next new arrival
//another difference is that the event 3 is inserted at the end of the system schedule
else if(csmasched[0][1]==3)
{
    tnow=csmasched[0][2];
    i=getarraysize();                   //getting size of system schedule

    //counts old arrivals during simulation period
    if(csmasched[0][2]>=simul0)
        Sumoldarr++;

    //check point ERROR3 for finding a sensing/resensing packet with a flag of
    //collision.
    if(csmasched[0][6]==1)
    {
        proutput<<"ERROR3: COLLIDE AT EVENT 3!!!!!!!!!!!!!!!!!"<<endl;
        goto endsimul0;                 //abort simulation
    }

    //Sense BTr
    if(REcheck(csmasched[0][3],neibool)==0) //0=no BTr; 1=BTr
    {
        //no BTr
        //Senses BTt
        if(TRcheck(csmasched[0][3],4,neibool)==1)    //1=BTt
        {
            //no BTr, but BTt signal is on.
            checkid=csmasched[0][3];    //recording source of processed packet
            decheckid=csmasched[0][4]; //recording destination of processed packet
```

```
for(int j=1;j<i;j++)
{

    if(csmasched[j][1]==4)
    {
        sourceid=csmasched[j][3]; //record source of checked packets
        destid=csmasched[j][4]; //record destination of checked pack

        //checks whether the BTt is belongs to neighbors.
        if(neibool[checkid][sourceid]==1)
        {
            //neighbors' BTt is on.
            //checks whether the BTt can be detected
            if(fabs(csmasched[0][5]-csmasched[j][5])>COLLtime)
            {
                //detected BTt, no collision.
                //skipping the colliding packets
                if(csmasched[0][6]!=1)
                {
                    tbo=countbackoff(0)-1; //getting backoff time
                    if(tbo>timeout)          //checking timeout
                    {
                        //timeout
                        //insert source node at the end of
                        //No-packet array
                        nopacket[count]=csmasched[0][3];
                        //increasing the number of nodes which is
                        //idle
                        count++;
                        upmove();  //deleting the processed row
                        goto endresensing;
                    }
                    //no timeout
                    csmasched[0][1]=3;            //setting event 3
                    csmasched[0][2]=tnow+pow(2,tbo)*backoff();
                    goto endresensing;

                }
            }

            //collision
            //flags collision for both packets
            else
```

```
                              {
                                   //scheduling RTS for the sensing packet
                                   csmasched[0][2]=tnow+BTttime;
                                   csmasched[0][5]=tnow;      //record start RTS time.
                                   csmasched[0][1]=4;
                                   csmasched[0][6]=1;        //setting 1--collision
                                   csmasched[j][6]=1;           //setting 1--collision
                              }
                         }
                    }
               }
          }
          //no BTt or BTr,
          //successfully sechedules RTS transmission
          else
          {
               csmasched[0][1]=4;                    //setting event 4.
               csmasched[0][5]=tnow;                  //recording the starting time of RTS.
               csmasched[0][2]=tnow+BTttime;     //scheduling RTS Transmission.
               goto endresensing;                     //ending the processing of event 3
          }
     }

     //BTr is on.
     else
     {
          tbo=countbackoff(0)-1;     //getting backoff times.
          if(tbo>timeout)             //checking timeout
          {
               //timeout
               nopacket[count]=csmasched[0][3];
               count++;                             //increasing the number of idle nodes
               upmove();                            //deleting the processed row
               goto endresensing;                  //ending the processing of event 3
          }

          //no timeout
          //schedules sensing/resensing, event 3
          csmasched[0][2]=tnow+pow(2,tbo)*backoff();      //backoff
          csmasched[0][1]=3;
     }

endresensing:
     bubbleSort();    //sorting schedule by bubble sorting of csmasched[][2].
```

```
    //if (csmasched[0][2]>=simul0)
    //prsche();      //print out system schedule after 200th packet
    //prnopacket();//print No-packet array.
}


