The University of Oklahoma
Graduate College

Achieving Low latency and High Packet Reception Ratio in Media Access Control Layer in VANET

A dissertation
Submitted To The Graduate Faculty
in partial fulfillment of the requirements for the
Degree of
Doctor Of Philosophy

by

Xianbo Chen
Norman, Oklahoma
2010

Achieving Low latency and High Packet Reception Ratio in Media Access Control Layer in VANET

A Dissertation Approved For The
School Of Electrical And Computer Engineering

By

Hazem H. Refai PhD, Chair

William Ray PhD

James Sluss PhD

Monte Tull PhD

Samuel Cheng PhD

To my lovely daughter Melody...

Acknowledgments

I would like to thank members of my doctoral committee, Dr. William Ray, Dr. James Sluss, Dr. Monte Tull, and Dr. Samuel Cheng for their advice and suggestions in the early stages of this work, and Dr. Xiaomin Ma from Oral Robert University for giving me the privilege of working with him. My appreciation also goes to my advisor, Dr. Hazem Refai, for his direction and aid in the fulfillment of this work, which would not have been possible without him. Lastly I thank my wife Huanhuan and my parents, for their constant support and understanding throughout this lengthy process.

Table of Contents

List of Figures

List of Tables

Abstract

Vehicular ad hoc networks (VANETs) or inter-vehicle communication (IVC) makes possible the development of a number of innovative and powerful transportation system applications. VANET technology proves an important extension of both cellular and wireless local area networks (WLANs) currently used in the transportation industry. It is widely recognized that the transportation industry serves as an ideal platform for a large number of existing and future wireless applications, many of which have yet to be developed for commercial use.

Safety messaging is one of the most critical uses for VANET, supporting a number of potential safety applications, e.g. emergency electronic brake lights, lane change and pre-crash warning, among others. Many applications require extremely low latency (less than 100ms) and highly reliable (over 99% packet delivery ratio) communication services. In order to satisfy these critical requirements, an efficient media access control (MAC) layer is necessary. At the time of this writing, a de facto standard of VANET MAC is being developed.

Extensive VANET MAC research with regard to safety applications has yet to be done. The proposed base for the VANET future standard uses an 802.11a media access layer whose performance-although studied-is known to contain deficiencies and was accomplished outside the VANET context. These factors motivated the author to initiate the study of VANET and MAC.

In this work, MAC for VANET MAC is extensively researched, and a history of MAC is initially reviewed. The special and critical requirements of VANET MAC are presented and four major categories were investigated and analyzed. Because the underdevelopment of 802.11p is based on the IEEE 802.11a, special consideration is given with regard to the performance of 802.11a MAC and associated requirements. Extensive research enhancements centering on safety applications of the 802.11 MAC are conducted. The author's research generated a platform in which VANET performance can be quantitatively evaluated, analyzed, and verified. The quantitative behavior of the current protocols/algorithms, which include delay and packet delivery ratio, are presented on this platform. Furthermore, the future protocol and algorithm proposals can be added into this platform so that a faster research cycle can be achieved. Through theoretical analysis and simulation, this investigation shows that current proposed VANET MAC and 802.11a MAC enhancements have yet met the critical requirements of VANET. The future work may focus on how to use this theoretical model and simulation tool to assist

MAC layer protocol design. Meanwhile, when new algorithms are proposed or accepted by the standard, this model and tool can serve as a fast and convenient platform, where the new algorithm can be easily added for the sake of evaluation and verification. The feasibility of relaxing some assumptions included therein, such as the hidden node problem in a two dimensional space, may also be studied to make the platform closer to a real system.

CHAPTER 1

Introduction

1.1 VANET

Although severity amelioration technologies, including seat belts, air bags and automatic braking systems (ABS), have been utilized for many years to provide passive protections to vehicle occupants in motor vehicles, nearly 6.2 million police-reported motor vehicle crashes occurred every 5 seconds on U.S. highways in 2005. On average, a person was injured in a police-reported motor vehicle crash every 12 seconds, and fatality occurred every 12 minutes [2].

Deployment of crash avoidance technologies such as cooperative collision avoidance (CCA) [3], has the potential to improve highway safety. Active protection is possible when motor vehicles are, in effect, able to observe what is happening around them, i.e. to foresee what will happen next, and advise motor vehicle operators accordingly of protective measures. As a result, for example, it is possible that fewer motor vehicles are involved in a chain collision. To achieve improved safety, motor vehicles must be equipped for vehicle-to-vehicle wireless communication of sufficient, promptly-conveyed information.

VANET addresses these aforementioned needs and requirements, and as shown in Fig. 1.1, VANET provides Intelligent Transportation System (ITS) capabilities that enhance existing applications and foster additional applications among which safety-related applications reside. To accomplish vehicle-to-vehicle communication, vehicles and roadside-units must have a common communication interface. However, such commonality has yet to be achieved among automobile manufacturers, wireless device vendors, and wireless service carriers. Hence, interoperability between vehicular communication devices is both costly and challenging.

Regardless of the final, agreed-upon VANET standard configuration, safety messaging must be supported. In [4], eight potential safety applications based on safety messaging were selected for further study:

- Traffic signal violation warning
- Curve speed warning
- Emergency electronic braking

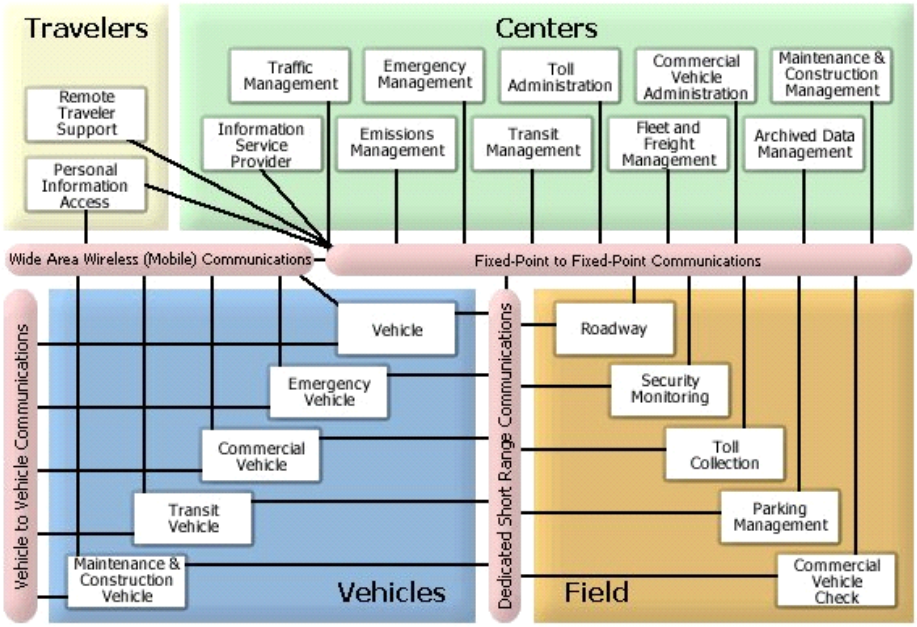

Figure 1.1: National ITS Architecture [1]

- Pre-crash warning lights
- Cooperative forward collision warning
- Left turn assistant
- Lane change warning
- Stop sign movement assistance

Potentially, these applications have the ability to enlarge a driver's vision range. A variety of VANET applications are illustrated in Fig. 1.2

The IEEE 1609 family of standards for wireless access in vehicular environments (WAVE) is currently under development with hope of solving interoperability issues. It is anticipated that the four standards serve as the foundation for a complete communication solution for potential VANET applications. The multi-channel operations are defined in IEEE P1609.4, which is based on IEEE 802.11p - the physical layer (PHY) and MAC layer amendment for WAVE to the predominant IEEE 802.11 technology. American Society for Testing and Materials (ASTM) first wrote the dedicated short range communication (DSRC), which is a synonym of WAVE. These terms are used interchangeably throughout this dissertation. Its PHY&MAC specifications are based IEEE 802.11a, as noted in one fo the standard documents. These were accepted and merged

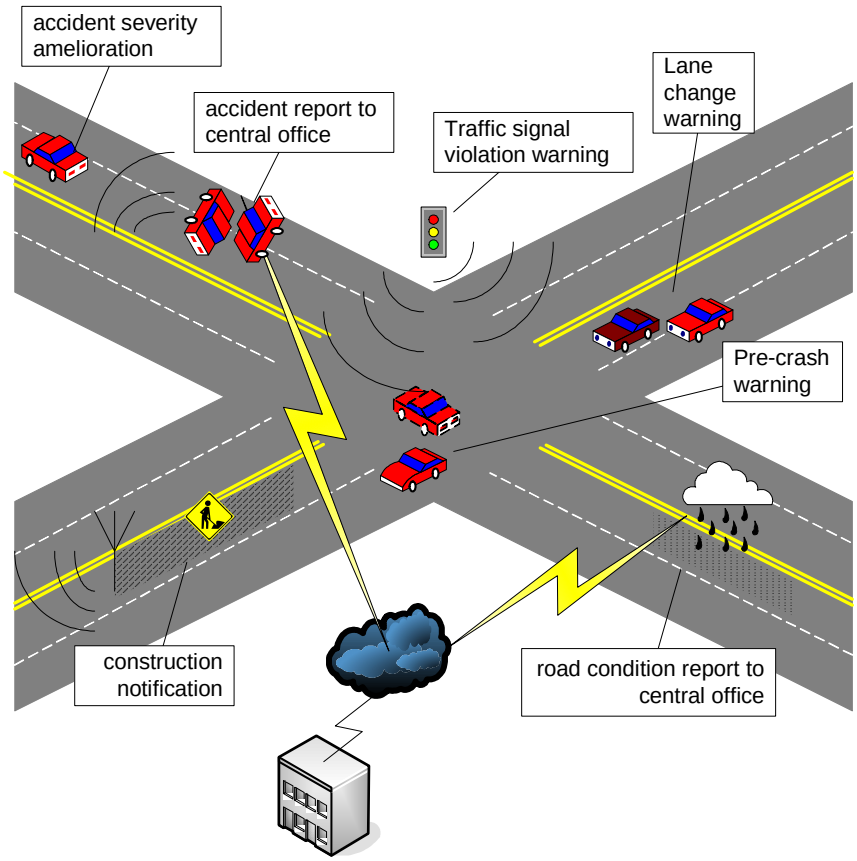

Figure 1.2: Safety-related VANET applications

into the IEEE 802.11 standard family as 802.11p, which is scheduled for publication in November 2010.

1.2 MAC History

The Media Access Control (MAC) data communication protocol sub-layer, also known as the Medium Access Control, is a part of the data link layer specified in the seven-layer OSI model (layer 2). It provides addressing and channel access control mechanisms making it possible for several terminals or network nodes to communicate within a multipoint network, typically a local area network (LAN) or metropolitan area network (MAN).

The MAC sub-layer acts as an interface between the Logical Link Control sublayer and the network's physical layer.

Media access control is often used as a synonym for multiple access protocol, since the MAC sublayer provides protocol and control mechanisms required for a certain channel access method. This makes it possible for several stations to share a connection to the same physical medium. Examples of a shared physical medium include bus networks (Ethernet), ring networks, and wireless networks.

1.3 Challenges And Requirements In VANET

The unique deployment environment of VANET and the nature of VANET safety critical applications (VSCAs) pose a number of technical challenges and requirements.

1.3.1 Point-to-point or point-to-multipoint: Unicast, broadcast, or multicast?

A wireless radio channel is, by its nature, broadcast because the wireless signal propagation is physically unbounded. Benefiting from this, point-to-multipoint communication is favorable for delivering safety-critical messages as soon as possible to as many neighboring vehicles as possible. For this reason, unicast is not proposed for VSCAs.

While both broadcast and multicast carry out point-to-multipoint communication, attention to their differences must be considered when designing VSCA communication protocols. They have yet to be clearly stated in the literature.

Broadcast transmits information to every node within its coverage, and yet the trans-

mitter does not recognize or count the number of receivers. As such, there is no practical and deterministic mechanism guaranteeing that the information can be delivered successfully to each potential receiver located within the broadcast range. The mechanism of negative acknowledgement (NAK) tone [5, 6, 7] can merely notify in general the failure of broadcast reception by a receiver but cannot indicate a specific receiver.

Multicast distributes information to a designated group whose identities namely location, quantity, addresses, and the like, should be known by the transmitter. As a result, a feasible mechanism can be conceived. Positive acknowledgement (ACK) from each receiver is but one example of a reliable information delivery mechanism.

In order to increase the broadcast's transmission reliability, countermeasures such as decreasing access collision, interference, and transmission bit error rate (BER) can be taken. In addition, ACK provides a deterministic success of transmission.

Multicast requires the sender to learn the identity of its receivers. This is achieved by continuously monitoring channel activities. Broadcast is relatively simpler, as it does not require this same procedure.

The broadcast nature of the communication is required for the development of VSCAs. A dedicated channel is proposed as a reserved conduit for safety messaging [4, 8].

The author posits that a mixed and cooperative use of broadcast and multicast should be of benefit for VSCAs. A possible combination algorithm is discussed in Chapter 6.

1.3.2 One-hop communication

It is intuitively understandable that the closer a vehicle is to an accident, the more dangerous the situation. Therefore, it is important that safety-critical messages are delivered as soon as possible to vehicles located within the vicinity of the message originator. In this way, a vehicle operator has ample time to take preventive measures. In other words, VSCAs are locally geo-significant.

When sending safety-critical messages to all local vehicles, two modes of communication are potentially of use, namely one-hop communication and multi-hop communication.

The designated transmission range of the DSRC transmitter is sufficient to cover the required VSCA vicinity [9, 10]. Following is the design specification of DSRC transceivers.

- Maximum Transmission power is 28.5 dBm which can cover 1000 m, and

- Receiver sensitivity is -77 dBm

Therefore, it can be concluded that the one-hop communication mode is supported or proposed for VSCAs in [11, 12, 9, 13, 14, 8, 4, 15].

1.3.3 Bounded latency

Latency is defined as the duration of time between the point at which a packet is generated and when it is successfully delivered. Because safety-critical events occur suddenly and are often short-lived, the effective and valid time span for countermeasures is accordingly short, a fact which challenges the bounded latency requirement underlying the communication system.

A $100ms$ latency requirement is proposed in [4] . For example, in an emergency electronic brake lights application, given the same deceleration rate and speed of 75 mph for every vehicle in a platoon, the 100ms latency criterion potentially prevents a chain collision at an even extreme headway value of around 4 meters, providing processing delays incurred from other subsystems are ignored. If instead, braking is dependent upon human reaction instead, minimum 32-meter headway is required [16].

The latency requirement is more stringent, i.e. around 20ms [4], for pre-crash warning applications.

1.3.4 Short packet length

Given a data rate, the shorter the length is, the shorter the transmission time. Furthermore, shorter packets are less exposed to the influence of channel burst error. Data packets that support most vehicle-to-vehicle communications were determined to be less than 100 bytes [4]. This poses a challenge to developing a compact protocol.

1.3.5 Adverse channel quality

The highly-dynamic vehicular environment is harsh. High mobility and varying transportation conditions result in a large Doppler effect and multi-path delay spread [17, 9, 18]. In such an environment, simulations show that the 802.11a/RA and DSRC physical layer exhibit high error rate [17, 9], which is confirmed in [18] through real-world experiments and measurements. The BER is highly sensitive to vehicle speeds and can reach an irreducible error floor ($> 10^{-3}$) when vehicles move at higher speeds [9].

A high-gain forward error correction mechanism or feedback mechanism is absolutely

necessary to combat adverse channel conditions existing in a VANET environment.

1.3.6 High packet delivery ratio (PDR)

High packet delivery ratio (PDR) is defined as the ratio of the number of broadcast packets received successfully by all receivers to the total number of packets sent. An accurate and timely information delivery is needed to alleviate and hopefully avoid safety-critical events. Thus a high delivery ratio (99% [19]) of packets conveying information must be achieved and guaranteed.

Three factors impede VANET from reaching the desired PDR.

1. Access collision. Packets transmitted by terminals collide when they transmit simutaneously. VANET's nature of shared broadcast medium and of distributed system makes access collision unavoidable. The hidden terminal problem further aggravates access collision. Such collisions can be alleviated by applying appropriate MAC protocols to coordinate terminal access activities.

2. Interference. This can be alleviated by either an appropriate physical layer algorithm design or the cooperation between MAC layer and physical layer - for transmission power control (TPC).

3. Multi-path fading and Doppler effect. This factor is discussed in the previous subsection.

Two of the aforementioned factors can be battled by improving MAC designs and implementations.

1.3.7 More severe hidden terminal problem

Since broadcast is recommended for VSCAs, the hidden terminal problem is more problematic, as illustrated in Fig. 1.3. A unicast scenario is shown in Fig. 1.3(a). The shaded area is the potential hidden terminal area, which is determined by the only one transmitter-receiver pair. The potential hidden terminal area is the area where all the potential hidden terminals might exist. In a broadcast scenario shown in Fig. 1.3(b), the potential hidden terminal area is apparently much larger because more than one receivers exist therein.

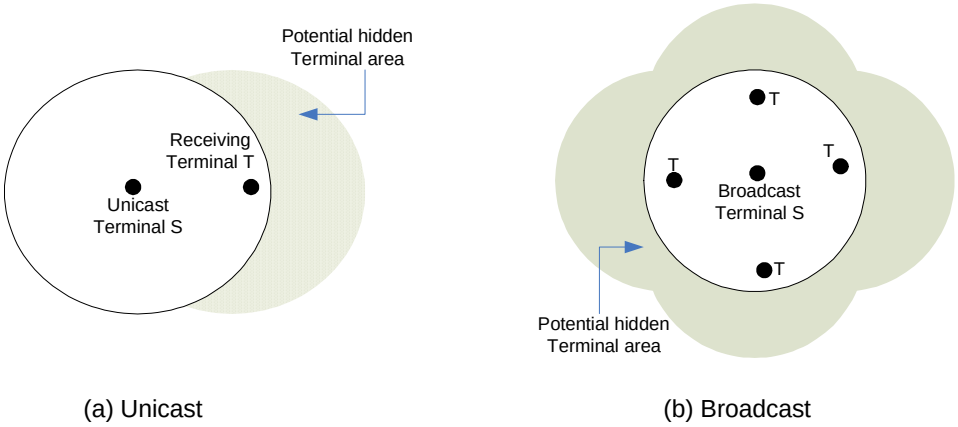

Figure 1.3: Hidden terminal problems

1.3.8 Other considerations

Link life for vehicles traveling in opposite directions [13] can be extremely short and possibly devastating for multi-hop communications, as a result of its one-hop nature, short packet length, high data rate, and 100 ms bounded delay requirement.

However, for safety messaging, mobility does not aggravate the hidden terminal problem because DSRC data rate is high and safety messages are short. Given the 100 ms delay bound and 120 mph speed, a vehicle can, at most, move approximately three meters. In other words, no more hidden terminals are activated when safety-related messages are being sent.

The research conducted in this work has the following contributions to the research community in the area of study:

- Thorough survey on the existing VANET MAC layer enhancement proposals.
- First identified the CFP process in the 802.11 protocol.
- Built a Markovian model for 802.11 one-hop broadcast MAC layer performance analysis
- Developed a simulation Platform for 802.11a based VANET MAC simulation.

The organization of this dissertation is as follows. Chapter 2 covers the current VANET MAC proposals and existing enhancements to IEEE 802.11 MAC. Chapter 3 compares the performance between DSRC MAC and R-ALOHA MAC. A theoretical model is presented in Chapter 4 for performance analysis of IEEE 802.11 MAC in a one-hop broadcast network under saturation traffic condition. The performance in a

more realistic environment with highway topology and un-saturated traffic is analyzed in Chapter 5. A simulation platform for 802.11-based VANET is presented in Chapter 4 and Chapter 5. Chapter 6 gives the conclusion and suggestions for future work.

CHAPTER 2

Background and Comparison

IEEE 802.11p is the proposed MAC protocol for VANET applications. At the time of this writing, researchers have proposed different IVC MAC schemes, which, to the author's best knowledge, consist of four general schemes. In this section, each scheme is briefly depicted and analyzed.