//**************RTS Transmission, Event 4***********************//
//if finds collisions flags, schedules sensing/resensing, event 3.
//no collisions, schedules the data transmission after ensuring that no ongoing
//transmission can prevent the channel initialization.
else if(csmasched[0][1]==4)
{
    tnow=csmasched[0][2];
    T2=TRcheck(csmasched[0][4],2,neibool);      //checking ongoing data tr
    T4=TRcheck(csmasched[0][4],4,neibool);      //checking ongoing RTS tr.

    if(csmasched[0][6]==1)      //checking the flag of collision
    {
        //collision is found
        tbo=countbackoff(0)-1;   //get backoff times
        if(tbo>timeout)            //checking timeout
        {
            //timeout
            nopacket[count]=csmasched[0][3];
            count++;                          //increasing the number of idle nodes
            upmove();                         //deleting the processed row
            goto end4nothing;                 //ending the event 4
        }

        //no timeout
        //schedules sensing/resensing, event 3
        csmasched[0][1]=3;
        csmasched[0][6]=0;                         //clearing the flag of collision
        csmasched[0][2]=tnow+td+pow(2,tbo)*backoff()+2*pdelay;      //backoff
    }

    //checks any interfering ongoing transmission
    else if(T2||T4==1)
    {
        //interference from ongoing transmissions
        tbo=countbackoff(0)-1;
        if(tbo>timeout)
        {
                //timeout
                nopacket[count]=csmasched[0][3];
                count++;                          //increasing the number of idle nodes.
                upmove();                         //deleting the processed row
                goto end4nothing;                 //ending the event 4
            }

            //no timeout, schedules event 3
            csmasched[0][1]=3;
            csmasched[0][6]=0;                    //clearing the flag of collision
            //scheduling fast retransmission
            csmasched[0][2]=tnow+td+2*pdelay+2*RTStime;
        }

        //no interference, schedules departure event 2
        else
        {
            csmasched[0][1]=2;
            csmasched[0][2]=tnow+BTrtime;
        }
        //end processing;
end4nothing:
        bubbleSort();    //sorting schedule by bubble sorting csmasched[][2].

        //if (csmasched[0][2]>=simul0)
        //prsche();        //print out system schedule after the 200th packet
        //prnopacket();  //print No-packet array.
    }
}
prcsmasche<<"end"<<endl;  //end of the process at the first 1800 packets arrivals

//*************************************//

//--------------------------------------------------------//
//processes the remainers after 1800 packets arrivals
//END at the 1500th packet, simulation is ended
//the ONLY difference between the processing of the first 1800 packets is that no new arrival
//event 1 is scheduled
//event 1 and event 3 are regarded as SAME.
while(csmasched[0][0]!=0)
{
    //---------------New Arrival, Event 1 & Sensing/Resensing, Event 3---------------------//
    if(csmasched[0][1]==1||csmasched[0][1]==3)
    {
```

```
if(csmasched[0][0]==1500)           //simulation end at the 1500th packet
      goto endsimul;

i=getarraysize();
tnow=csmasched[0][2];

if(i>=100)
{
      proutput<<"ERROR0:The system has problem"<<endl;
      proutput<<"Packet Number is "<<packet_n<<endl;
      *d=0;
      *g=0;
      goto endsimul0;
}

if(csmasched[0][6]==1)
{
      proutput<<"ERROR4:COLLIDE AT EVENT 1/3!!!!!!!!!!!!!!!!!!"<<endl;
      goto endsimul;
}

if((csmasched[0][2]>=simul0)&&(csmasched[0][1]==3))
      Sumoldarr++;

//---------------- RTS Transmission, Event 4--------------------//
if(REcheck(csmasched[0][3],neibool)==0)
{
      if(TRcheck(csmasched[0][3],4,neibool)==1)
      {
            checkid=csmasched[0][3];
            decheckid=csmasched[0][4];

            for(int j=1;j<i;j++)
            {

                  if(csmasched[j][1]==4)
                  {
                        sourceid=csmasched[j][3];
                        destid=csmasched[j][4];