2.1 Space division multiple access (SDMA)

SDMA was proposed for inter-vehicle communication in [20, 21, 22]. Fundamentally, its concept is to partition the geographical area into multiple divisions and to map each division to a channel or channels. Thus, the function of partitioning divisions (P-FUNC) and the function of mapping divisions to channels (M-FUNC) are two critical SDMA functions. Meanwhile, a real-time and accurate positioning system is mandated. While SDMA might be efficient for straight highway scenarios, it is difficult to deploy as a general IVC MAC solution because homogenous, stable, and optimal P-FUNC and M-FUNC are hard to derive for various transportation scenarios. As such, previously derived functions target straight highway scenarios, however, as noted above, these cannot be generalized for all cases. For example, in urban areas where roads are meshed, its format and ambient are subject to continuously change. If P-FUNC and M-FUNC are developed for each road in accordance with the algorithm for straight highway scenarios [20, 21, 22], several issues might arise. These are explained in the following sections.

2.1.1 The workload of generating P-FUNC and M-FUNC will be tremendous

For straight parallel highways, such as the one shown in Fig. 2.1, it is relatively uncomplicated to generate P-FUNC and M-FUNC.

However, realistic situations are not always this simple. Fig. 2 illustrates a meshed road structure [23]. Generating P-FUNC and M-FUNC for this scenario is more complex. SDMA is accomplished manually on a single scenario basis, and because there are thousands of different meshed road structures in this world, manually generating fucntions for each may not be a sound solution. An automatic generation of P-FUNC and M-FUNC is indeed necessary for SDMA MAC solutions.

Figure 2.1: Example of straight parallel highway

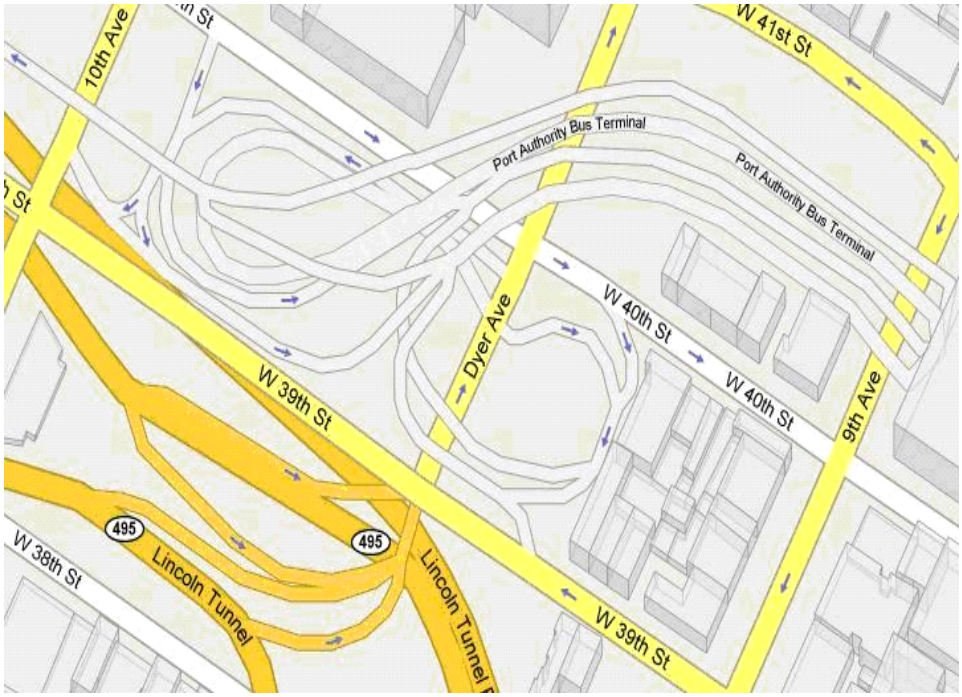

Figure 2.2: Example of meshed road structure (NYC)

2.1.2 Updating P-FUNC and M-FUNC in all vehicles in a synchronous and timely manner will be hardly achievable

The homogeneity of P-FUNC and M-FUNC in all vehicles must be maintained in order to provide reliable communication networks. In real world scenarios, road construction is constant as old roads are torn down and new are built. Road changes will likely trigger a change in P-FUNC and M-FUNC. Therefore managing change and distributing vehicle updates proves a difficult design challenge for SDMA MAC-based VANET systems.

2.1.3 Other issues

Multiple vehicles occupying a single division, FDMA simultaneous multiple receptions, TDMA bounded latency, division border effect, and position accuracy, add complexity to the vehicular SDMA design and implementation.

So although the use of SDMA might theoretically achieve a minimum number of collisions and bounded delays under certain assumptions, the method might also suffer feasibility issues.

2.2 ADHOC MAC

ADHOC MAC [24, 25, 26, 27, 28] is a protocol developed within the CarTalk2000 project [29, 30] funded by the European Commission. The basis of ADHOC MAC is Reliable R-ALOHA (RR-AHOHA). It is well-known that R-ALOHA can effectively coordinate channel usage in centralized networks. RR-ALOHA aims to fulfil the role as R-ALOHA, albeit in a dynamic and distributed manner. Essentially, when running RR-ALOHA protocol, it is expected that the setup of a reliable mobile ad hoc wireless communication is accomplished through the timely exchange of TDMA slot usage information.

Two primary advantages are expected when deploying RR-ALOHA for VANET.

1. Hidden terminal problems and exposed terminal problems are greatly alleviated, and
2. Highly reliable one-hop unicast, one-hop broadcast, and multi-hop broadcast are supported.

However, whether or not VANET can make use of these two advantages is truly dependent upon the speed and accuracy of exchanging slot usage information among

vehicles. Research in [25] shows

1. at least two frame cycles must be spent reserving a slot, and
2. the probability of achieving the reservation in two frame cycles is merely around 50%, and
3. the frame length given is 100 milliseconds.

These factors indicate that a minimum time ($> 200ms$) is needed to successfully obtain the basic channel (BCH) in a static scenario and that the average time is even longer. Furthermore [26], if mobility and heavier offered-traffic are considered, additional time to compensate for access collision is required for vehicles to constantly release/request slots. The short link life shown in [13] and the possible lengthy time required for a new terminal to acquire the BCH implies that vehicles moving in opposite direction might not be able to renew their BCHs and communicate when passing each other. Meanwhile the latency of safety-critical messaging recommended in [4] is only 100 ms. Further study of RR-ALOHA is needed to show whether or not a vehicle with emergency information and without slots can obtain a slot in a timely manner.

In addition, questions relating to the verification of RR-ALOHA suitability for VANET should be answered.

2.2.1 How to determine the optimal number of slots in one frame?

It is difficult to dynamically adjust the size of the frame in VANET; hence it should be fixed. Size is related to other parameters, such as typical packet length, minimum access delay, data rate, and network capacity, among others.

2.2.2 How to solve possible collision when multiple nodes contend for the same slot?

The probability of collision is foreseen to increase when more than one node competes for available slots. A mechanism must be provided to coordinate the contention for slots among multiple terminals. In [24, 25, 27, 28], a transmission probability p was set as

$$p = \frac{1}{k} \tag{2.1}$$

k is derived from

$$k = M - x \tag{2.2}$$

where M is the fixed maximum number of terminals in any two-hop cluster and x is the number of terminals that have already acquired the channel.

This algorithm proves conservative when the actual number of terminals is much less than M and too aggressive when the number is greater than M.

Since it is impossible to know how many terminals are actually contending for slots at any given time according to the RR-ALOHA mechanism, it is a significant challenge to obtain the optimal p and avoid access collision. J. J. Blum et al [13] showed similar concern.

Previous literature assumes an ideal wireless channel. Wireless communication is, however, notorious for its harsh channel quality. Because the BER in VANET is high [17, 9, 18] and frame information (FI) of RR-ALOHA can be as long as 2500 bits per slot, the error rate might be extremely high and the RR-ALOHA mechanism might suffer tremendously. Accurate RR-ALOHA operations are based on the successful FI decoding. An inaccurate FI transmission will cause corresponding terminals to either erroneously relinquish the reserved slots or continuously fail slot acquisition.

2.2.3 How to effectively reduce RR-ALOHA protocol overhead?

The total RR-ALOHA overhead can be as high as 2500 bits [25]. The average length of safety-related messages, however, is only 200 to 400 bytes [4]. This low efficiency (44% to 56%) might not be acceptable.

2.3 D-MAC

Directional MAC (D-MAC) exerting directional antennas has been thoroughly researched for packet radio networks [31, 32, 33, 34, 35, 36, 37].

Addtionally, certain aspects of the application of directional antennas in wireless ad hoc networks have been researched [38, 31, 39, 40, 41, 42, 43]. In summary, the advantages of applying directional antenna in wireless ad hoc network, i.e. throughput increase due to simultaneous transmissions, are achieved only if the following two key constraints are addressed:

1. Constrain the wave propagation between the source and destination (the beam) as narrowly as possible.

2. If such a narrow constraint is achievable, coordinate multiple simultaneous in-

stances of such narrow transmissions.

The literature often assumes a directional antennas consist of ideally non-overlapping M beams and that the span of each beam is calculated as $360/M$ degree. It is obvious that on one hand the larger M, the narrower the beam. On the other hand, an increase of M will increase the hardware design cost, e.g. antenna interference, multiple transceivers, and the like.

Terminal coordination is dependent on the presence of the following information at each terminal:

- the antenna (beam) the terminal should use to transmit to the neighbor, and
- the antenna (beam) its neighbor uses to receive transmission.

There are two ways to accomplish these tasks. One is that the terminal knows its neighbors' position information [31, 39, 40, 41, 42, 43]. The other one is that a terminal constantly listens to the channel and subsequently builds a location table [38].

The DMAC proposal using directional antennas is feasible for static or relatively low dynamic ad hoc networks. However, for VANET applications, this needs to be further improved and evaluated.

The transportation ambient is complicated. Many obstacles can block line-of-sight for directional antenna beams. Even when location information of a receiver is known, it might be impossible for a direct beam link between transmitter and receiver. Also, building the location table could be time consuming, which is in conflict with the real-time requirement of VANET.

Repeaters are required at intersections and other places where vehicles line-of-sight is needed, not possible for communication. This phenomenon necessitates additional investment in VANET.

With high mobility in VANET, antenna element usage must be constantly changed. This might considerably complicate the inter-beam control procedure. The impact of mobility for an omni-directional antenna based system was studied in [44] and it was deemed trivial for safety-related message broadcast.The impact of mobility for DMAC should be studied.

D-MAC performance for VANET was investigated in [45, 42]. In [45], D-MAC's broadcast performance, namely throughput and delay, was shown to be lower than omni-MAC.

Other possible issues regarding the usage of directional antennas in VANET, e.g. security consideration and the hidden terminal problem are discussed in [13].

2.4 802.11p MAC

A brief introduction of the evolution history of 802.11p can be found in Section 1.1. The distributed coordination function (DCF) serves as its core mechanism [46].

The simplicity is a prominent advantage when compared with other MAC shecmes. However, this simplicity is achieved at the sacrifice of other performance parameters. This and other shortcomings are further investigated in Chapter 4 and Chapter 5.

SDMA, RR-ALOHA, and DMAC have not been pervasively deployed in WLAN. With regard to market applications, IEEE 802.11p has a solid foundation resultant from the ubiquity of IEEE 802.11 networks (Wi-Fi). From an engineering perspective, it is advantageous and prudent to inherit core mechanisms from the predecessor, unless, of course, new applications are not compatible with the previous technology. The predominance of IEEE 802.11 in the market, coupled with the active development of IEEE 802.11p and IEEE 1609 standards suggest that the 802.11 based MAC has a greater potential of acceptance as the de facto VANET MAC than the three MAC protocols mentioned above.

Several limitations of 802.11 MAC, including unfairness, high collision probability in large networks, and unreliable broadcast service, are well known. Addressing these limitations has resulted in a vast amount of research with the aim to increase the performance of IEEE 802.11 MAC [46].

Because 802.11p is currently under development based on 802.11, inherent limitations might persist. Furthermore, recent VANET applications pose additional challenges for the underlying communication system. In this dissertation, previously proposed 802.11 MAC enhancements are examined along with their ability to successfully meet the special requirements of each VANET safety-critical application. It is determined whether or not enhancements to 802.11 MAC fit into and provide QoS for VSCAs.

2.5 Existing Enhancements on IEEE 802.11 MAC

Due to the aforementioned deficiency of IEEE 802.11 MAC for VSCAs, enhancements are needed for deployment of IEEE 802.11 based VANET. In this section, existing 802.11 MAC enhancements are first researched, and then categorized. An analysis of suitability

for VSCAs follows.

2.5.1 Repetition

A conventional flooding technique is widely considered the oldest and simplest repetition scheme. This, along with its severe impact on the network performance, is well-known. Clearly, this type of arbitrary technique is not suitable for VSCAs.

An extension layer between the logical link control layer and the standard MAC layer is proposed to control repetitions in [11, 47, 15]. Packet lifetime is divided into even slots, and the packet is repetitively passed over to the MAC layer on each slot. Packet reception rate is studied in [11, 15], however a key performance index - delay is not addressed.

The ECHO repetition scheme is a controlled flooding technique proposed in [14], although detailed performance indices, such as throughput and delay, are not provided [14].

Repetition schemes may statistically improve the reception rate of broadcast packets, however, this is not guaranteed during a relatively short period of time. This is especially true when a burst of interference-induced bit error occurs.

Repetition has the potential to burden the channel and degrade its performance, i.e. sending multiple copies of the same packet increases data traffic and decreases effective throughput.

Because repetition schemes reside upon the generic 802.11 MAC, none of the aforementioned reasons that account for the low performance are addressed. Likewise, reasons for the low performance still persist.

Simply repeating a broadcast packet is not efficient for ad hoc networks.

2.5.2 Reception confirmation mechanisms

Ideally, if each receiver confirms the consequence (success or failure) of reception to the sender, failed trasmissons might be retransmitted by the sender. Without consideration for other performance indices, highly satisfying PDR can be achieved. The author believes the reception confirmation mechanism is necessary for high performance in VSCAs.

Unicast in 802.11 DCF utilizes the straightforward positive acknowledgement (ACK) for successful transmission confirmation. If the unicast sender does not receive an ACK in a predetermined period of time, the sender deems the transmission a failure and

engages in re-transmittal. It should be noted that the broadcast based on 802.11 DCF does not support the acknowledgement mechanism.

Extensive efforts have been taken to apply a similar ACK/NAK mechanism in 802.11 DCF unicast in broadcast [48, 49, 50, 51, 52, 5, 53, 14].

In [48], a random access MAC protocol that supports broadcast, namely BSMA, is proposed. Its operations are outlined as follows [48]:

1. Collision avoidance phase.
2. Source sends RTS to all neighbors and sets the timer to WAIT-FOR-CTS.
3. Upon receiving RTS (and when not in YIELD state), neighbors send and set the timer to WAIT-FORDATA.
4. If the source receives CTS, sends DATA and sets the timer to WAITFORNAK. Else, if no CTS and WAIT-FOR-CTS timer expires, back off and go to step 1. Nodes that are not involved in the broadcast exchange, upon receiving CTS, set their state to YIELD and set their timer long enough to allow for the broadcast exchange to complete.
5. Neighbors send NAK if the WAIT-FOR-DATA timer expires and DATA has not been received.
6. If the source receives NAK before the WAIT-FOR-NAK timer expires, back off and go to step 1. Else, if no NAK and WAIT-FOR-NAK timer expires, the broadcast is complete. Go to step 1 and get ready to transmit new DATA.

According to BSMA's mechanism, three cases might exist to deteriorate its performance:

1. Even when only one neighbor replies CTS (other neighbors did not hear the RTS due to collision or interference), the source still proceeds to the next step, which cannot fulfill the high reliability requirements of VSCAs.
2. The source must be equipped with very good capture capability in order to pick up one good CTS when multiple CTS packets coexist on the channel. However, according to the ad hoc nature, the neighbors who reply CTS simultaneously might all be located close enough to the source, which invalidates the source's capture capability [54].

3. If NAK is lost or corrupted, a transmission is still deemed as a good one.

Similar to [49], proposals in [50, 51, 52, 5, 53, 14] try to collect acknowledgements from the neighbors in variant ways and to take action according to the outcome of the acknowledge collection. Notwithstanding all of them emphasize the PDR increase after their algorithms are applied, none of them consider the impact of their new algorithms on the key VSCAs delay performance and the overall network performance, such as throughput.

The reception confirmation can also be achieved by using a receiver-initiated scheme [55, 56, 57, 58]. The most prominent issue of applying a receiver-initiated scheme for VSCAs is to identify the initiator among many receivers to guarantee the delivery to other receivers other than the initiator. As a result, receiver-initiated scheme might not be efficiently applied either to broadcast or to VANET.

It must be noted that ACK is definitely needed to guarantee delivery in VSCAs. The key is how to ACK the sender in an efficient and effective way with minimum overhead (delay, bandwidth, implementation complexity and cost). The ACK process must be accomplished within the required latency.

2.5.3 Adaptive backoff

The backoff mechanism directly affects the collision sequel. If the size of the contention window or the number of backoff slots can be adjusted in real-time for VANET, the gain on the performance is clear.

Thus intensive research has been conducted to optimize the function of the 802.11 backoff mechanism in order to reduce collision as much as possible. There are two categories of backoff optimization.

2.5.3.1 Dynamically tuning contention window size

In the 802.11 standard [46], it is stated that the number of backoff slots is chosen uniformly from [0, $CW - 1$], where CW is the current size of contention window. It is evident that the channel resource is wasted when the number of terminals N in the network is much less than CW and that the access collision is high when $N >> CW$. A statistically ideal performance is expected to be achieved when $N \approx CW$. Since the 802.11 MAC is based on a binary exponential backoff (BEB) whose granularity of variation is too coarse, finer tuning mechanisms [59, 60, 61, 62, 63, 64, 65] have

been proposed in order to resolve the collision by estimating the number of contending terminals. However in the real world, it is not that easy to accurately estimate the number of contending terminals. For example, in [64] a symptotic condition is assumed, which might not be valid in a real network. For the highly dynamic VANET, the accurate estimation is even harder.

A fast collision resolution (FCR) algorithm is proposed in [66]. According to FCR, some terminals are picked out from all the competing nodes to transmit successfully or to collide, which gives these terminals probabilistic priority to send or contend first. Because the minimum contention window is 3, which is very small, and because FCR depends on the capability to sense the occurrence of collisions, it does not work in broadcast mode. Therefore, it is not directly applicable for VSCAs.

[67] adjusts the size of the contention window by overhearing the sequence number of received packets and accordingly evaluating the network congestion. However, it might not work in the highly dynamic VANET because the evaluation and adjustment are based on post-processing of information.

2.5.3.2 Cooperatively selecting the number of backoff slots

The early backoff announcement (EBA) algorithm proposed in [68] announces the value of next backoff timer of a certain terminal in its previous packet. By overhearing the channel, other terminals are supposed to be able to know the terminal's tentative choice of the number of backoff slots and avoid choosing that value in its own next backoff. As a result, collisions are expected to be reduced.

After entering a network, terminals with EBA must first spend some time on learning which values have been used by others. Then, they are able to cooperatively pick the number of backoff slots. This might not work in VANET because vehicles join and leave a network at high velocity. Not enough time can be guaranteed for them to learn the usage of backoff slots.

EBA also implicitly requires each terminal to reach a saturation condition, which is not realistic. Otherwise, the backoff usage information can easily become stale and channel slots are wasted.

Furthermore, the prerequisite of EBA is a reliable share of contention window usage. If collision or a fading channel causes the packet carrying EBA information to become corrupted, the cooperation of backoff slot usage among terminals is broken. This can occur easily in IEEE 802.11 broadcast, thus in 802.11 MAC based VANET as well.

2.5.4 Priority

IEEE 802.11e enhanced DCF (EDCF) [69] prioritizes different applications in order to provide QoS to delay-sensitive applications such as Voice over IP. In a fixed way, it assigns different backoff ranges and different inter-frame spaces to different applications. With a shorter backoff range and inter-frame space, those delay-sensitive applications are supposed to receive higher QoS.

Other priority-based enhancements have also been proposed [70, 71, 72, 73]. The common principle existing in them is similar to IEEE 802.11e: assign different priority with different parameters.

However, all of them face the same issue that the probability of collision might be still high if the number of terminals which have equal-priority packets is large. If 802.11e or its counterparts are applied in VANET, it might encounter serious performance degradation when the vehicle density is high.

2.5.5 Location aided MAC

Location aware multicast MAC (LAMM) proposed in [52] states that if the total coverage area $A(S')$ of a subset S' of all the receivers S satisfies

$$A(S') \geq A(S) \tag{2.3}$$

where $A(S)$ is the total coverage area of all the receivers and if the ACKs from S' are successfully received by the sender, the sender is able to conclude that all receivers S have received the broadcast packet. However, the statement is based on the assumption that the transmission error is caused primarily by packet collision, which might not be advisable in VANET, because the bit error rate is high in VANET [17, 9, 18].

At the same time, the location sensitivity of VANET applications does remind that it might be possible to use location information to evaluate the significance of a neighbor to the current broadcast information. If the significance is high enough (for example, higher than a given threshold), multicast with ACK is used to deliver the packet to that neighbor. Algorithms to evaluate the significance are critical and depend on multiple criteria such as distance to the sender, relative speed, trajectory tendency, etc. However, it is still an open issue to update location information in a real-time manner.

2.5.6 Broadcast RTS/CTS

The key to prevent hidden terminal problem from happening to a receiver is to silence all the receiver's neighbors when it is receiving.

As shown in Fig. 1.3, because of the existence of multiple receivers for broadcast transmission, the potential hidden terminal area in broadcast is greatly enlarged, which in turn aggravates the access collision and interference. It is really difficult to effectively achieve that all the potential hidden terminals are silent every time the source broadcasts.

However, [74, 48] directly applies unicast RTS/CTS into broadcast. This may make the source believe the transmission is going well while many receivers are suffering from collision or interference due to hidden terminal problem.

The broadcast medium window (BMW) in [49] breaks down a broadcast into multiple unicasts. Apparently, this is not a feasible and practical way for VSCAs broadcast because too much overhead and delay are introduced. Although the batch mode multicast MAC (BMMM) in [52] optimizes the BMW by sending RTS/CTS and RAK/ACK pairs in a batch mode, the overhead and the delay incurred are still proportional to the number of receivers.

In the leader based protocol (LBP) in [75], a leader is selected to receive RTS and reply CTS on behalf of all the receivers. Since the leader cannot represent all receivers, an acceptably reliable transmission cannot be guaranteed even though single CTS scheme avoids the collision caused by simultaneously multiple CTS.

2.5.7 Transmission power control (TPC) and adaptive rate

If a TPC mechanism (different nodes transmit at a different and adaptive power level) were implemented realistically, it would have three folds of advantages

- Less collision/interference
- Higher spatial reuse
- Less energy consumed

For wireless transceivers on a vehicle, technically energy saving is not an issue although environmentally it is.

It is very straightforward that the former two advantages indeed can help increase packet delivery ratio.

The proposed TPC schemes either suffer from practicality issues or are throughput oriented [76]. From the author's point of view, more studies on whether TPC is a practically viable option for VANET is worthwhile.

All the proposed TPC schemes focus on unicast services. In VSCAs, multiple receivers exist for one broadcast frame. Hence a sender needs to know the channel conditions between the sender and each one of the receivers are in order to effectively implement TPC. This adds more challenges than it has in unicast.

A node dynamically varies its modulation and error-coding schemes to fight against time-varying channel, which is called adaptive rate. It is theoretically and practically reasonable for unicast service [76]. However, it faces the same challenges that TPC has in VSCAs. In order to adopt a rate, the sender needs first to know the channel situations between it and each of the receivers.

2.5.8 Out-of-band signaling (OBS)

Because control packets and data packets can collide with one another if they go on the same channel, schemes of OBS to shunt different traffic into different channels to reduce packet collisions have been proposed [76, 77]. OBS can be either in the way of a busy tone or in the way of control packets, i.e. sending control packets such as RTS/CTS in a separate channel.

Researchers in [76] use the busy tones to alleviate hidden terminal problems or exposed terminal problems in unicast services. Researchers in [77] use busy tone technique to alleviate channel access collision. It is good for a one-hop network to utilize the channel to its maximum performance, but might not be good for VANET where hidden terminals might desynchronize signaling and deteriorate severely the accuracy of the signaling. Furthermore, since it is a throughput-oriented enhancement, it might not be directly applied in VANET for VSCAs. Researchers in [53] use multicast RTS (MRTS) and busy tones to alleviate the hidden terminal problem and to acknowledge the reception of the multicast packet. Since the sender must include the information of all the receivers in the MRTS, a reliable broadcast cannot be guaranteed. MRTS still suffers from packet collision and channel fading, given that adding MRTS with receiver information makes the packet length long. Researchers in [5] use a negative CTS tone to replace CTS and use a negative ACK to replace ACK. A serious problem exists in this mechanism. If RTS collides, which is common when the number of nodes is high and when hidden terminals exist, the receivers are unable to recognize whether it is a collision of multiple RTS (it might be just noise) and they are not able to send out the

NCTS tone.

The most common attempt of OBS control packets is to send RTS/CTS in a channel different from that of data packets [76]. This indeed removes control packets from competing in the same channels with data packets, which provide more bandwidth for data at the first glance. And based on the success exchange of control packets on the separate control channel, the multiple access on the data channel is expected to be resolved. However, control packets such as RTS/CTS themselves are packets. They also suffer from fading and collision among themselves when the traffic load is high.

The busy tone scheme provides simple solution to multiple access problems and have the advantages of small bandwidth and simple coding which simplifies the transceiver design, it is thus worthwhile investigating it to realize its viability and suitability in VANET.

CHAPTER 3

Broadcasting in VANET

An overview of DSRC MAC and R-ALOHA and their details of broadcasting are presented in this chapter.

ALOHA is the origin of the contention-based MAC mechanism, which can be summarized as "Send anytime. Resend if failed." Both CSMA and R-ALOHA use ALOHA's basic contention mechanism. The difference is that some private new features were added to ALOHA to improve the basic mechanism to accommodate specific situations.

3.1 Overview of DSRC MAC

DSRC MAC uses carrier sense multiple access with collision avoidance (CSMA/CA). CSMA/CA is a random-access, contention-based protocol for which the basic idea first appeared in ALOHA. It is well known that ALOHA has a very low efficiency due to the high probability of collision. DSRC MAC improves the performance of contention-based protocol by integrating the following mechanisms to provide feasible, multiple access among all stations in a distributed manner on a single wireless medium.

3.1.1 RTS/CTS

RTS/CTS stands for Request To Send/Clear To Send. RTS/CTS mechanism works as follows:

Before a data packet is ready to be sent, the sender always sends an RTS packet to the receiver and then waits for the receiver to send back CTS.

When a receiver receives the RTS packet, it promptly replies with a CTS packet.

After receiving the CTS packet, the sender sends the data packet.

Transmission of RTS/CTS between sender and receiver lets all neighbors of the sender and the receiver know the data is coming so that the two most notorious problems-hidden terminal problems and exposed terminal problems-are eliminated.

3.1.2 Physical and virtual carrier-sense

Due to the nature of a wireless medium, it is impossible for terminals to detect packet collision. Therefore, physical carrier-sense and virtual carrier-sense of DSRC MAC only determine the channel status: busy or idle. If a terminal senses the channel busy by either physical carrier-sense or virtual carrier-sense, it refrains from sending data. This decreases the number of packet collisions greatly.

The physical layer implements the physical carrier-sense and conveys the result to the MAC layer.

Virtual carrier-sense exploits the so-called network allocation vector (NAV) to fulfill the sensing function. "The NAV maintains a prediction of future traffic on the medium based on duration information that is announced in RTS/CTS frames prior to the actual exchange of data. The duration information is also available in the MAC headers of all frames sent during the CP other than PS-Poll Control frames." [69]

3.1.3 Inter-Frame Spacing (IFS)

IFS is the time interval between frames. Terminals use carrier-sense function to determine a specific IFS. There are four types of IFS in DSRC MAC:

- Short inter-frame space (SIFS)
- Point Coordination Function (PCF) inter-frame space (PIFS)
- Distributed Coordination Function (DCF) inter-frame space (DIFS)
- Extended inter-frame space (EIFS)

For the details of each IFS, please refer to [69]. In this thesis, only DIFS and EIFS are discussed. Fig. 3.1 is the illustration of IFS. IFS function includes four aspects:

- Provide the transceiver sufficient time to switch transmission/receiving functions
- Provide fair access to medium among all terminals
- Provide access priority
- Alleviate channel failure caused by collision or bit error

Each physical layer has its own fixed IFS. DSRC uses orthogonal frequency division multiplexing (OFDM) as its physical layer and its related IFSs are defined in [10][69].

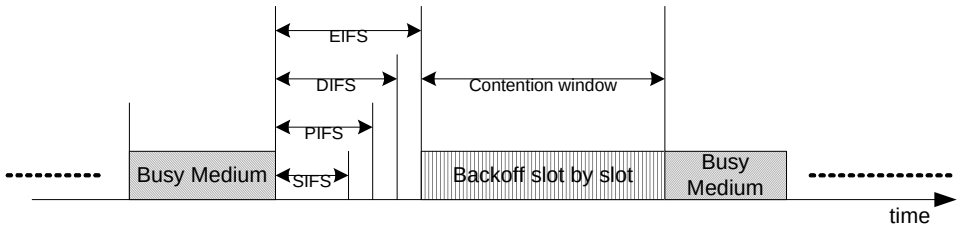

Figure 3.1: Inter-Frame Spacing

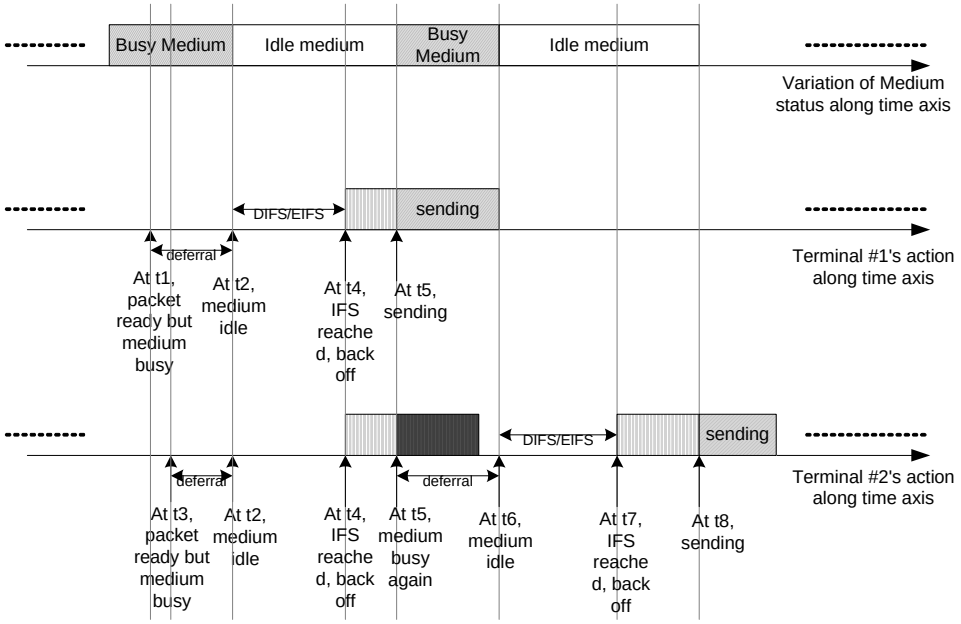

Figure 3.2: An example of backoff mechanism

3.1.4 Random backoff

As mentioned previously, one terminal executes a carrier-sense to test the medium before initiating a data transfer. If the medium is busy, the terminal defers transmission until the medium is continuously idle for a period of time equal to DIFS/EIFS. After this DIFS/EIFS idle time, if the backoff timer of the terminal contains zero, the terminal generates randomly a non-negative integer equal to the number of backoff slots, which is uniformly distributed on [0, w-1], where w is the size of the contention window. Backoff then occurs. During backoff, terminals continue sensing the medium, and terminals must freeze the backoff procedure when the medium becomes busy again before the backoff timer expires.

The random backoff mechanism scatters the terminals that have been deferred due to the same event in time domain and reduces collisions. Fig. 3.2 shows an example of the principle of backoff.

3.1.5 Positive acknowledge and retransmission

Air links are not only expensive but unreliable because of multi-path fading and interference. The error-prone wireless medium increases the vulnerability of the data being carried on it. The DSRC MAC specification [10] requires that each unicast data packet transmission must be positively acknowledged by the receiver. If the sender does not receive the positive acknowledgement in a specified period, the packet must be scheduled for retransmission. Positive acknowledgement and retransmission improve the reliability of the communication; however, they also increase service overhead.

3.1.6 Fragmentation and defragmentation

It has been mentioned that the wireless medium is error prone. Based on the bit error rate of a channel and the error-correction mechanism that can fix a limited number of bit errors, a packet size range exists for achieving the highest probability of receiving a packet successfully. Beyond the size range, i.e., the length of one packet is longer than the packet size for optimum packet reception, the probability drops tremendously. The fragmentation process cuts the long packet into shorter fragments at the transmitter side and defragmentation reassemblies those fragments back to the original long packet on the reception side. Therefore, fragmentation and defragmentation plus the retransmission mechanism, can effectively improve the successful reception of large-sized packets.

3.2 DSRC MAC Broadcast

The broadcast procedure of 802.11 MAC follows the basic medium access protocol of distributed coordination function (DCF) with three functions disabled. RTS/CTS is not used in broadcast. "The RTS/CTS mechanism cannot be used for MPDUs with broadcast and multicast immediate address because there are multiple destinations for the RTS, and thus potentially multiple concurrent senders of the CTS in response." [69]

No positive acknowledgement and retransmission exist. "There is no MAC-level recovery on broadcast or multicast frames, except for those frames sent with the ToDS bit set. As a result, the reliability of this traffic is reduced, relative to the reliability of directed traffic, due to the increased probability of lost frames from interference, collisions, or time-varying channel properties." [69]

The fragmentation operation is inhibited for broadcast packets. "Only MPDUs with a unicast receiver address shall be fragmented. Broadcast/multicast frames shall not be

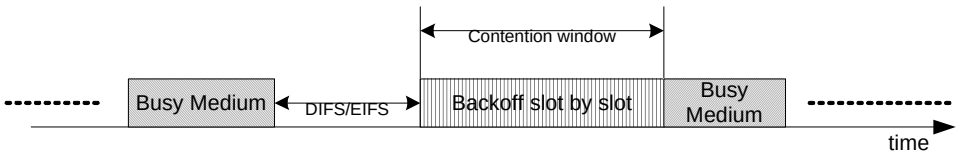

Figure 3.3: DSRC MAC broadcast

fragmented even if their length exceeds $aFragmentationThreshold$." [69]

Broadcast of DSRC MAC occurs when a broadcast packet arrives at DSRC MAC layer from upper layer and the MAC senses the channel status first and stores the status. Next, once an idle period equal to DIFS/EIFS is observed, MAC takes the next operation according to the stored channel status and the current value of its backoff time.

If the current value of the backoff counter is not zero, MAC begins the backoff countdown process. If the current value of the backoff counter is zero and the status of the medium is busy, MAC generates random backoff time and begins the backoff countdown process. If the current value of the backoff counter is zero and the status of the medium is idle, MAC begins data packet transmission immediately.

During the backoff countdown process, carrier-sense persists. If the medium becomes busy again, MAC goes back to the DIFS/EIFS observation process. If the backoff timer expires, MAC begins data packet transmission right away.

During or after transmission, MAC does not monitor the success or failure of the transmission. Once the transmission is completed, MAC simply releases the medium and competes for it when a new packet is ready to be sent. Fig. 3.3 illustrates the broadcast of DSRC MAC.

To fully understand DSRC MAC, two concepts must be emphasized.

First, if one terminal has an immediate packet in its queue upon completion of the transmission of the previous packet, the current queued packet cannot be transmitted until a random backoff is executed. This means that there must be a random backoff between two consecutive packets. "When an MSDU has been successfully delivered or all retransmission attempts have been exhausted, and the STA has a subsequent MSDU to transmit, then that STA shall perform a backoff procedure."

Second, after a broadcast transmission, a DIFS or an EIFS must be observed before backoff or the next transmission. DIFS is observed in most cases. EIFS is used by a terminal when it has received an erroneous packet. But once a good packet is received later, that terminal switches to use normal DIFS again.

Table 3.1: Limitation of DSRC MAC Broadcast

Limitations	**Consequences**	**Reasons**
RTS/CTS forbidden	Collision increased	Hidden terminal problems Broadcast storm
No positive ACK	Reliability decreased Collision increased	Unknown tx result Fixed contention window Broadcast storm
No fragmentation	Collision increased	Long packets Fixed contention window High bit error

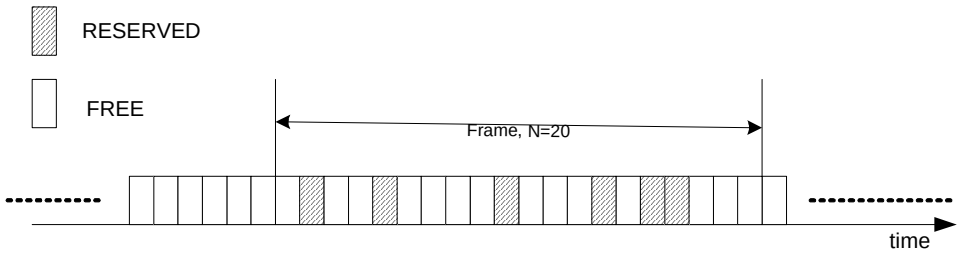

Figure 3.4: R-ALOHA frame and slot

3.3 Limitations of DSRC MAC Broadcast

As previously stated, 802.11 MAC has three limitations that might degrade its service quality in IVC. These three limitations are summarized in TABLE 3.1.

3.4 Broadcast in R-ALOHA

3.4.1 Overview of R-ALOHA

Based on ALOHA, R-ALOHA [78] includes the following differences:

Time is divided into frames. Each frame consists of N slots.

A slot must be reserved by the terminal before it starts the transmission.

Fig. 3.4 shows R-ALOHA.

One slot is marked as RESERVED when it is used by a terminal. Otherwise, it is

marked as FREE. Any terminal without a reserved slot will have to reserve a FREE slot before the terminal is allowed to transmit. If the reservation is recognized as successful, the slot is reserved for the reserving terminal for the subsequent frame. No other terminals can access that slot until it is released. If the reservation fails, the terminal has to contend for this or another slot in order to transmit a packet.

R-ALOHA is further divided into two ways of observing slot usage:

Slot-by-slot R-ALOHA

Frame-by-frame R-ALOHA

Slot-by-slot. In slot-by-slot R-ALOHA, terminals exchange with one another the observations of slot usage and decide the reservation of each slot in a slot-by-slot manner. A ready terminal contends for an empty slot with probability p, called contention probability. The terminal generates a random number x on $(0, 1)$. If p is greater than or equal to x, this terminal will contend for this empty slot, otherwise it will postpone its contention until next time when p is greater than or equal to x. If there is only one terminal contending for the empty slot, the terminal's reservation for the slot is successful. If there is more than one terminal contending for the same empty slot, collision occurs; and no terminal reserves that slot. The colliding terminals have to contend later. It is easy to see that the value of the contention probability is very important for the success rate of slot reservation. If the value of the contention probability is too high, the probability of collision of more than one terminal contending for the same slot is also very high. On the other hand, if the value of contention probability is too low, slots are wasted. In the simulation performed for the research on which this paper is based, as will be addressed in a subsequent chapter, three values of 0.2/0.1/optimized were used. The optimized value results in the best theoretical value of contention probability. It is equal to the reciprocal of the number of terminals currently contending for a slot. Although some special techniques are required to implement this optimized value mechanism in an ad hoc network, the author used this optimized value directly in the simulation to compare performance difference with various contention probabilities.

Frame-by-frame. In frame-by-frame R-ALOHA, all exchanges of slot usage and slot reservation are conducted at the beginning of each frame. Each ready terminal chooses one empty slot from all the available slots. If one empty slot is chosen by only one terminal, that terminal successfully reserves the slot. However, if two or more terminals choose the same empty slot, collision occurs; and no terminal reserves the slot. During a frame, no signaling messages are exchanged, and only data packets are transmitted by the terminals that have successfully reserved slots at the beginning of the frame.

R-ALOHA in an infrastructure network requires a central coordinator that collects, decides, and distributes the slot usage. This means that every terminal is under the control of the coordinator, and the reservation can be achieved. The packet reservation multiple access (PRMA) protocol is such a centralized R-ALOHA protocol [79].