                        if(neibool[checkid][sourceid]==1)
                        {
                              if(fabs(csmasched[0][5]-csmasched[j][5])>COLLtime)
                              {
```

```
                        if(csmasched[0][6]!=1)
                        {
                            tbo=countbackoff(0)-1;
                            if(tbo>timeout)
                            {
                                nopacket[count]=csmasched[0][3];
                                count++;
                                upmove();
                                goto endnewres;
                            }
                            csmasched[0][1]=3;
                            csmasched[0][2]=tnow+pow(2,tbo)*backoff();
                            goto endnewres;
                        }
                    }

                    else
                    {
                        csmasched[0][2]=tnow+BTttime;
                        csmasched[0][5]=tnow;
                        csmasched[0][1]=4;
                        csmasched[0][6]=1;
                        csmasched[j][6]=1;
                    }
                }
            }
        }
    }
    else
    {
        csmasched[0][1]=4;
        csmasched[0][5]=tnow;.
        csmasched[0][2]=tnow+BTttime;
        goto endnewres;
    }
}
else
{
    tbo=countbackoff(0)-1;
    if(tbo>timeout)
    {
        nopacket[count]=csmasched[0][3];
        count++;
        upmove();
```

```
                    goto endnewres;
                }
                csmasched[0][2]=tnow+pow(2,tbo)*backoff();
                csmasched[0][1]=3;
            }

endnewres:
            bubbleSort();

            //if (csmasched[0][2]>=simul0)
            //prsche();
            //prnopacket();
        }

        //---------------Departure, Event 2--------------------------------//
        else if(csmasched[0][1]==2)
        {
            tnow=csmasched[0][2];
            nopacket[count]=csmasched[0][3];
            count=count+1;
            Ndepack=csmasched[0][0];

            if(csmasched[0][2]>=simul0)
            {
                Sumdelay+=tnow-ini_time[Ndepack];
                n_dep_packet++;
            }

            upmove();
            //prsche();
            //prnopacket();
        }

        //-----------------RTS Transmission, Event 4---------------------------------//
        else if(csmasched[0][1]==4)
        {
            tnow=csmasched[0][2];
            T2=TRcheck(csmasched[0][4],2,neibool);
            T4=TRcheck(csmasched[0][4],4,neibool);

            if(T2||T4==1)
            {
                tbo=countbackoff(0)-1;
                if(tbo>timeout)
                {
                        nopacket[count]=csmasched[0][3];
                        count++;
                        upmove();
                        goto end14nothing;
                }
                csmasched[0][1]=3;
                csmasched[0][6]=0;
                csmasched[0][2]=tnow+td+2*RTStime+2*pdelay;
                //csmasched[0][2]=tnow+td+pow(2,tbo)*backoff()+2*pdelay;
            }

            else if(csmasched[0][6]==1)
            {
                tbo=countbackoff(0)-1;
                if(tbo>timeout)
                {
                        nopacket[count]=csmasched[0][3];
                        count++;
                        upmove();
                        goto end14nothing;
                }

                csmasched[0][1]=3;
                csmasched[0][6]=0;
                csmasched[0][2]=tnow+td+pow(2,tbo)*backoff()+2*pdelay;
            }

            else
            {
                csmasched[0][1]=2;
                csmasched[0][2]=tnow+BTrtime;
            }

end14nothing:
            bubbleSort();

            //prsche();
            //prnopacket();
        }
    }
    //---------------------------------------------------------------//