However, in IVC, such a coordinator does not exist. R-ALOHA in IVC must be implemented in a distributed way. Although some MAC designs were proposed to realize distributed R-ALOHA, they did not support a reliable broadcast service. Examples of these MAC designs includes distributed packet reservation multiple access (D-PRMA) protocol, hop reservation multiple access (HRMA) protocol, five-phase reservation protocol (FPRP), etc [79]. The next section of this paper describes the details of R-ALOHA broadcast and the reason those distributed R-ALOHA designs are not suitable for reliable R-ALOHA broadcast service.

3.4.2 R-ALOHA Broadcast

It is key for R-ALOHA implementation (either centralized or distributed) that a mechanism exists that guarantees that reservation information and competition results are communicated effectively. For centralized R-ALOHA or point-to-point R-ALOHA, the previously mentioned protocols have been implemented and proved to be effective [79]. For broadcast R-ALOHA however, the situation is different making it hard to implement.

The aim of broadcast is to deliver the broadcast packet to everyone inside the network. Therefore, more than one receiver might exist. This is different from the point-to-point case depicted by Fig. 3.5(a) in which each packet is destined for one receiver. Hence, in the point-to-point case, each slot is divided into several segments to carry signaling messages (Ex. REQ and ACK), and a slot reservation can be determined to be successful if the signaling message exchange succeeds on that slot between the source and the destination terminal or determined to have failed because the signaling messages collided with other contending messages or because of interference. Multiple receivers prevent this mechanism from being applied in the broadcast case depicted by Fig. 3.5(b) because the channel will be in chaos if all the receivers reply at the same time.

3.4.3 Limitations of R-ALOHA Broadcast

High mobility has the greatest impact on the reservation protocol. Vehicles move at high and different speeds and in different directions. A vehicle might enter and leave networks very frequently. This might cause the following problem:

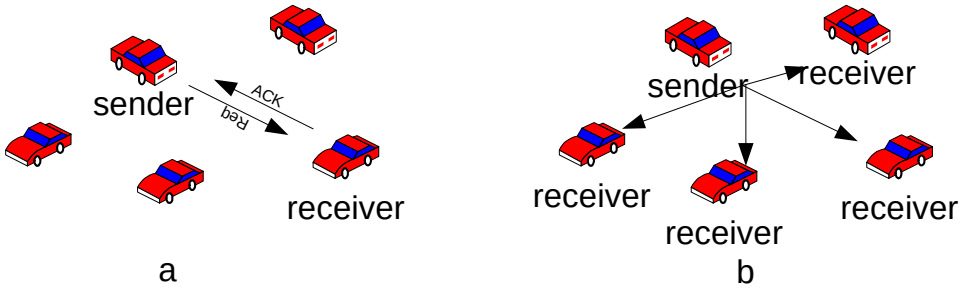

Figure 3.5: (a) R-ALOHA point-to-point (b) R-ALOHA broadcast

The very frequent request and release of a slot when a terminal enters and leaves a network. For R-ALOHA, a slot is reserved and released for one packet.

More collisions happen when a terminal enters a new network bearing its previous reserved slot.

3.5 Simulation

The simulation is conducted in a two-dimensional space. The whole space is divided into small squares. Each square is the same size, 64 square meters (eight by eight) according to the size of the distance between vehicles. Initially, each terminal is randomly positioned on a corner of a square, and only one terminal is put on one corner. Fig. 3.6 shows the terminal position arrangement. The circles represent the radio coverage of the transceivers.

3.5.1 Simulation Setting

For the sake of simplicity, the simulation was conducted in a one-hop broadcast network, which means the hidden terminal problem does not exist and the signal propagation simulation was disabled. The mobility module was also disabled. But the mobility factor was considered by introducing the signal-to-noise ratio at a certain vehicle velocity [9].

Global parameters influencing the whole simulation platform and their values used in the simulation are listed in TABLE 3.2.

TABLE 3.3 lists the parameters used in DSRC MAC simulation.

TABLE 3.4 lists the parameters used in R-ALOHA simulation.

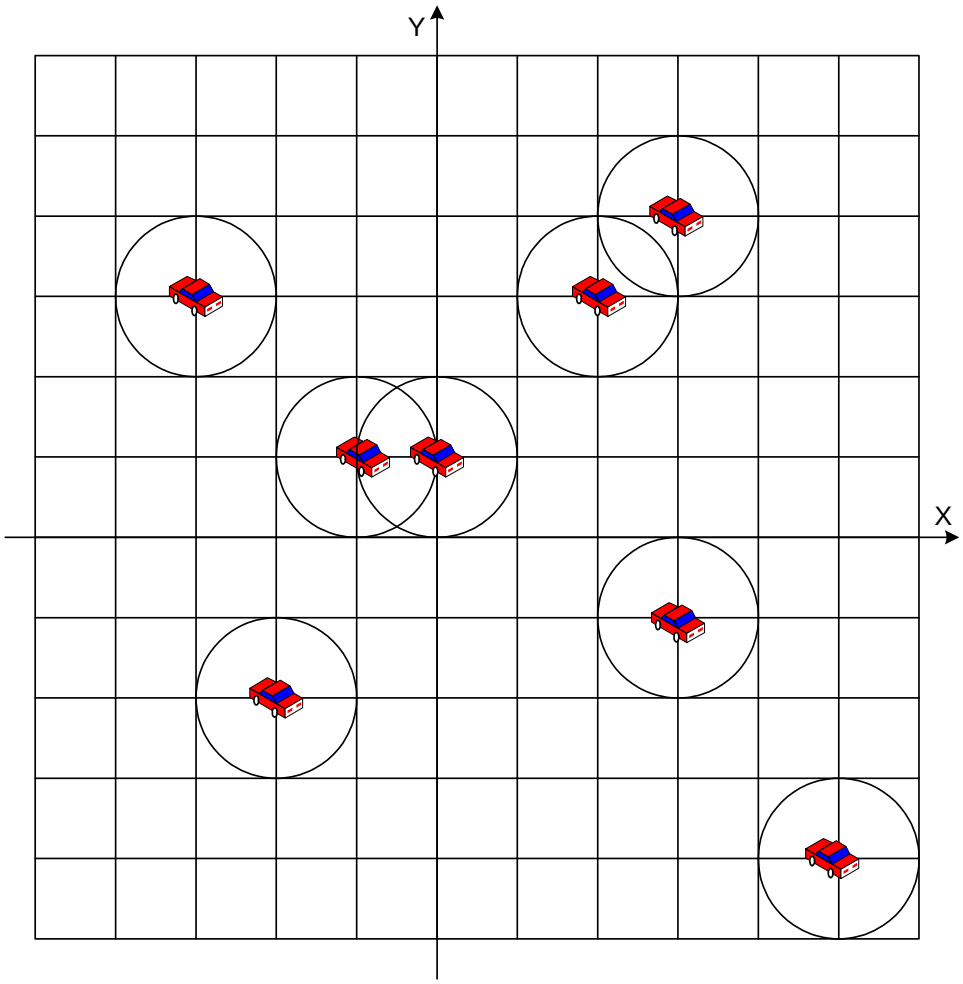

Figure 3.6: Terminal position arrangement

Table 3.2: Global Parameters

Parameter Name	**Value**
Simulation duration	$100s$
Data rate	$12Mbps$
Poisson packet interval	$0.01s$
Number of terminal	5 205
Terminal average velocity	$85mph$
Signal to noise ratio [9]	$13dB$
Bit Error Rate	10^{-3}
Frame error correction bits	$3bits$
Capture effect	off
Maximum MPDU length	$800bits$

Table 3.3: DSRC MAC parameters

Parameters	**Value**
Min contention window	31
slot size	$16us$
SIFS	$32us$
DIFS	$64us$

Table 3.4: R-ALOHA parameters

	slot per frame	**slot size**	**REQ probability**
Slot-by-slot	N/A	$80us$	0.2/0.1/optimized
Frame-by-frame	120	$80us$	N/A

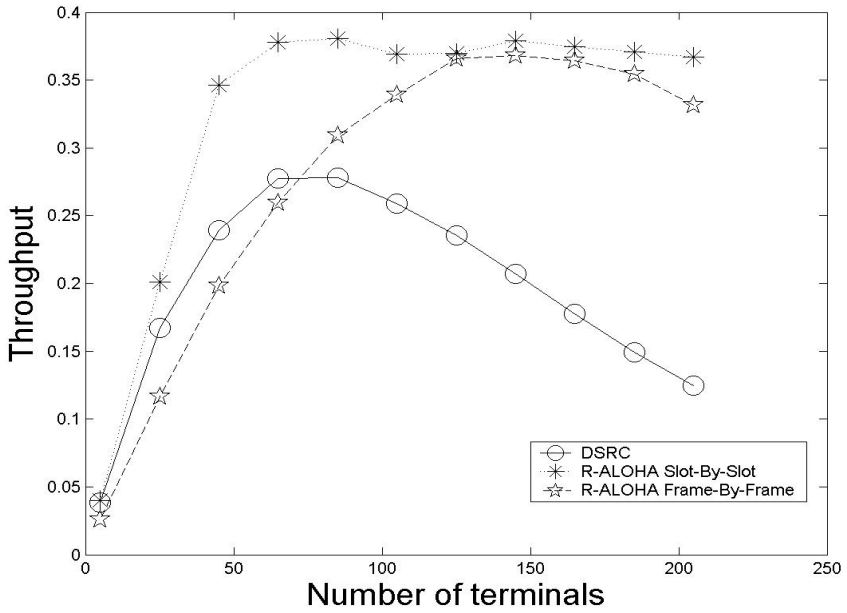

Figure 3.7: Normalized throughput

3.6 Results and Analysis

3.6.1 Normalized Throughput

Normalized throughput S is defined as the ratio of two periods of time duration T_r and T_t.

$$S = \frac{T_r}{T_t} \tag{3.1}$$

T_r is defined as the time duration when only one of the nodes transmits (no collision) and the packet transmitted by the node is received by at least one receiver.

T_t is the total time duration of the simulation.

Fig. 3.7 shows the normalized throughput performance of each protocol, all three of which is fairly low (under 40%) under broadcasting circumstances. The main reason for the low throughput is that each terminal has to contend for a channel every time it has a packet to send because no packet fragmentation is allowed in broadcasting.

3.6.1.1 DSRC MAC Throughput

According to [69] "Only MPDUs with a unicast receiver address shall be fragmented. Broadcast/multicast frames shall not be fragmented even if their length exceeds $aFragmentationThr$ The broadcast message in the simulation is not fragmented by MAC layer, which means each node will have to compete for the channel through the DIFS mechanism every time it has a broadcast packet to send. The normalized throughput curve of DSRC MAC consists of three stages.

In the first stage, the normalized throughput increases from 5% to around 28% as the number of terminals increases from 5 to 65. The average range of traffic of this stage is 500 to 6500 packets per second, and the contention window size is fixed at 32. Under this circumstance, the random backoff number selection mechanism is still able to scatter each ready terminal (ready to send a packet) to a different time slot, which means the probability of collision (two or more terminals start transmission at the end of the same time slot) remains low. When the traffic is low, some of the time slots are not used while the collision rate is also low. The traffic accruement brought by the increase in the number of terminals statistically makes more slots be used. Meanwhile, collision increases. However, the increase of slot usage is higher than the increase of collision; therefore, the normalized throughput increases.

In the second stage, the normalized throughput remains at around 28% to 29% when the number of terminals is between 65 and 85. The slot usage and collision rate are equal in this stage; therefore, the normalized throughput peaks and remains at that level.

The third stage occurs when 85 terminals are in use. Throughput drops when more than 85 terminals are used. Having more than 85 terminals results in higher traffic, thus, there is a greater probability that a slot will be contended for by more than one terminal, resulting in collision. Collision wins over the slot usage.

The normalized throughput performance of DSRC MAC is the lowest of the three protocols studied. The low throughput exists because of the high collision probability brought by the basic access mechanism and the fixed contention window size. However, DSRC MAC has the advantage of simplicity.

3.6.1.2 Slot-by-slot R-ALOHA throughput

The normalized throughput curve of slot-by-slot R-ALOHA consists of two stages.

The first stage starts when the number of terminals equals five and ends when the number of terminal is 65. The normalized throughput increases linearly in this stage.

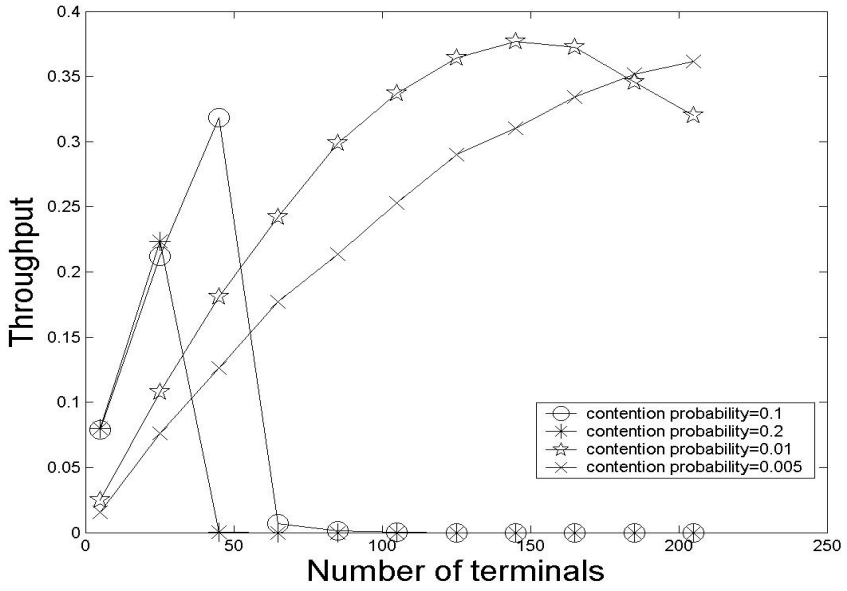

Figure 3.8: Normalized throughput of slot-by-slot R-ALOHA with different contention probability

The reason for the increase is that more traffic (caused by the increased number of terminals) makes slot usage increase.

The normalized throughput curves and then stabilizes and remains around 37% in the second stage. This stability of throughput is explained later in this dissertation.

Slot-by-Slot R-ALOHA shows the best throughput performance among the three protocols studied. However, the normalized throughput is achieved by using the optimized contention probability. Fig. 3.8 shows normalized throughput of slot-by-slot R-ALOHA with different contention probability.

If contention probability is not set as optimized, the throughput is poorer than optimized throughput. It is even unacceptable. When the contention probability is 0.2, the throughput drops suddenly to almost zero when the number of terminals reaches 25. When the contention probability is 0.1, the throughput drops suddenly also to almost zero when the number of terminals reaches 45. Although the severe situation of sudden drop disappears when 0.01 or 0.005 is chosen as the contention probability, the throughput cannot remain at a maximum level in the same way optimized contention probability does. Eventually, the throughput will drop if the optimized contention probability is not applied. Therefore, how to implement the optimized mechanism or how to maintain an acceptable contention probability in the IVC ad hoc network is a key problem for the development of slot-by-slot R-ALOHA.

Fig.. 3.7 shows slot-by-slot R-ALOHA throughput stability when the optimized contention probability is chosen. Recalling that the definition of optimized contention probability is the reciprocal of the number of terminals that currently are contending for the current slot.

3.6.1.3 Frame-by-frame R-ALOHA throughput

The normalized throughput performance curve of frame-by-frame R-ALOHA lies between the throughput of slot-by-slot R-ALOHA and DSRC MAC. It consists of 3 stages.

Stage I: Throughput increases as the number of terminals increases from 5 to around 120

Stage II: Throughput remains at about 39% when the number of terminals remains around 120

Stage III: Throughput starts to drop when the number of terminals exceeds 120

If the number of terminals is less than the frame size (each frame consists of 120 slots) there is a low probability that a collision will occur when two or more terminals choose the same slot. When the number of terminals is greater than 120, more and more terminals collide in the same slot. Thus, more and more slots are wasted because of the collision. The above statement determines the shape of the throughput performance curve of frame-by-frame R-ALOHA.

3.6.2 Average Delay

Average delay is defined as the average time lapse between the time a packet is generated and the time it is sent.

Fig. 3.9 shows the average delay of each protocol.

The latency requirement for IVC is 100ms [4] and delay performance of each protocol satisfies this requirement. The maximum delay is around 0.045 sec.

Fig. 3.10 shows delay of DSRC, which looks extremely low (less than 220us). This extremely low delay is a result of the mechanism used in 802.11 MAC.

In 802.11 MAC, there is no fragmentation and positive acknowledgement for broadcast/multicast. In this simulation, the contention window size is fixed to a minimum value for broadcasting. Therefore, the latency for each packet is equal to the sum of the time spent on waiting for DIFS and the backoff time with a maximum value is $15 * 16 = 240us$ and an average value of $8 * 16 = 120us$. Behind this low latency, how-

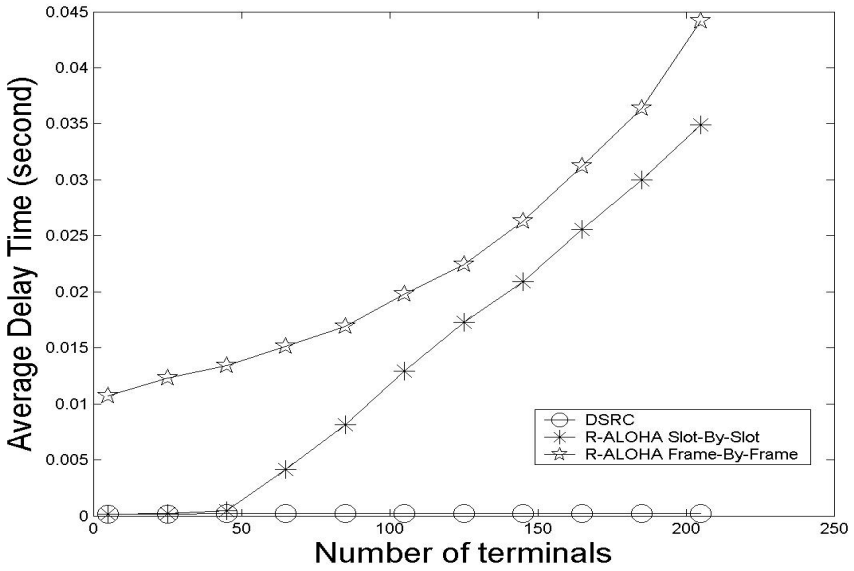

Figure 3.9: Average delay

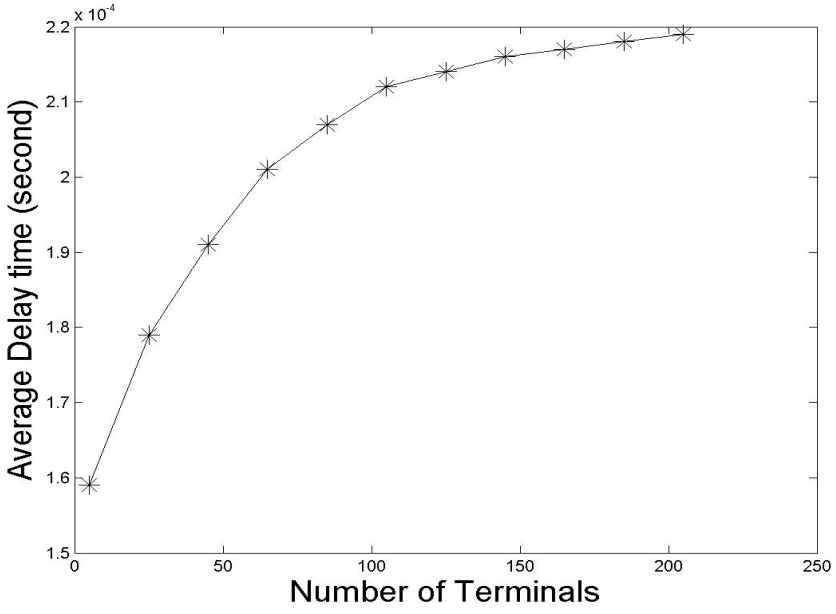

Figure 3.10: Delay of DSRC

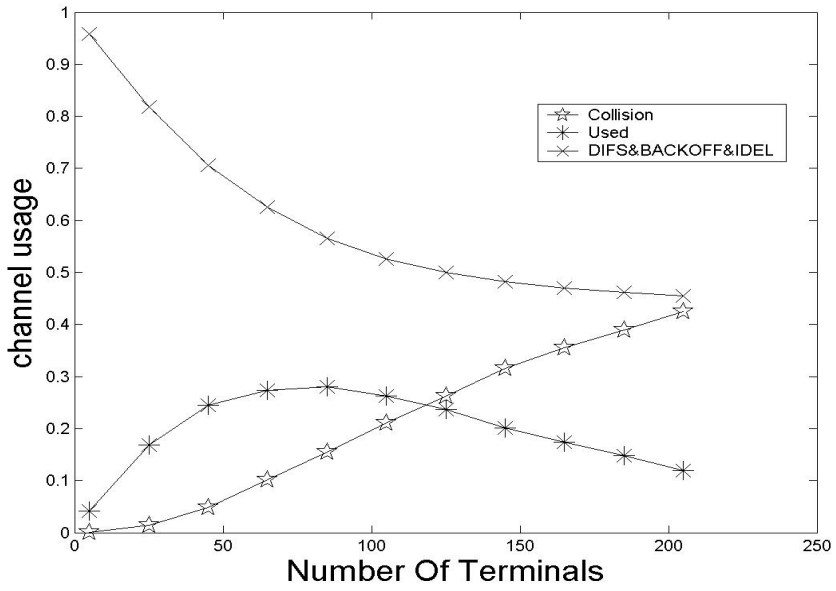

Figure 3.11: Channel usage of DSRC

ever, is the high probability of packet collision. In order to show this, Fig. 3.11 shows the plot of channel usage. It is clear that the collision and DIFS backing-off increase while the number of terminals increases and occupy almost 90% of the channel, which makes the throughput drop.

3.6.3 Packet Delivery Ratio

Packet delivery ratio is defined as the ratio of the number of packets successfully received to the number of packets expected to be received.

Fig. 3.12 shows the packet delivery ratio of each protocol.

Fig. 3.13 and Fig. 3.14 show the tendency of packet delivery ratio of R-ALOHA separately.

In R-ALOHA, the failure of packet delivery is solely caused by error bits. Although the ratio drops while traffic increases, it still remains at a very high level, which is over 97%.

However in DSRC, collision is the most important factor affecting packet delivery ratio. Fig. 3.11 illustrates the reason why the packet delivery ratio in DSRC is becoming so low as traffic increases.

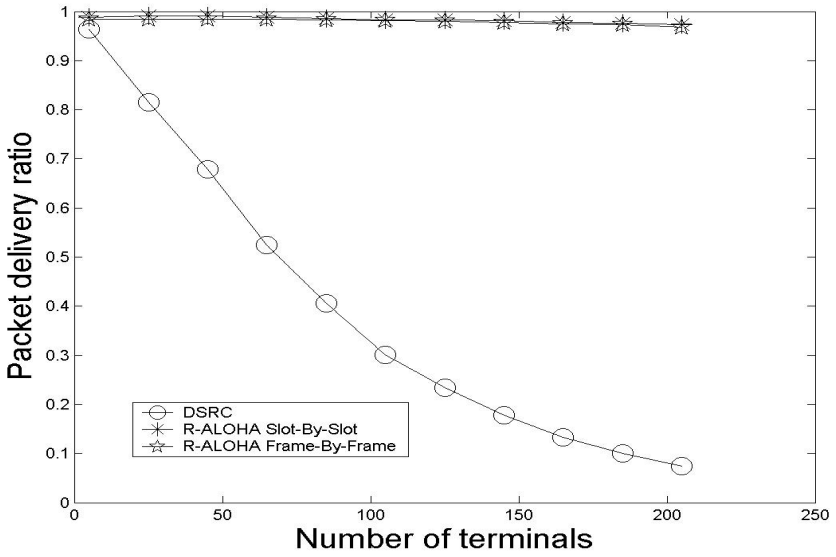

Figure 3.12: Packet Delivery Ratio

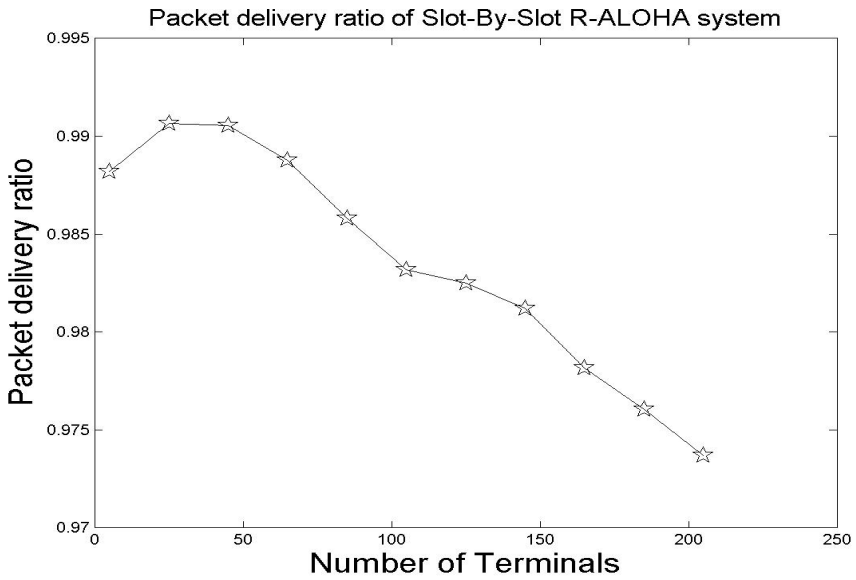

Figure 3.13: Packet Delivery Ratio of Slot-by-slot R-ALOHA

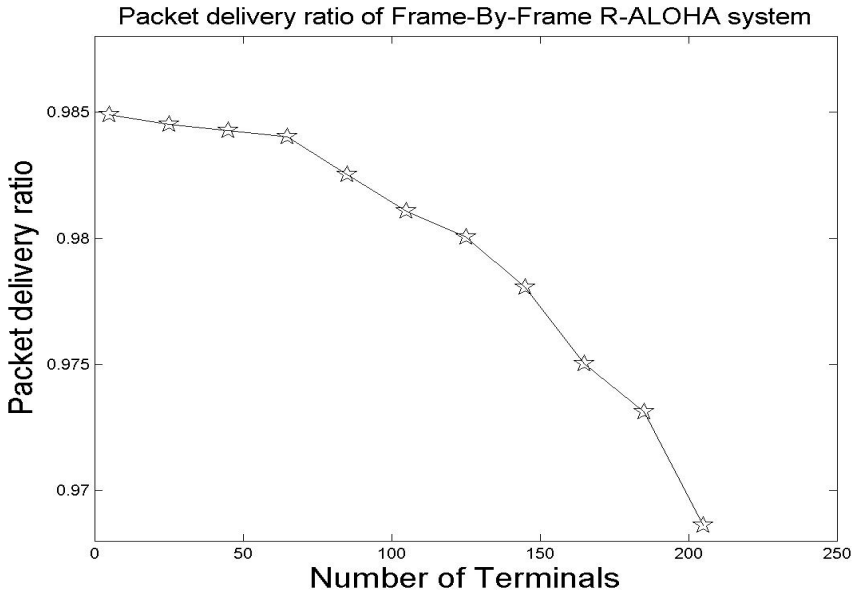

Figure 3.14: Packet Delivery Ratio of Frame-by-frame R-ALOHA

CHAPTER 4

Saturation Performance of IEEE 802.11 MAC

In this chapter, the performance of IEEE 802.11 broadcast investigated by developing an analytic model as well as by conducting simulation.

The saturation performance of the distributed coordination function (DCF) mode of the IEEE 802.11 MAC protocol has been extensively studied [59, 80, 81, 82]. However all of the studies focused on IEEE 802.11 unicast services. A phenomenon which is referred to as backoff counter *consecutive freeze process* (CFP) exists in IEEE 802.11 network. This phenomenon in unicast has been addressed and analyzed in [80, 82]. In unicast service, after a busy period, only the station which just had a successful transmission may consecutively access the channel if its newly chosen backoff time is zero [80]. However, in broadcast service where an acknowlegement (ACK) or retransmission mechanism is not available, all stations that just finished their transmissions may have a chance to consecutively access the channel even if a collision has just occurred. Therefore, CFP in broadcast happens more often than that in unicast service. This aggravation of CFP in broadcast causes the analytical models for unicast [59, 80, 81, 82] to not be directly applicable for the analysis of the broadcast service, which will be shown in the subsequent sections.

A simple but very accurate model is constructed to characterize backoff counter operation to evaluate performance of broadcast service in IEEE 802.11 including throughput, packet delay, and packet delivery ratio, under the same assumptions of saturation condition and ideal channel as in [59, 80, 82]. The close form expressions of these performance indices are derived and the results of the analytical model are validated by simulation.

4.1 How CFP Happens

In light of IEEE 802.11 standards [46], the following four facts are known.

1. the initial backoff time T_{ib} is generated by using

$$T_{ib} = Uniform(0, w-1) \times \sigma \tag{4.1}$$

   where w is the contention window size and σ is the time duration of one backoff slot;

2. transmission shall commence whenever the backoff time counter reaches zero;

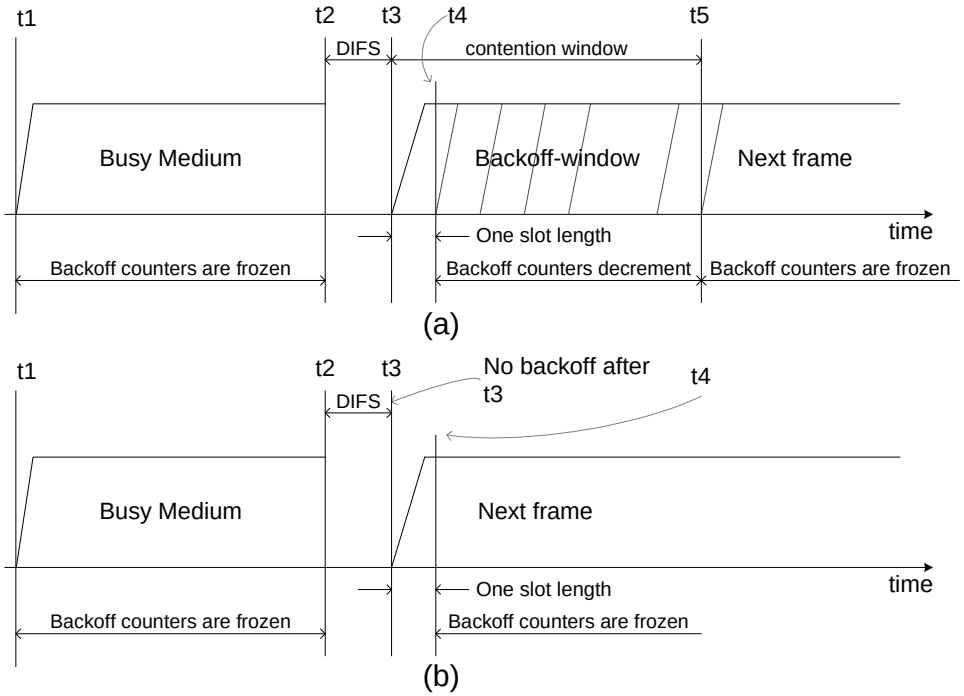

Figure 4.1: Example of backoff counter operation (a) backoff without consecutive freeze; (b) backoff with consecutive freeze

3. if no medium activity is detected for the duration of a particular backoff slot, then the backoff procedure shall decrement its backoff time counter by one slot;

4. if the medium is determined to be busy at any time during a backoff slot, then the backoff procedure is suspended; that is, the backoff time counter shall not decrement for that slot.

As a result of the above four facts, it is possible that a station that just completes a transmission and has a new packet to send chooses zero as its initial backoff time, and starts to transmit right after a DIFS [80, 82]. Such consecutive transmissions shall seize the channel continuously and leave other stations no chances to backoff, which is referred to as CFP. CFP is further illustrated in Fig. 4.1 and briefly explained as follows. Suppose that three stations (station1, station2, and station3) are in the network and the time duration between t3 and t4 is equal to the time duration of one backoff slot. From t1 to t2, station1's transmission makes the channel busy and freezes station2's and station3's backoff time counters. At this time, the value of either station2's or station3's backoff time counter is nonzero. Station1 finishes transmission at t2. Then, a DIFS is observed by all stations until t3. If station1 immediately has a second packet to send (this is definite under the saturation assumption), it randomly generates a new initial

backoff time at t3. If the value of the initial backoff time that station1 randomly chooses is nonzero, minimum value in the backoff counters of all three stations must be greater than or equal to one so that the channel is idle before t4. According to the above fact No. 3 each station will back off for at least one time slot before the channel becomes busy again at t5. This is shown in Fig. 4.1(a). However, as shown in Fig. 4.1(b), if station1 chooses zero as its new initial backoff time, according to fact No. 2 it starts the transmission right at t3. Before t4, the transmission of station1 is definitely sensed by station2 and station3 which are meanwhile doing the backoff procedure. Therefore according to fact No. 4, the backoff procedure of station2 and station3 is suspended, which makes the backoff time counters of station2 and station3 become consecutively frozen for at least two transmission periods.

4.2 CFP Differences in Between Unicast and Broadcast

Although it is the same aforementioned mechanism that causes CFP in both IEEE 802.11 unicast and broadcast, its scale and impact are significantly different in the unicast from in the broadcast.

According to IEEE 802.11 [46], a station randomly selects its backoff time when either a successful transmission or a transmission failure due to collision or channel interference has just occurred. A zero value is probable according to (4.1).

In the unicast, CFP may happen only after a station just successfully transmitted a packet. Acknowlegement and request-to-send/clear-to-send (RTS/CTS) frames are exchanged to increase the reliability of the unicast traffic. In the case of transmission failure, the involved transmitters must wait for ACK/CTS timeout $T_{timeout}$, which is defined as [46]:

$$T_{timeout} = SIFS + T_{ACK/CTS} + T_H + \sigma \tag{4.2}$$

where $SIFS$ is the length of short inter-frame space (SIFS), $T_{ACK/CTS}$ is the time to send an ACK or a CTS frame, T_H is the time to send the preamble and the physical layer header. This timeout delays the involved transmitters to access the channel again after a transmission failure even if the transmitters select zero as their new backoff time; this permits other stations to backoff accordingly. In other words, only one station is monopolizing the channel when CFP happens in the unicast.

In the broadcast, CFP could happen either after a succussful transmission or after a failed transmission. Because no ACK and RTS/CTS frames are used in broadcast, stations that have just sent a frame cannot know the outcome of the transmission. If

anyone of them has a new packet ready right after the end of the transmission, it is possible to select zero as its new backoff time and thus monopolizes the channel. In other words, one or more stations could possibly monopolize the channel simultaneously when CFP happens in the broadcast. In addition, lack of MAC-level acknowledgement mechanism in the broadcast causes that the contention window size of each broadcasting station stays constant and minimum. Thus the station's probability of selecting zero as backoff time, which is the reciprocal of the contention window size, stays constant and maximum. These collectively lead to more frequent occurrence of CFP in the broadcast than in the unicast.

4.3 Performance Analysis

CFP was studied in [80, 82] for the unicast. Given the number of contending stations, they assumed the following two probabilities are constant and do not depend on the backoff procedure.

- the probability that a packet transmitted shall collide and;
- the probability that a station transmits in a randomly chosen slot.

However, the CFP differences in between IEEE 802.11 unicast and broadcast make this assumption no longer tenable. In broadcast, these probabilities actually do depend on the backoff procedure. Therefore, these two models cannot be directly applied for the analysis of broadcast and the backoff operation of broadcast stations cannot be characterized as only a Markov process.

By exploiting the special features of the backoff counter in IEEE 802.11 broadcast, the backoff counter process is divided into two involving sub-processes: the sequential backoff process (SBP) which involves the general backoff procedure without zero initial backoff time and the CFP which involves consecutive transmissions as a result of zero initial backoff time.

4.3.1 Packet Transmission Probabilities

First, an analytic model is constructed for the SBP in which the probability that a station transmits in a virtual slot time is assumed reasonably to be constant as it was in [59, 80, 81]. A virtual slot is the time interval between two consecutive backoff counter decrements of non-transmitting stations [81]. Consider a fixed number n of

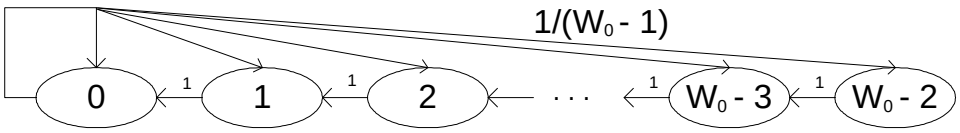

Figure 4.2: Markov chain model for SBP in broadcast

contending stations. As described in [81], the stochastic process $\{b_k\}$, indexed by virtual slot number k, is a one dimensional discrete-time Markov chain. The state transition diagram describing backoff counter decrement is shown in Fig. 4.2, with the only nonnull one-step transition probabilities being

$$\begin{cases} P\{k|k+1\} = 1, & k \in [0, W_0 - 3]; \\ P\{k|0\} = 1/(W_0 - 1), & k \in [1, W_0 - 2]. \end{cases} \tag{4.3}$$

From Fig. 4.2, the following relations can be derived through chain regularities

$$b_k = \tfrac{W_0 - k - 1}{W_0 - 1} b_0, \quad 0 < k \leq W_0 - 2 \tag{4.4}$$

$$\sum_{k=0}^{W_0-2} b_k = 1. \tag{4.5}$$

Let τ_s be the probability that a station transmits in the SBP during a virtual slot time. As any transmission occurs when the backoff counter value is equal to zero, solving (4.4) and (4.5), we have

$$\tau_s = b_0 = \frac{2}{W_0}. \tag{4.6}$$

Then, the CFP is analyzed. Packet transmissions triggered by zero initial backoff time could happen right after a busy period resulted from the SBP, which is referred to as first freeze stage, or could happen right after a freeze stage, which is referred to as the i^{th} freeze stage ($i = 2,\ 3,\ \ldots$). Under the saturation condition, a station that just completes a transmission in the previous freeze stage will attempt to send one more packet out. It chooses zero as its initial backoff time with a uniform probability $1/W_0$. The CFP will continue until zero initial backoff time is no longer chosen by any station which attempts to send new packets out. Here, performance analysis on the CFP is approached through evaluating transmission probability of each station in each stage. Define $\tau_f(i)$ ($i = 1,\ 2,\ \ldots$) to be the probability that a station transmits i packets consecutively in a virtual slot due to zero initial backoff time. Hence, given the initial probability τ_s, the probability that such a station transmits in the first freeze stage is derived as

$$\tau_f(1) = \frac{\tau_s}{W_0}. \tag{4.7}$$

Consequently, the probability that such a station transmits in the i^{th} freeze stage can be evaluated as

$$\tau_f(i) = \frac{\tau_f(i-1)}{W_0} = \frac{\tau_s}{W_0^i}, \quad i = 2,\ 3,\ \ldots \tag{4.8}$$

4.3.2 Channel Performance

Now, let's consider evaluation of channel performance resulted from SBP. Let p_{bs} denote the probability that the channel is busy as a result of SBP. Given τ_s, we have

$$p_{bs} = 1 - (1 - \tau_s)^n. \tag{4.9}$$

Furthermore, the probability p_{ss} that a successful transmission occurs is derived as

$$p_{ss} = \binom{n}{1}\tau_s(1 - \tau_s)^{n-1} = n\tau_s(1 - \tau_s)^{n-1}. \tag{4.10}$$

Then, let's consider the channel performance resulted from the CFP. Knowing $\tau_f(i)$ ($i = 1,\ 2,\ \ldots$), the probability $p_{sf}(i)$ that a successful transmission occurs in the i^{th} freeze stage is obtained

$$p_{sf}(i) = n\tau_f(i)(1 - \tau_f(i))^{n-1}. \tag{4.11}$$

Similarly, the probability $p_{bf}(i)$ that the channel is busy in the i^{th} freeze stage, is derived as

$$p_{bf}(i) = 1 - (1 - \tau_f(i))^n. \tag{4.12}$$

Let $E[N_{sf}]$ denote the average number of successful transmissions in the CFP in a virtual slot time, which is expressed as

$$E[N_{sf}] = \sum_{i=1}^{\infty} 1 \cdot p_{sf}(i) = \sum_{i=1}^{\infty} \frac{n\tau_s}{W_0^i}(1 - \frac{\tau_s}{W_0^i})^{n-1}. \tag{4.13}$$

Under the conditions $\frac{\tau_s}{W_0^i} = \frac{2}{W_0^{i+1}} \ll 1, i > 0$, and $W_0^i \gg n > 1$,

$$(1 - \frac{\tau_s}{W_0^i})^n \approx 1 - \frac{n\tau_s}{W_0^i} \tag{4.14}$$

holds, and provides the following approximate expression:

$$E[N_{sf}] \approx \frac{n\tau_s}{W_0 - 1} - \frac{n(n-1)\tau_s^2}{W_0^2 - 1}. \tag{4.15}$$

Let $E[N_{bf}]$ be the average number of freeze stages on which the channel is busy. We have

$$E[N_{bf}] = \sum_{i=1}^{\infty} 1 \cdot p_{bf}(i) = \frac{n\tau_s}{W_0 - 1}. \tag{4.16}$$

Let T_s be the average time the channel is sensed busy because of a successful transmission and T_c be the average time the channel is sensed busy by each station during a collision. It is assumed that the packet average length is $E[P]$. Let δ be the propagation delay, and $DIFS$ be the time period for a distributed inter-frame space. All are in the same unit.

$$T = T_s = T_c = T_H + E[P] + DIFS + \delta. \tag{4.17}$$

4.3.3 Performance Indices

4.3.3.1 Saturation Throughput

Let S be the normalized saturation throughput, expressed as the fraction of time the channel is used to transmit payload bits successfully in a virtual slot time. Effects on the throughput from both the SBP and the CFP are combined. Considering that there must be an idle backoff slot in each virtual slot, we obtain

$$S = \frac{E[P]'}{E[L_{vs}]} \tag{4.18}$$

where

$$E[P]' = (1 \cdot p_{ss} + E[N_{sf}]) \cdot E[P] \tag{4.19}$$

is the average amount of payload bits transmitted successfully in a virtual slot and

$$E[L_{vs}] = \sigma + (1 \cdot p_{bs} + E[N_{bf}]) \cdot T \tag{4.20}$$

is the average time of a virtual slot.