    //COLLECTS STATISTICS OF SIMULATION
```

```
endsimul:
    simultime=csmasched[0][2]-simul0;          //total simulation time
    *d=Sumdelay/n_dep_packet;                  //average delay
    *g=(Sumnewarr+Sumoldarr)/simultime;        //total arrival rate.
    *a=n_dep_packet/simultime;                 //average throughput

endsimul0:        //print out simulation results of each run
    prcsmasche<<"simulation time="<<simultime<<endl;
    proutput<<"Simulation start: "<<simul0<<endl;
    proutput<<"Total number of departures = "<<n_dep_packet<<"."<<endl;
    proutput<<"Average delay is "<<*d<<"."<<endl;
    proutput<<"Total number of new arrivals = "<<Sumnewarr<<"."<<endl;
    proutput<<"Total number of old arrivals = "<<Sumoldarr<<"."<<endl;
    proutput<<"simulation time = "<<simultime<<endl;
    proutput<<"Throughput= "<<*a<<"."<<endl;
    proutput<<"Total arrival rate = "<<*g<<"."<<endl;

    //prinitime();   //print initial time
}


//checks whether neighbors are transmitting RTS or data
// returns T value,1-neighbors are transmitting,0-no.
int Tbtma3_3::TRcheck(double id,int e,int nbool[][100])
{
    int size,T=0;
    int index=0;
    int sour_id,check_id;
    check_id=id;

    size=getarraysize();      //getting the size of the system schedule.

    for(index=1;index<size;index++)
    {
        if(csmasched[index][1]==e)        //check event.
        {
            sour_id=csmasched[index][3];
            if(nbool[check_id][sour_id]==1)
            {T=1;          //neighbor is transmitting data or RTS
            break;}
        }
    }
    return T;
}
```

```
//checks whether neighbors are receiving
// returns T value,1-neighbors are receiving,0-no.
int Tbtma3_3::REcheck(double id, int nbool[][100])
{
    int size,T=0;
    int index=0;
    int sour_id,check_id;
    check_id=id;

    size=getarraysize();      //getting the size of the system schedule.

    for(index=1;index<size;index++)
        {
            if(csmasched[index][1]==2)        //check event 2.
            {
                sour_id=csmasched[index][4];
                if(nbool[check_id][sour_id]==1)
                {T=1;           //neighbor is receiving
                goto ret;}
            }
        }
ret:  return T;
}

//gets random backoff time, uniformly distributed.
double Tbtma3_3::backoff()
{
    double w;
regetbackoff:
    w=static_cast<double>(rand())/RAND_MAX;
    if (w==0)
        goto regetbackoff;
    return w*0.5*datatime;
}

//gets new generation
void Tbtma3_3::schenewsense(int index,int packetN, int cont,int neib[][100])
{
    csmasched[index][0]=packetN;       //scheduling the ID of next arrival packet.
    csmasched[index][1]=3;             //setting event 3
    csmasched[index][2]=csmasched[0][2]+gettime();   //scheduling next sensing time.
    csmasched[index][5]=csmasched[index][2];
    //gets source node
    int sourcein,nei_num,sour_id,destin;
    if (cont!=0)
    {
        sourcein=get_node(cont);
        csmasched[index][3]=sour_id=nopacket[sourcein];

        //removes the node from No-packet array.
        nopacket[sourcein]=nopacket[cont-1];
        nopacket[cont-1]=0;

        //gets destination node
        nei_num=neib[sour_id][0];
        if (nei_num==0)
            //no neighbor is found, setting the destination node as 99999
            csmasched[index][4]=99999;
        else
        {
            //randomly picking up the index [1,neighbor number].
            destin=get_node(nei_num)+1;
            csmasched[index][4]=neib[sour_id][destin];   //getting destination.
        }
    }
    //no source node can be chosed
    else
    {
        csmasched[index][3]=csmasched[index][4]=999999;
        csmasched[index][1]=1;
    }

}

//gets random time with exponential distribution
double Tbtma3_3::gettime(void)
{
    double u;
regettime:
    u=static_cast<double>(rand())/RAND_MAX;   //scaling double type random number (0,1).
    if (u==1||u==0)
        goto regettime;
    return -log(1-u)/rate;                    //arrival_time=-ln(u)/arrival_rate, according to Poisson.
}
```

```
//radomly gets index of node
int Tbtma3_3::get_node(int mod)
{
    //scaling integers produced by rand()%mod; get random number(0,mod-1).
    return rand()%mod;
}