4.3.3.2 Saturation Delay

$E[D]$ is the average delay a packet experiences between the time at which the packet is generated and the time at which the packet is successfully received under the saturation condition. It includes the medium access delay (due to backoff, and channel busy, interframe spaces, etc.), transmission delay, and propagation delay. The delay that a successfully transmitted packet experiences could be either the delay D_b for a successful packet transmission by sequential backoff process, or the delay D_f for a successful packet transmission by choosing zero initial backoff time.

It is evident that $E[D_f] = T$. On the other hand, the average delay $E[D_b]$ is given by

$$E[D_b] = E[X] \cdot E[L_{vs}] \tag{4.21}$$

where $E[X]$ denotes the average number of the virtual slot that a packet successfully transmitted observes. As we know that each virtual slot contains only one backoff slot time, $E[X]$ equals to the average number of initial backoff slot. Therefore, we have $E[X] = W_0/2$.

Let p_{fr} be the probability that given a successful transmission it is carried out through choosing zero initial backoff time, which is evaluated as

$$p_{fr} = \frac{E[N_{sf}]}{E[N_{sf}] + 1 \cdot p_{ss}}. \tag{4.22}$$

So, the average saturation delay $E[D]$ is obtained as

$$E[D] = p_{fr} \cdot E[D_f] + (1 - P_{fr}) \cdot E[D_b]. \tag{4.23}$$

4.3.3.3 Packet delivery ratio (PDR)

PDR is defined as the ratio of the number of packets successfully received to the number of packets transmitted. So, PDR can be interpreted as $1 \cdot p_{ss} + E[N_{sf}]$ that the number of successfully transmitted packets during a virtual slot over $E[N_{pktvs}]$ which is the average number of stations transmitting packets in a virtual slot.

$$PDR = \frac{1 \cdot p_{ss} + E[N_{sf}]}{E[N_{pktvs}]}. \tag{4.24}$$

Given the probability τ that a station transmits in a virtual slot, the average number of stations transmitting packets in a virtual slot is $n\tau$ [83]. So, we have

$$E[N_{pktvs}] = n\tau_s + \sum_{i=1}^{\infty} n\tau_f(i) = n\tau_s(1 + \frac{1}{W_0 - 1}). \tag{4.25}$$

4.4 Simulation

A highly efficient MatLab program was developed to conduct a fast computer simulation on one-hop 802.11 MAC layer broadcasting. The simulation time is significantly shortened by calculating and predicting the time when the channel becomes active again to skip periods when the channel content is not meaningful. For example backoff, DIFS, collision, etc. Following pseudo code describes the simulation structure briefly.

`main.m`

```
  Parameter setup based on user input;
  Parameter print and confirm in main.m;
  Construct simPara and pass it to function satu_bc_80211_init();
satu_bc_80211_init.m
  Initialize nodes' location based on the input from main.m;
  Initialize packet generation time on each node;
  Initialize backoff count using the minimum window size;
  Initialize the first event on each node;
  Initialize statistic variables;
  Return to main.m
main.m
  Pass the initialized parameters to function satu_bc_80211();
satu_bc_80211.m
  Find the next event time;
  Identify all the node that'll be active @ next event time;
  Calculate if collision is happening at this time;
  Do statistics;
  Calculate the next event time for each timed out node;
  Go back to the first step unless simulation time is up;
  Return simulation results to main.m
main.m
  Print and plot simulation results;
```

The source code of the simulation can be found in Appendix B.

4.5 Numerical Results And Discussions

All the parameters used in the analytical model and in the computer simulation follow the parameters in paper [59] for consistency, and are summarized in TABLE 4.1.

Fig. 4.3 shows the saturation throughput over the number of stations with different contention window sizes. As it is seen in Fig. 4.3 that the analytical model highly matches the simulation results. It is also observed that the saturation throughput for broadcast is much lower than that of basic access mechanism or RTS/CTS mechanism for unicast service in IEEE 802.11 [59]. Increasing window size helps improve the throughput performance of the broadcast service. In contrast to the throughput evaluation without

Table 4.1: IEEE 802.11 FHSS System Parameters

Parameters	**Value**
Packet Payload, P	8184 $bits$
MAC header	272 $bits$
PHY header	128 μs
Bit rate	1 $Mbit/s$
Propagation delay, δ	1 us
DIFS	128 μs
Slot time, σ	50 μs

taking CFP into consideration, the throughput derived from our proposed model is higher as the network traffic becomes heavier.

Fig. 4.4 shows the saturation packet delay over the number of stations with different contention window sizes. The analytical results (lines) coincide with the simulation results (symbols) very well. As demonstrated in Fig. 4.4, the larger the number of stations is, the longer the packet delay will be expected when the contention window size is relatively small ($W_0 = 32$). Explanation for this observation is that as the contention window size is small compared to the number of stations in the network, the probability that zeros are chosen as initial backoff times increase. Therefore, there are more and more successful transmissions triggered by choosing zeros as initial backoff times while the network traffic becomes heavier. Such packets are transmitted right after a DIFS period, which reduces the average delay. Under such a circumstance, ignoring the CFP in the analytic model will bring significant errors to the evaluation of the packet delay, and will even result in different tendency.

Fig. 4.5 shows the packet delivery ratio over the number of stations with different contention window sizes. The analytical results (lines) match the simulation results (symbols) very well. The packet delivery ratio demonstrates similar characteristics to those of the throughput. As we observe, ignoring the CFP will bring some modeling errors to the evaluation of the PDR when the window size is relatively small ($W_0 = 32$) and the network traffic is heavier ($n > 30$). It is also observed that the PDR only depends on the number of stations n and the selected contention window size W_0.

When a bigger window size is chosen ($W_0 = 128$), no significant difference can be

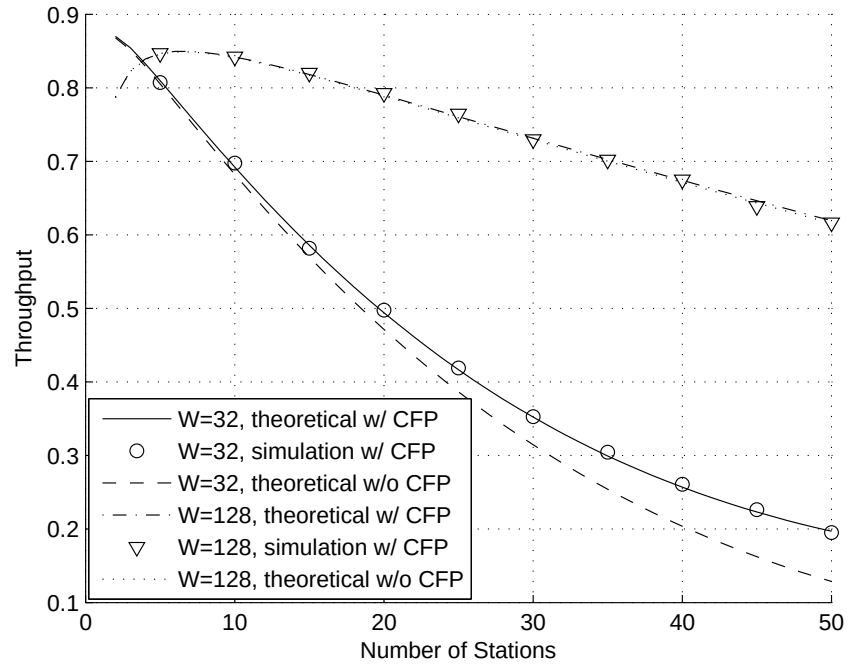

Figure 4.3: Saturation throughput of IEEE 802.11 for broadcast

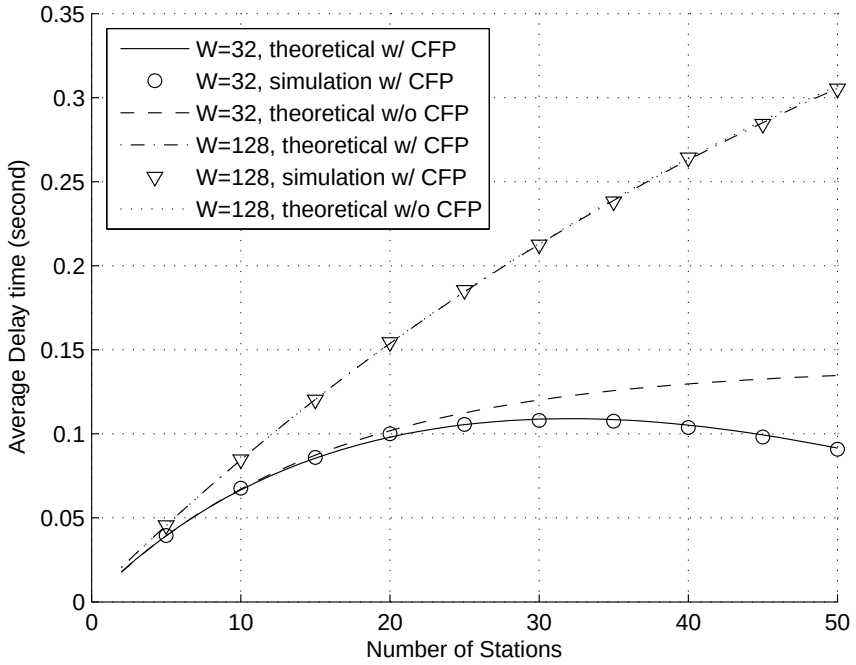

Figure 4.4: Saturation delay of IEEE 802.11 for broadcast

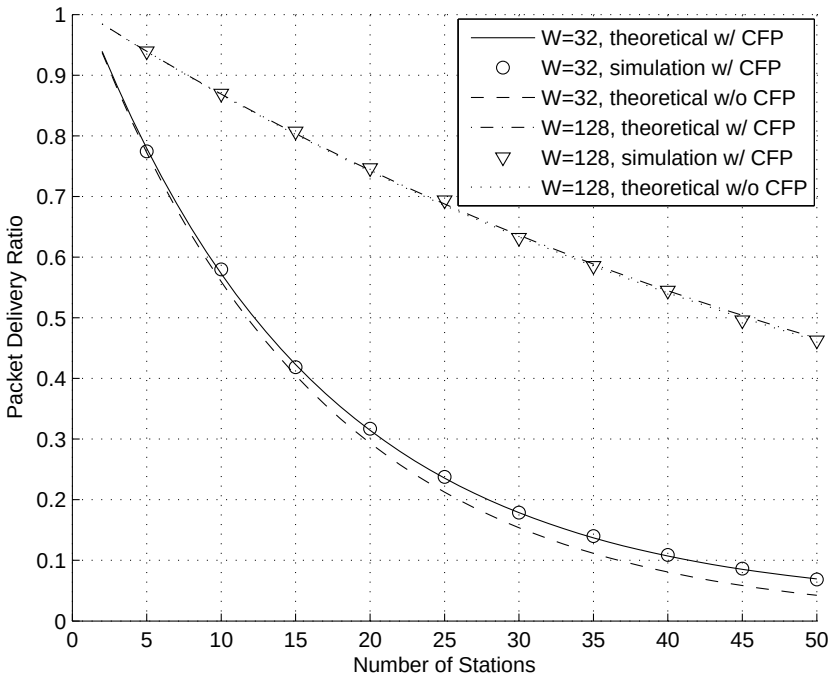

Figure 4.5: Saturation packet delivery ratio of IEEE 802.11 for broadcast

observed between the performance with CFP and the performance without taking CFP into consideration. This is because CFP happens less likely when the contention window size becomes bigger.

CHAPTER 5

Performance of IEEE 802.11 MAC In A Highway Scenario

With regard to DSRC, studies were conducted to analyze or to improve its performance [3, 16, 45, 53, 84, 19]. In [3] limitations of 802.11a in DSRC environment are identified. In [16] certain communication protocols are used to manage inter-vehicle communication for highway cooperative collision avoidance. Some protocols [45][53] introduce feedback or acknowledgement information from receivers to reduce unnecessary forwarding transmission. A cluster-based DSRC architecture for QoS provisioning over vehicular ad hoc networks is proposed in [84] to support the real-time transmission of safety related messages. After analyzing the DSRC control channel for inter-vehicle safety messaging, authors of [19] propose several MAC protocols to improve the reception reliability (namely, the PDR) and the channel throughput.

Broadcast service is the basis for the research in the aformentioned papers. This is fairly intuitive and reasonable because inter-vehicle safety applications are local and because broadcast is able to cover the locality with low latencies. In the government research reports [4][85], the broadcast is also proposed as the communication mode for the highway safety application because of its significant advantage over point-to-point wireless communications. Although the broadcast performance of IEEE 802.11 MAC is studied in [86], the quantitative performance of the current DSRC broadcast has not been researched. As a matter of fact, it is very important to know quantitatively the performance in a given environment in order to successfully apply a technology to a real application or to make improvements.

A quantitative approach is presented to evaluate the quality of service that the inter-vehicle communication (using IEEE- and ASTM-adopted dedicated short range communication (DSRC)) can provide to the inter-vehicle safety application in the context of highway scenarios. Based on the proposed model, performance indices such as delay and packet delivery ratio (PDR) are analyzed. Outputs of the model are validated by computer simulations. Our quantitative model provides not only a convenient framework to conduct fast evaluation for the communication of the inter-vehicle safety applications but also a means to analyze the suitability of 802.11a-based DSRC for highway safety applications. Furthermore, it reveals the needs for further improvements on the protocol development.

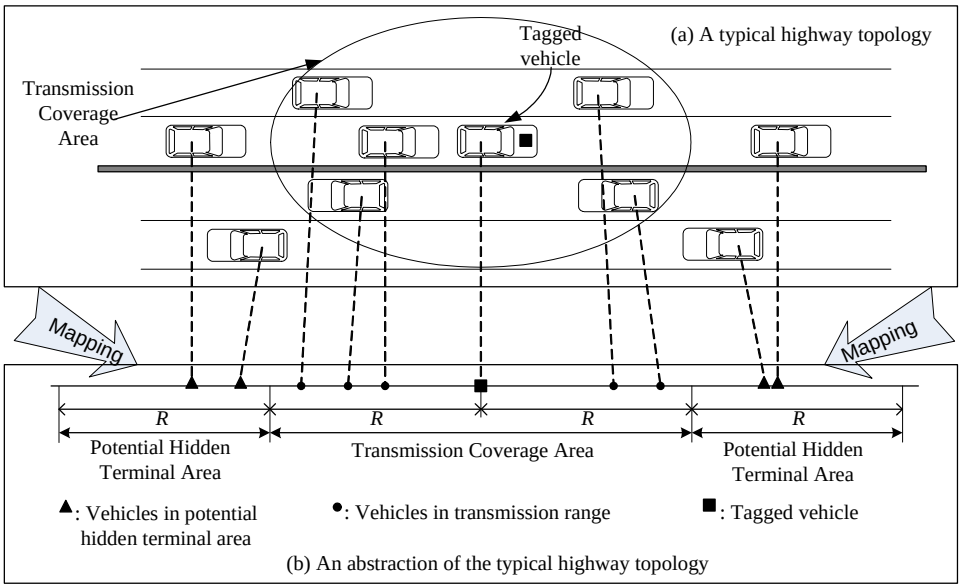

Figure 5.1: Highway topology abstraction: 2-D to 1-D

5.1 System Model And Analysis

5.1.1 Assumptions

Real-world radio networks are influenced by many factors. In this study, some assumptions are made to build a simplified yet reasonable and extendable model.

A typical highway topology illustrated in Fig. 5.1 (a) is abstracted into a one-dimensional mobile ad hoc network consisting of a collection of statistically identical mobile stations randomly located on a line. This abstraction is viable when the network size is exceedingly large and mobile nodes are placed with certain finite network density [87][88]. In this scenario, as shown in Fig. 5.1, the distance between the parallel lanes is neglected compared to the extended length of the vehicle network. The distribution of vehicles on this abstract line is according to a Poisson point process with a density β (in vehicles per meter); i.e. the probability $P(i,l)$ of finding i vehicles in length of l is given by

$$P(i,l) = \frac{(\beta l)^i \cdot e^{-\beta l}}{i!}. \tag{5.1}$$

A vehicle is chosen (called tagged vehicle) and placed in the origin. Its transmission and sensing range are assumed equal to R. The potential hidden terminal area of the tagged vehicle in broadcast communication drops in the range of $[R, 2R]$ and $[-2R, -R]$.

At each vehicle, packets arrive in Poisson process with rate λ (in packets per second). Here, we notice that the same assumption has been widely used to keep the tractability

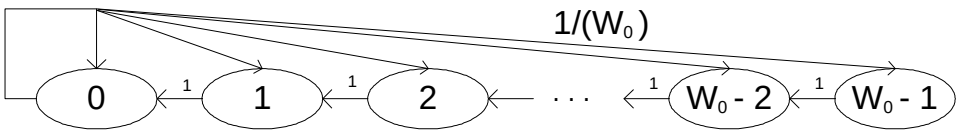

Figure 5.2: Markov chain model for IEEE 802.11 broadcast

of the anaytical model [89]. At the MAC layer, the packet queue length of each terminal is unlimited. The reason for this assumption is that safety related messages at each vehicle are short, too critical to drop, and expected to arrive once in a while. Hence each vehicle can be modelled as a discrete time M/G/1 queue.

Because of the hidden terminals problem and the unsaturated traffic, terminals' knowledge of time slots might be heterogeneous, which makes the synchrony of backoff operation described in [59] seem no longer valid. In this model, this asynchrony is ignored on purpose to approximate the process. This approach is validated by finding that the theoretical results match the simulation very closely. Therefore it is assumed that the asynchrony is negligible and the quasi-synchrony is advisable under the given conditions in this paper.

Based on the above assumptions, in this paper, the performance for a snapshot of the highway vehicular network is studied in a perfect channel.

5.1.2 Backoff Process in IEEE 802.11 Broadcast

A model is constructed to characterize backoff counter process in IEEE 802.11 broadcast network which is a simplified one as in [59]. The stochastic process $\{b_k\}$, indexed by backoff counter value k of a broadcast terminal, is a one-dimensional discrete-time Markov chain. The state transition diagram describing backoff counter decrement is shown in Fig. 5.2, with the only nonnull one-step transition probabilities being

$$\begin{cases} P\{k|k+1\} = 1, & k \in [0, W_0 - 2]; \\ P\{k|0\} = 1/(W_0), & k \in [1, W_0 - 1]. \end{cases} \tag{5.2}$$

We can derive following relations through chain regularities

$$b_k = \frac{W_0 - k}{W_0} b_0; \quad \sum_{k=0}^{W_0 - 1} b_k = 1. \tag{5.3}$$

Let τ be the probability that a station transmits. Because a transmission occurs when the backoff counter value equals zero, solving (5.3), we have

$$\tau = b_0 = \frac{2}{W_0 + 1}. \tag{5.4}$$

When the channel is detected busy, the backoff timers of the detecting terminals will be suspended and deferred a time period of T' which is expressed as

$$T' = DIFS + T_H + E[P] + \delta, \tag{5.5}$$

where $DIFS$ is the time period for a distributed inter-frame space, the packet header T_H includes physical layer header plus MAC layer header, $E[P]$ is the average packet length, and δ is the propagation delay. All are in the same unit.

5.1.3 Performance of the Tagged Vehicle

Let N_{cs} denote the average number of vehicles in carrier sensing range of the tagged vehicle and let N_{tr} denote the average number of vehicles in transmission range of the tagged vehicle, From Fig. 5.1, we have

$$N_{cs} = N_{tr} = 2\beta R. \tag{5.6}$$

Let N_{ph} denote the average number of vehicles in the potential hidden terminal area:

$$N_{ph} = 4\beta R - N_{cs}. \tag{5.7}$$

Now, let's calculate channel performance from the tagged vehicle's point of view. Define p_b as the probability that the tagged vehicle senses channel busy. Knowing that the channel is busy if there is at least one vehicle transmitting in the transmission range of the tagged vehicle, we have

$$p_b = 1 - \sum_{i=0}^{\infty}(1 - p_0\tau)^i \frac{(N_{tr})^i}{i!} e^{-N_{tr}} = 1 - e^{-N_{tr}p_0\tau}, \tag{5.8}$$

where p_0 is the probability that there are packets ready to transmit, which will be derived later in the paper.

5.1.4 Service Time

The characteristics of each station in the network is modeled as an M/G/1 queue and approach service time distribution through a probability generating function (PGF). The MAC layer service time is the time interval from the time instant when a packet becomes the head of the queue and starts to contend for transmission, to the time instant when the packet is received. Apparently, the distribution of the MAC layer service time is a discrete probability distribution when the time unit of the backoff timer is a time

slot σ. It is understood that the backoff counter in each station will be decremented by a slot once an idle channel is sensed, and will wait for a transmission time once a busy channel is sensed. For a station in broadcast communication, the transition for backoff counter decremented by one slot can be expressed by the following PGF,

$$H_d(z) = (1 - p_b)z + p_b \cdot z^{\lfloor T/\sigma \rfloor}. \tag{5.9}$$

Denote q_i as the steady state probability that the packet service time is $i\sigma$. Let $Q(z)$ be the PGF of q_i, which is

$$Q(z) = \sum_i q_i z^i. \tag{5.10}$$

Due to the simplicity of notation in the z-transform domain and the one-to-one correspondence between $Q(z)$ and q_i, let's discuss how to calculate $Q(z)$ instead of individual q_i.

Now, it is possible to draw the generalized state transition diagram for the broadcast transmission process as shown in Fig. 5.3. Knowing that successful transmission and transmission with collision take the same amount of time in broadcast, $SC(z) = z^{\lfloor (P+T_H)/\sigma \rfloor}$ can be obtained. From Fig. 5.3, the transfer function of the linear system or distribution of the service time can be derived as

$$Q(z) = \sum_i q_i z^i = \frac{z^{\lfloor \frac{P+T_H}{\sigma} \rfloor}}{W_0} \sum_{i=0}^{W_0-1} H_d^i(z). \tag{5.11}$$

Based on (5.11), the arbitrary n^{th} moment of service time can be obtained by differentiation. Therefore, the average service time can be obtained by

$$T_{ave}^s = \sum_i q_i(i\sigma) = Q'(z)|_{z=1}. \tag{5.12}$$

In order to derive the average service time distribution, the probability p_0 must be determined, while p_0 calculation depends on duration of service time. Here, the iterative algorithm is applied to calculate p_0. The iterative steps are outlined as follows.

Step 1 : Initialize $p_0 = 1$, which is the saturated condition.

Step 2 : With p_0, calculate p_b according to (5.8).

Step 3 : Calculate service time distribution through PGF.

Step 4 : Service rate $\mu = 1/Q'(1)$.

Step 5 : if $\lambda < \mu$, $p_0 = \lambda/\mu$; otherwise, $p_0 = 1$.

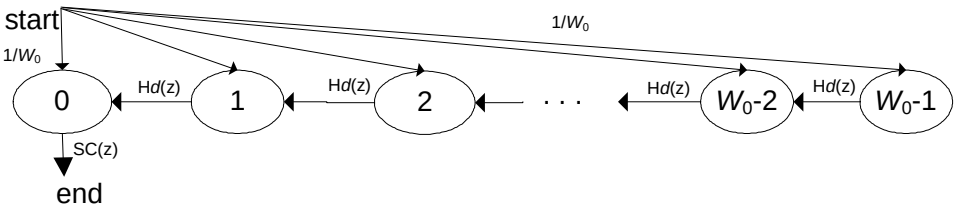

Figure 5.3: Generalized state transition diagram for broadcast

Step 6 : If p_0 converges with the previous values, then stop the algorithm; otherwise, go to step 2 with the updated p_0.

5.1.5 Performance Indices

5.1.5.1 Delay

Packet transmission delay $E[D]$ is the average delay that a packet experiences between the time at which the packet is generated and the time at which the packet is successfully received. It includes the medium service time (due to backoff, transmission delay, and propagation delay, etc.) and the queuing delay.

The expected queuing delay can be obtained by the *Pollaczek-Khinchin* mean value formula for M/G/1 queues.

$$E[D_q] = \frac{\lambda(Q''(1) + Q'(1))}{2(1-\rho)}. \tag{5.13}$$

The average packet transmission delay can be calculated as

$$E[D] = E[D_q] + T^s_{ave} + DIFS + \sigma + \delta. \tag{5.14}$$

5.1.5.2 Packet Delivery Ratio

Packet delivery ratio (PDR) is defined as the ratio of the number of packets successfully received to the number of packets transmitted. So, PDR can be interpreted as, given a broadcast packet sent, the probability that all vehicles within transmission range of the tagged vehicle receive the packet successfully. Taking hidden terminal into account, we have

$$PDR = P(N_{cs}) \cdot P(N_{ph}) = e^{-(N_{cs} + \frac{T_v}{\sigma + p_b T'} N_{ph} - 1) p_0 \tau}, \tag{5.15}$$

where

$$P(N_{cs}) = \sum_{i=0}^{\infty} (1 - p_0\tau)^i \frac{(N_{cs}-1)^i}{i!} e^{-(N_{cs}-1)}, \tag{5.16}$$

is the probability that none of the other vehicles within the transmission range of the tagged vehicle transmits when the tagged vehicle starts transmission and

$$P(N_{ph}) = \Big[\sum_{i=0}^{\infty}(1-p_0\tau)^i \frac{(N_{ph})^i}{i!} e^{-N_{ph}}\Big]^{\lfloor \frac{T_v}{\sigma+p_b T'} \rfloor}, \tag{5.17}$$

is the probability that none of the vehicles in the two potential hidden terminal areas (see Fig. 5.1) transmits during the hidden terminal vulnerable period $T_v = 2(E(P) + T_H)$. The expression $\lfloor \frac{T_v}{\sigma+p_b T'} \rfloor$ discretizes T_v into the number of time slots.

5.2 Simulation

An event and message driven computer simulation program is developed by using MatLab. This simulation program support 802.11 basic unicast, RTS/CTS unicast, and broadcast. The following pseudo code describes the implementation briefly.

`ao2dot11.m`

```
Define the state machine;
Define message types;
Define event types;
Define three packet types;
Define other constants such as PHY and MAC parameters;
Set current state of each node to IDLE;
Initialize message box;
Initialize each communication node;
while simulation time is not up, do the following
  Find the nearest event time;
  Notify associated stations about the event by messaging;
  All messages are stored in the message box;
  Process all messages according to priority;
Print and plot simulation results;
```

The source code of the simulation can be found in Appendix C.

Table 5.1: Other Parameters

Parameters	**Value**	**Parameters**	**Value**
Propagation delay	1 μs	Tx range, R	500 m
Average packet length	variable	Packet arrival rate, λ	variable
Vehicle density, β	variable	CWMin, W_0	15
SIFS	32 μs	Slot size, σ	16 μs
PHY preamble	40 μs	PLCP header	4 μs

5.3 Numerical Results And Discussions

All DSRC parameters used in this paper follow [10] and are listed in TABLE 5.1.

Fig. 5.4 and Fig. 5.5 depict the delay and packet delivery ratio, respectively, over the density of vehicles on the road with varied data rate R_d Mbps, average packet length $E[P]$ bytes, and packet arrival rate λ (packets per minute). The results from the model and the simulation match well. The differences between them are mainly due to limited road range in the simulation and possible asynchrony of time slots among vehicles. The biggest delay observed is about 0.6 ms. Therefore, the latency requirement of 100 ms in [4] and the range from 0.1 second to 0.4 second for the CCA system [16] can definitely be satisfied. The phenomenon results from high date rate, small backoff window size, and the proposed one-hop direct transmission strategy. However, The packet delivery ratios decrease fast away from the reliability requirement 0.99 [19] for DSRC safety critical messaging when the vehicle density increases. Therefore, when designing such a DSRC broadcast-based highway safety system, designers can relax the stringent constraints of considering the delay requirement and have the flexibility to use this to compensate the deficiency of other performance indices, such as PDR.

The delay and PDR both benefit from a shorter message length because the shorter the message is, the lower the probability of collision shall be. Therefore, it is significantly critical to have a semantically concise and accurate description of highway safety events in a DSRC highway safety system because this determines the length of the message.

In Fig. 5.4, it is observed that the delay decreases when the parameter setting is changed from $R_d = 12, \lambda = 2$ to $R_d = 24, \lambda = 10$. It is because the increase of data rate decreases the duration of the vulnerable period, given the same $E[P]$. Although the increase of λ indeed increases the data traffic which might be able to lengthen the

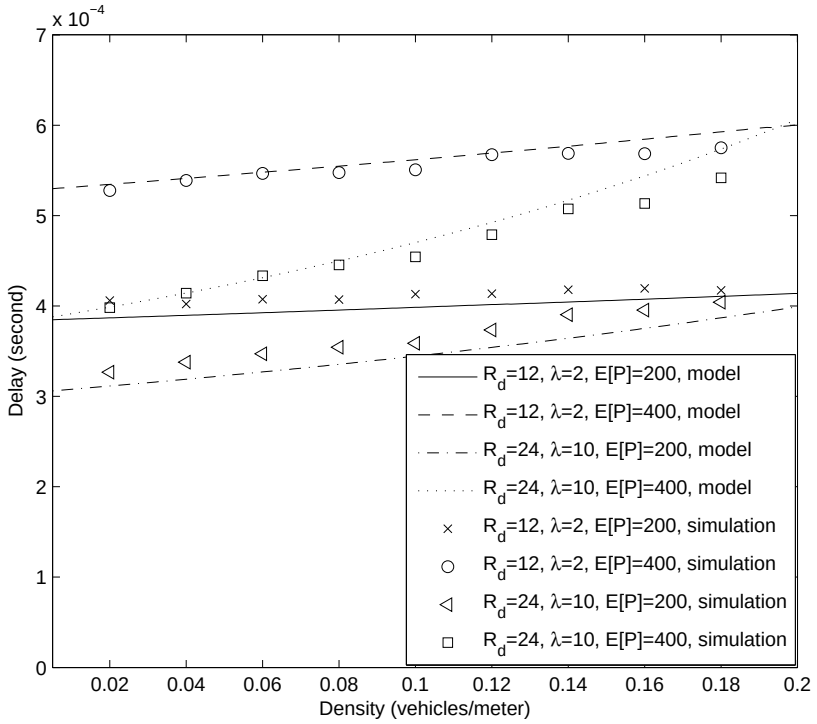

Figure 5.4: Delay of DSRC Highway safety messaging

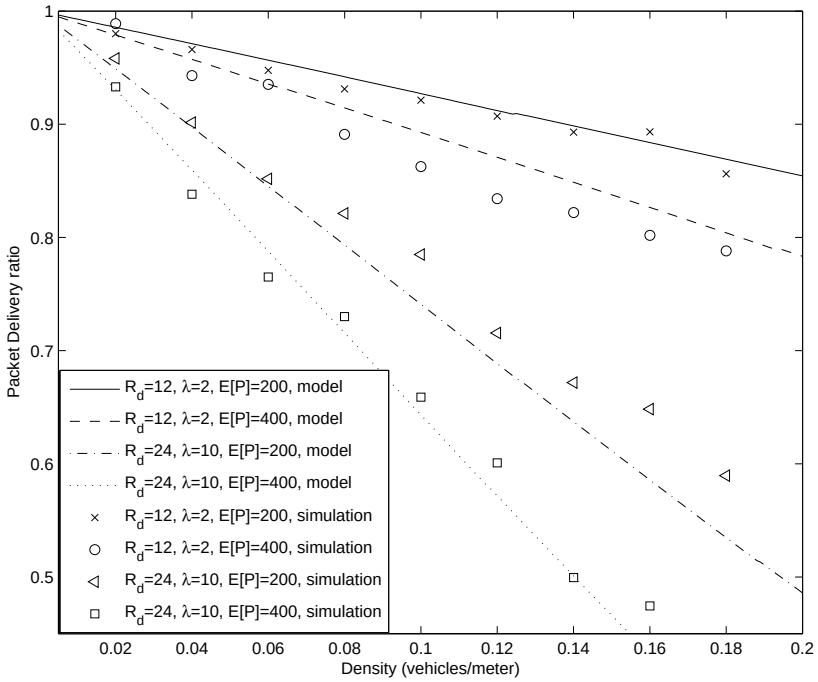

Figure 5.5: PDR of DSRC highway safety messaging

delay, the overall change of the delay still drops due to the increase of the data rate.

CHAPTER 6

Conclusion and Future Work

This research effectively demonstrates that the proposed VANET MAC and 802.11a MAC enhancements have yet to meet critical requirements, e.g. when applying 802.11a MAC to a one-hop broadcast network with saturated traffic, both delay and PDR performance is unacceptable. Although a 100ms delay is observed in a highway scenario with un-saturated traffic, a low PDR performance persists.

As the supporting technology for the future Intelligent Transportation System, VANET will provide a versatile and flexible communication platform for vehicle-to-vehicle and vehicle-to-roadside communication. Researchers and engineers continue their diligent work developing proposals toward a VANET MAC and PHY that will serve as the de facto standard. The guidelines for future design and enhancement are as follows.

- The legacy IEEE 802.11 back-off mechanism must be improved to reduce collision caused by simultaneous transmission. The current adaptive back-off mechanisms presented in section 2.5.3 are not effectively implemented in VANET. Further study and design work are required.

- A priority mechanism might be considered when multiple packet priority exists. For example, a safety-related broadcast packet could perhaps have higher priority than a normal packet.

- For (multicast, ACK) + (broadcast, NAK), we propose the use of multicast to deliver safety-critical messages to the most vulnerable neighboring vehicles. (How to define and determine critical vulnerability remains uncertain for specific applications, but could be, for example, the vehicle closest in proximity.) As mentioned in Section 2.5.5, the locality of safety-critical applications is an important characteristic, one that has eluded researchers to this point. Locality is the determinate for identifying that the most critical messages are those closest to the event-bearing vehicle. Meanwhile, broadcast is applied to neighboring, but not the most critical, vehicles of the sender. LAMM [52] can be utilized to achieve satisfactory QoS in multicast for those vehicles nearest to the sender, thus increasing the broadcast QoS. For multicast, ACK is used; for broadcast, NAK. As long as the reception rate within the specified range is satisfied, the broadcast is acceptable. A high reception rate within the transmission range of the transmitter is unnecessary.

- Utilize alternate forwarding schemes for different scenarios. For example, trucks (NLOS) must forward safety messages; however, cars (small vehicles LOS) do not.
- Delay can be sacrificed to obtain PDR gain. In Fig. 5.4, the delay is always observed lower than $2.7ms$. The PDR shown in Fig. 5.5 is too low to satisfy the reliability requirement. Because the latency requirement for VSCAs is $100ms$ which is much longer than $2.7ms$, it is advisable to modify the protocol so as to achieve higher PDR; however, it is speculated that this modification might incur allowable penalty on the latency.
- A uniform testing environment (real or simulation) is needed to verify the validity of each VANET MAC proposal.
- Due to the vital importance of a safety critical message, it is imperative that a continuous failure of transmission occurs as a result of a burst of errors. In this instance, safety-related events might occur as a result of the burst loss of information. However, the average performance of the MAC might still be acceptable in spite of the burst loss, as it is compensated for in volumes of successful transmission. Therefore, when designing MAC to support VSCAs, it is unacceptable to look at the long-term average QoS. The way in which the protocol deals with the burst of errors should also be a critical criterion to measure the QoS.

The theoretical model and simulation platform presented are expected to serve as convenient tools for rapid algorithm evaluation and verification, thus providing an accurate design reference for researchers and engineers. For future work, assumptions in the current model will be strategically removed, making the new model a more realistic representation, e.g. the hidden node problem considered in two-dimensional space. New algorithms are expected subsequent to their acceptance into the standard.

References

[1] *http://ops.fhwa.dot.gov.*

[2] National Highway Traffic Safety Administration, *Traffic Safety Facts 2005.* USDOT, 2005.

[3] Z. Jing and S. Roy, "MAC for dedicated short range communications in intelligent transport system," *IEEE Communications Magazine*, vol. 41, no. 12, pp. 60–67, 2003.

[4] The CAMP Vehicle Safety Communications Consortium, *Vehicle Safety Communications Project Task 3 Final Report - Identify Intelligent Vehicle Safety Applications Enabled by DSRC.* USDOT, 2005.

[5] S. K. S. Gupta, V. Shankar, and S. Lalwani, "Reliable multicast MAC protocol for wireless LANs," in *Proc. of the IEEE International Conference on Communications (ICC2003)*, 2003.

[6] D. Towsley, J. Kurose, and S. Pingali, "A comparison of sender-initiated and receiver-initiated reliable multicast protocols," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 398–406, 1997.

[7] M. Yamamoto, J. F. Kurose, D. F. Towsley, and H. Ikeda, "A delay analysis of sender-initiated and receiver-initiated reliable multicast protocols," in *Proc. of the 16th IEEE Conference on Computer Communications (INFOCOM1997).*

[8] M. Torrent-Moreno, M. Killat, and H. Hartenstein, "The challenges of robust intervehicle communications," in *Proc. of the 62th IEEE Vehicular Technology Conference (VTC2005-Fall)*, 2005.

[9] J. Yin, T. ElBatt, G. Yeung, B. Ryu, S. Habermas, H. Krishnan, and T. Talty, "Performance evaluation of safety applications over DSRC vehicular ad hoc networks," in *Proc. of the 1st ACM International Workshop on Vehicular Ad Hoc Networks (VANET'04)*, 2004.

[10] A. E2213-03, *Standard Specification for Telecommunications and Information Exchange Between Roadside and Vehicle Systems 5 GHz Band Dedicated Short Range Communications (DSRC) Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* ASTM Intl., 2003.

[11] Q. Xu, T. Mak, J. Ko, and R. Sengupta, "Vehicle-to-vehicle safety messaging in DSRC," in *Proc. of the 1st ACM International Workshop on Vehicular Ad Hoc Networks (VANET'04)*, 2004.

[12] M. Torrent-Moreno, D. Jiang, and H. Hartenstein, "Broadcast reception rates and effects of priority access in 802.11-based vehicular ad-hoc networks," in *Proc. of the 1st ACM International Workshop on Vehicular Ad Hoc Networks (VANET'04)*, 2004.

[13] J. J. Blum, A. Eskandarian, and L. J. Hoffman, "Challenges of intervehicle ad hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 347–351, 2004.

[14] D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich, "Design of 5.9 ghz dsrc-based vehicular safety communication," *IEEE Wireless Communications*, vol. 13, no. 5, pp. 36–43, 2006.

[15] Q. Xu, T. Mak, J. Ko, and R. Sengupta, "Layer-2 protocol design for vehicle safety communications in dedicated short range communications spectrum," in *Proc. of the 7th International IEEE Conference on Intelligent Transportation Systems (ITSC2004)*, 2004.

[16] S. Biswas, R. Tatchikou, and F. Dion, "Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety," *IEEE Communications Magazine*, vol. 44, no. 1, pp. 74–82, 2006.