//deletes the first row of array.
void Tbtma3_3::upmove()
{
    int index=0;
    int size;
    size=getarraysize();

    for(index=0;index<size;index++)
    {
        csmasched[index][0]=csmasched[index+1][0];
        csmasched[index][1]=csmasched[index+1][1];
        csmasched[index][2]=csmasched[index+1][2];
        csmasched[index][3]=csmasched[index+1][3];
        csmasched[index][4]=csmasched[index+1][4];
        csmasched[index][5]=csmasched[index+1][5];
        csmasched[index][6]=csmasched[index+1][6];
        csmasched[index][7]=csmasched[index+1][7];
    }
}

//ascendes order in csmasched[][2]
void Tbtma3_3::bubbleSort()
{
    int size,index=0;
    double holdp,holde,holdt,holds,holdd,holdb,holdc,holdco;
    size=getarraysize();    //getting the size of the system schedule.

    if(size>1)
    {
    for(int index=1;index<size;index++)
        for(int j=0;j<size-1;j++)
            if (csmasched[j][2]>csmasched[j+1][2])
            {
                holdp=csmasched[j][0];
                holde=csmasched[j][1];
                holdt=csmasched[j][2];
                holds=csmasched[j][3];
                holdd=csmasched[j][4];
                holdb=csmasched[j][5];
                holdc=csmasched[j][6];
                holdco=csmasched[j][7];
                csmasched[j][0]=csmasched[j+1][0];
                csmasched[j][1]=csmasched[j+1][1];
                csmasched[j][2]=csmasched[j+1][2];
                csmasched[j][3]=csmasched[j+1][3];
                csmasched[j][4]=csmasched[j+1][4];
                csmasched[j][5]=csmasched[j+1][5];
                csmasched[j][6]=csmasched[j+1][6];
                csmasched[j][7]=csmasched[j+1][7];
                csmasched[j+1][0]=holdp;
                csmasched[j+1][1]=holde;
                csmasched[j+1][2]=holdt;
                csmasched[j+1][3]=holds;
                csmasched[j+1][4]=holdd;
                csmasched[j+1][5]=holdb;
                csmasched[j+1][6]=holdc;
                csmasched[j+1][7]=holdco;
            }
        }
}

//prints the No-packet array
void Tbtma3_3::prnopacket()
{
        for(int index=0;index<100;index++)
            printpacket<<setw(3)<<nopacket[index];
        printpacket<<endl;
}


//prints system schedule
void Tbtma3_3::prsche()
{
        //outputfile<<"schedule table (event 1--arrival,0--departure)"<<endl;
        prcsmasche<<"packet"<<setw(7)<<"event"<<setw(18)<<"time"
            <<setw(4)<<"S"<<setw(4)<<"D"
            <<setw(15)<<"starttime"<<setw(15)<<"collision"<<endl;
        int index=0;
        while((csmasched[index][0]!=0)&&(index<100))
        {
            prcsmasche<<setw(6)<<csmasched[index][0]
                  <<setw(5)<<csmasched[index][1]
                  <<setw(20)<<setprecision(11)<<csmasched[index][2]
                  <<setw(4)<<csmasched[index][3]
                  <<setw(4)<<csmasched[index][4]
                  <<setw(20)<<setprecision(11)<<csmasched[index][5]
                  <<setw(12)<<csmasched[index][6]
                  <<setw(12)<<csmasched[index][7]<<endl;
            index++;
      }
}

//gets size of array.
int Tbtma3_3::getarraysize()
{
      int index;
      for (index=1;index<102;index++)
      {
            if (csmasched[index][0]==0)
                  goto reindex;
      }

reindex:
      return index;
}

//counts backoff times of each packet.
int Tbtma3_3::countbackoff(int ii)
{
      return ++csmasched[ii][7];
}

//prints initial time of each packets
void Tbtma3_3::prinitime()
{
      for(int index=0;index<1600;index++)
            printinitime<<setprecision(8)<<ini_time[index]<<endl;

}
```