[17] S. Sibecas, C. A. Corral, S. Emami, and G. Stratis, "On the suitability of 802.11a/RA for high-mobility DSRC," in *Proc. of the 55th IEEE Vehicular Technology Conference (VTC2002-Spring)*, 2002.

[18] B. Gallagher, H. Akatsuka, and H. Suzuki, "Wireless communications for Vehicle safety: Radio Link Performance & Wireless Connectivity Methods," *IEEE Vehicular Technology Magazine*, 2006.

[19] Q. Xu, K. Hedrick, R. Sengupta, and J. VanderWerf, "Effects of vehicle-vehicle/roadside-vehicle communication on adaptive cruise controlled highway systems," in *Proc. of the 56th IEEE Vehicular Technology Conference (VTC2002-Fall)*, 2002.

[20] S. V. Bana and P. Varaiya, "Space Division Multiple Access (SDMA) For Robust Ad Hoc Vehicle Communication Networks," in *Proc. of the 4th Internatinal IEEE Conference on Intelligent Transportation Systems (ITSC2001)*, 2001.

[21] S. Katragadda, C. N. S. Ganesh Murthy, M. S. Ranga Rao, S. Mohan Kumar, and R. Sachin, "A Decentralized Location-based Channel Access Protocol For Inter-vehicle Communication," in *Proc. of the 57th IEEE Vehicular Technology Conference (VTC2003-Spring)*, 2003.

[22] J. J. Blum and A. Eskandarian, "A reliable link-Layer protocol for robust and scalable intervehicle communications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 1, pp. 4–13, 2007.

[23] *maps.google.com*.

[24] F. Borgonovo, L. Campelli, M. Cesana, and L. Coletti, "MAC for ad-hoc inter-vehicle network: services and performance," in *Proc. of the 58th IEEE Vehicular Technology Conference (VTC2003-Fall)*, 2003.

[25] F. Borgonovo, A. Capone, M. Cesana, and L. Fratta, "ADHOC MAC: new MAC architecture for ad hoc networks providing efficient and reliable point-to-point and broadcast services," *Wireless Networks*, vol. 10, no. 4, pp. 359–66, 2004.

[26] F. Borgonovo, L. Campelli, M. Cesana, and L. Fratta, "Impact of user mobility on the broadcast service efficiency of the ADHOC MAC protocol," in *Proc. of the 51th IEEE Vehicular Technology Conference (VTC2005-Spring)*, 2005.

[27] F. Borgonovo, A. Capone, M. Cesana, and L. Fratta, "ADHOC: a new, flexible and reliable MAC architecture for ad-hoc networks," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC2003)*, 2003.

[28] ——, "RR-ALOHA, a Reliable R-ALOHA broadcast channel for ad-hoc inter-vehicle communication networks," in *Med-Hoc-Net*, 2002.

[29] *Intelligent Transportation Systems and Road Safety Report*. European Transport Safety Council (ETSC), 1999.

[30] *www.cartalk2000.net.*

[31] K. Young-Bae, V. Shankarkumar, and N. H. Vaidya, "Medium access control protocols using directional antennas in ad hoc networks," in *Proc. of the 19th IEEE Conference on Computer Communications (INFOCOM2000)*, 2000.

[32] C. Lau and C. Leung, "A Slotted ALOHA packet radio networks with multiple antennas and receivers," *IEEE Transactions on Vehicular Technology*, vol. 39, no. 3, pp. 218–226, 1990.

[33] N. Pronios, "Performance considerations for slotted spread-sprectrum random access networks with directional antennas," in *Proc. of the 41th IEEE Global Communication Conference (GLOBECOM1998)*, 1998.

[34] J. Ward and R. Compton, "Improving the performance of slotted ALOHA packet radio network with an adaptive array," *IEEE Transactions on Communications*, vol. 40, pp. 292–300, 1992.

[35] ——, "High throughput slotted ALOHA packet radio networks with adaptive arrays," *IEEE Transactions on Communications*, vol. 41, pp. 460–470, 1993.

[36] Y. T.-S. and K.-W. Hung, "Design algorithms for multihop packet radio networks with multiple directional antennas stations," *IEEE Transactions on Communications*, vol. 40, no. 11, pp. 1716–1724, 1992.

[37] I. Zander, "Slotted ALOHA multiple packet radio networks with directional antennas," *Electronic letters*, vol. 26, no. 25, 1990.

[38] K. Thanasis, J. Gentian, and T. Leandros, "A MAC protocol for full exploitation of directional antennas in ad-hoc wireless networks," in *Proc. of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc2003)*, 2003.

[39] A. Nasipuri, S. Ye, J. You, and R. E. Hiromoto, "A MAC protocol for mobile ad hoc networks using directional antennas," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC2000)*, 2000.

[40] Z. Huang, C.-C. Shen, C. Srisathapornphat, and C. Jaikaeo, "A busy-tone based directional MAC protocol for ad hoc networks," in *Proc. of IEEE Military Communications Conference (MILCOM2002)*, 2002.

[41] C. Romit Roy, Y. Xue, H. V. Nitin, and R. Ram, "Using directional antennas for medium access control in ad hoc networks," in *Proceedings of the 8th Annual International Conference on Mobile Computing And Networking.*

[42] M. Sadashivaiah, R. Makanaboyina, B. George, and R. Raghavendra, "Performance evaluation of directional MAC protocol for inter-vehicle communication," in *Proc. of the 61th IEEE Vehicular Technology Conference (VTC2005-Spring)*, 2005.

[43] R. R. Choudhury and N. H. Vaidya, "Deafness: a MAC problem in ad hoc networks when using directional antennas," in *Proc. of the 12th IEEE International Conference on Network Protocols (ICNP2004)*, 2004.

[44] X. Ma, X. Chen, and H. H. Refai, "Delay and broadcast reception rates of highway safety applications in vehicular ad hoc networks," in *Proc. of the 26th IEEE Conference on Computer Communications (INFOCOM2007) MOVE Workshop*, 2007.

[45] M. Y. Ravi, C. H. Adithya, S. Mohan, and M. Ranga, "Reliable MAC broadcast protocol in directional and omni-directional transmissions for vehicular ad hoc networks," in *Proc. of the 2nd ACM International Workshop on Vehicular Ad Hoc Networks (VANET)*, 2005.

[46] IEEE-802.11, *Part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications*. IEEE, 1999.

[47] Q. Xu, S. R., J. D., and C. D., "Design and analysis of highway safety communication protocol in 5.9 GHz dedicated short range communication spectrum," in *Proc. of the 57th IEEE Vehicular Technology Conference (VTC2003-Spring)*, 2003.

[48] K. Tang and M. Gerla, "Random access MAC for efficient broadcast support in ad hoc networks," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC2000)*, 2000.

[49] ——, "MAC reliable broadcast in ad hoc networks," in *Proc. of IEEE Military Communications Conference (MILCOM2001)*, 2001.

[50] S. Shiann-Tsong, T. Yihjia, and C. Jenhui, "A highly reliable broadcast scheme for IEEE 802.11 multi-hop ad hoc networks," in *Proc. of the IEEE International Conference on Communications (ICC2002)*, 2002.

[51] J. Chen and M. Huang, "BEAM: broadcast engagement ACK mechanism to support reliable broadcast transmission in IEEE 802.11 wireless ad hoc networks," in *Proc. of the 60th IEEE Vehicular Technology Conference (VTC2004-Fall)*, 2004.

[52] M.-T. Sun, L. Huang, A. Arora, and T.-H. Lai, "Reliable MAC layer multicast in IEEE 802.11 wireless networks," in *International Conference on Parallel Processing (ICPP2002)*, 2002.

[53] W. Si and C. Li, "RMAC: a reliable multicast MAC protocol for wireless ad hoc networks," in *International Conference on Parallel Processing (ICPP2004)*, 2004.

[54] M. Zorzi and R. R. Rao, "Capture and retransmission control in mobile radio," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 8, pp. 1289–1298, 1994.

[55] H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma, "Performance of reliable transport protocol over IEEE 802.11 wireless LAN: analysis and enhancement," in *Proc. of the 21th IEEE Conference on Computer Communications (INFOCOM2002)*, 2002.

[56] Y. Wang and J. J. Garcia-Luna-Aceves, "A new hybrid channel access scheme for ad hoc networks," in *Proc. of the 1st Annual Mediterranean Ad Hoc Networking Workshop (Med-hoc-Net)*.

[57] S. Shiann-Tsong, T. Chen, C. Jenhui, and Y. Fun, "An improved data flushing MAC protocol for IEEE 802.11 wireless ad hoc network," in *Proc. of the 56th IEEE Vehicular Technology Conference (VTC2002-Fall)*, 2002.

[58] D. Kim, C. K. Toh, and Y. Choi, "ROADMAP: a robust ACK-driven media access protocol for mobile ad hoc networks," in *Proc. of IEEE Military Communications Conference (MILCOM2001)*, 2001.

[59] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, 2000.

[60] K.-J. Noh, W.-Y. Choi, and S.-K. Lee, "Adaptive and dynamic tuning of the operation parameter value for QoS and fairness in wireless LAN," in *Proc. of the 60th IEEE Vehicular Technology Conference (VTC2004-Fall)*, 2004.

[61] L. Gannoune, S. Robert, N. Tomar, and T. Agarwal, "Dynamic tuning of the maximum contention window (CWmax) for enhanced service differentiation in IEEE 802.11 wireless ad-hoc networks," in *Proc. of the 60th IEEE Vehicular Technology Conference (VTC2004-Fall)*, 2004.

[62] Y. Xiao, F. H. Li, K. Wu, K. K. Leung, and Q. Ni, "On optimizing backoff counter reservation and classifying stations for the IEEE 802.11 distributed wireless LANs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 7, pp. 713–722, 2006.

[63] F. Cali, M. Conti, and E. Gregori, "Dynamic tuning of the IEEE 802.11 protocol to achieve a theoretical throughput limit," *IEEE/ACM Transactions on Networking*, vol. 8, no. 6, pp. 785–799, 2000.

[64] ——, "IEEE 802.11 wireless LAN: capacity analysis and protocol enhancement," in *Proc. of the 17th IEEE Conference on Computer Communications (INFOCOM1998)*, 1998.

[65] A. L. Toledo, T. Vercauteren, and X. Wang, "Adaptive optimization of IEEE 802.11 DCF based on bayesian estimation of the number of competing terminals," *IEEE Transactions on Mobile Computing*, vol. 5, no. 9, pp. 1283–1296, 2006.

[66] Y. Kwon, Y. Fang, and H. Latchman, "Design of MAC protocols with fast collision resolution for wireless local area networks," *IEEE Transactions on Wireless Communications*, vol. 3, no. 3, pp. 793–807, 2004.

[67] B. Nathan and J. Guo, "Increasing broadcast reliability in vehicular ad hoc networks," in *Proc. of the 3rd international workshop on Vehicular ad hoc networks (VANET)*, 2006.

[68] J. Choi, J. Yoo, S. Choi, and C. Kim, "EBA: an enhancement of the IEEE 802.11 DCF via distributed reservation," *IEEE Transactions on Mobile Computing*, vol. 4, no. 4, pp. 378–390, 2005.

[69] IEEE-802.11, *Part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications, Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements.* IEEE, 2005.

[70] A. Veres, A. T. Campbell, M. Barry, and S. Li-Hsiang, "Supporting service differentiation in wireless packet networks using distributed control," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 10, pp. 2081–2093, 2001.

[71] I. Aad and C. Castelluccia, "Differentiation mechanisms for IEEE 802.11," in *Proc. of the 20th IEEE Conference on Computer Communications (INFOCOM2001)*, 2001.

[72] J. L. Sobrinho and A. S. Krishnakumar, "Real-Time traffic over the IEEE 802.11 medium access control layer," *Bell Labs Tech. J.*, 1996.

[73] M. Demirbas and M. Hussain, "A MAC layer protocol for priority-based reliable multicast in wireless ad hoc networks," in *Proc. of International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM2006)*, 2006.

[74] K. Tang and M. Gerla, "MAC layer broadcast support in 802.11 wireless networks," in *Proc. of IEEE Military Communications Conference (MILCOM2000)*, 2000.

[75] J. Kuri and S. K. Kasera, "Reliable multicast in multi-access wireless LANs," *Wireless Networks*, vol. 7, no. 4, pp. 359–369, 2001.

[76] H. Zhai, J. Wang, X. Chen, and Y. Fang, "Medium access control in mobile ad hoc networks: challenges and solutions," *Wireless Communications and Mobile Computing (Special issue on Ad Hoc Networks)*, 2006.

[77] J. W. Tantra and F. Chuan Heng, "Achieving near maximum throughput in IEEE 802.11 WLANs with contention tone," *IEEE Communications Letters*, vol. 10, no. 9, pp. 658–660, 2006, 1089-7798.

[78] W. Crowther, R. Rettberg, D. Walden, S. Ornstein, and F. Heart, "A system for broadcast communications: reservation ALOHA," in *the Hawaii International Conference on System Sciences*, 1968.

[79] C. S. R. Muthy and B. S. Manoj, *Ad Hoc Wireless Networks, Architectures and Protocols*. Prentice Hall.

[80] G. Bianchi and I. Tinnirello, "Remarks on IEEE 802.11 DCF performance analysis," *IEEE Communications Letters*, vol. 9, no. 8, pp. 765–767, 2005.

[81] X. J. Dong and P. Varaiya, "Saturation throughput analysis of IEEE 802.11 wireless LANs for a lossy channel," *IEEE Communications Letters*, vol. 9, no. 2, pp. 100–102, 2005.

[82] C. H. Foh and J. W. Tantra, "Comments on IEEE 802.11 saturation throughput analysis with freezing of backoff counters," *IEEE Communications Letters*, vol. 9, no. 2, pp. 130–132, 2005.

[83] K. S. Trivedi, *Probability and statistics with reliability, queuing and computer science applications*. John Wiley & Sons, INC., 2002.

[84] H. Su, X. Zhang, and H.-H. Chen, "Cluster-Based DSRC Architecture for QoS Provisioning over Vehicle Ad Hoc Networks," in *Proc. of the 49th IEEE Global Communication Conference (GLOBECOM2006)*, 2006.

[85] FCC, *Amendments regarding DSRC*. FCC, 2004.

[86] X. Chen, H. H. Refai, and X. Ma, "Saturation Performance of IEEE 802.11 Broadcast Scheme in Ad Hoc Wireless LANs," in *Proc. of the 66th IEEE Vehicular Technology Conference (VTC2007-Fall)*, 2007.

[87] O. Dousse, P. Thiran, and M. Hasler, "Connectivity in ad-hoc and hybrid networks," in *Proc. of the 21th IEEE Conference on Computer Communications (INFOCOM2002)*, 2002.

[88] C. H. Foh, G. Liu, B. S. Lee, B.-C. Seet, K.-J. Wong, and C. P. Fu, "Network connectivity of one-dimensional MANETs with random waypoint movement," *IEEE Communications Letters*, vol. 9, no. 1, pp. 31–33, 2005.

[89] H. Zhai, Y. Kwon, and Y. Fang, "Performance analysis of IEEE 802.11 MAC protocols in Wireless LANs," *Wireless Communications and Mobile Computing, John Wiley & Sons, Ltd*, vol. 4, no. 8, pp. 917–931, 2004.

APPENDIX A

Glossary

ABS	Automatic Braking System
ACK	Acknowledgement
ASTM	American American Society for Testing and Materials
BCH	Basic Channel
BEB	Binary Exponential Backoff
BER	Bit Error Rate
BMMM	Batch Mode Multicast MAC
BMW	Broadcast Medium Window
CCA	Cooperative Collision Avoidance
CFP	Consecutive Freeze Process
CTS	Clear To Send
DIFS	DCF Inter-Frame Space
DCF	Distributed Coordination Function
DSRC	Dedicated Short Range Communication
DMAC	Directional MAC
EBA	Early Backoff Announcement
EDCF	Enhanced DCF
EIFS	Extended Inter-Frame Space
FCR	Fast Collision Resolution
FDMA	Frequency Division Multiple Access
FI	Frame Information
FPRP	Five-Phase Reservation Protocol
FHSS	Frequency Hopping Spread Spectrum
HRMA	Hop Reservation Multiple Access

IFS	Inter-Frame Space
IVC	Inter-Vehicle Communication
LAMM	Location Aware Multicast MAC
LBP	Leader Based Protocol
LOS	line Of Sight
MAC	Media Access Control
MRTS	Multicast RTS
NAK	Negative Acknowledgement
NLOS	Non-Line Of Sight
OFDM	Orthogonal Frequency Division Multiplexing
PCF	Point Coordination Function
PDR	Packet Delivery Ratio
PGF	Probability Generating Function
PHY	Physical Layer
PIFS	PCF Inter-Frame Space
QoS	Quality of Service
REQ	Request
RTS	Request To Send
SBP	sequential Backoff Process
SDMA	Space Division Multiple Access
SIFS	Short Inter-Frame Space
TDMA	Time Division Multiple Access
TPC	Transmission Power Control
WLAN	Wireless Local Area Networks
VANET	Vehicular Ad Hoc Network
VSCA	VANET safety Critical Applications

APPENDIX B

MatLab Code Conducting Fast One-hop 802.11 MAC layer broadcasting Simulation

B.1 Main File

```
% main.m
%
% IVC MAC performance simulation system
%
% by Xianbo Chen
%
% Copyright 2006~2007. All right reserved

clear;

TIME_UNIT = 1e-6;           % us

SIM_DURATION=input('simulation duration [3 seconds]? '); if
0==length(SIM_DURATION)
  SIM_DURATION = 3/TIME_UNIT;
else
  SIM_DURATION = SIM_DURATION/TIME_UNIT;
end

NUM_OF_SIM=input('# of simulation repetition [1]? '); if
0==length(NUM_OF_SIM)
  NUM_OF_SIM = 1;
end

TX_RANGE=input('tx range [500 meters]? '); if 0==length(TX_RANGE)
  TX_RANGE = 500;  %meter
end

EXPWAY_LEN=input('express way length [5000 meters]? '); if
```

```
0==length(EXPWAY_LEN)
  EXPWAY_LEN = 5000;
end

TER_DENSITY=input('terminal density [0.02 0.04 0.06 .08 .1 .12 .14
.16 .18 #/meter]? '); if 0==length(TER_DENSITY)
  TER_DENSITY = [0.02 0.04 0.06 .08 .1 .12 .14 .16 .18];
end

P_TRAFFIC_ARRIVAL_RATE=input('periodical packet arrival rate [5
#/second]? '); if 0==length(P_TRAFFIC_ARRIVAL_RATE)
  P_TRAFFIC_ARRIVAL_RATE = [5];
end

E_TRAFFIC_ARRIVAL_RATE=input('emergent packet arrival rate [1
#/second]? '); if 0==length(E_TRAFFIC_ARRIVAL_RATE)
  E_TRAFFIC_ARRIVAL_RATE = [1];
end

PKT_LEN=input('packet length[200 bytes]? '); % 1023 bytes
if 0==length(PKT_LEN)
  PKT_LEN = 200 * 8;         % bit
else
  PKT_LEN = PKT_LEN * 8;    % bit
end

BER=input('bit error rate [0]? '); if 0==length(BER)
  BER = 0;
end

DATA_RATE=input('data rate[24 Mbps]? '); if 0==length(DATA_RATE)
  DATA_RATE = 24* 1e6 * TIME_UNIT;           % bit per us
else
  DATA_RATE = DATA_RATE * 1e6 * TIME_UNIT;           % bit per us
end
```

```
E_CW_L=input('Lower bound of CW size for emergent packets [0]? ');
if 0==length(E_CW_L)
  E_CW_L = 0;
end

E_CW_U=input('Upper bound of CW size for emergent packets [15]? ');
if 0==length(E_CW_U)
  E_CW_U = 15;
end

P_CW_L=input('Lower bound of CW size for periodical packets [16]?
'); if 0==length(P_CW_L)
  P_CW_L = 16;
end

P_CW_U=input('Upper bound of CW size for periodical packets [63]?
'); if 0==length(P_CW_U)
  P_CW_U = 63;
end

PREAMBLE_LEN=input('PHY preamble length (ofdm:40us, fh:96us,
ds:144us) [40 us]? '); if 0==length(PREAMBLE_LEN)
  PREAMBLE_LEN = round(40e-6/TIME_UNIT);
else
  PREAMBLE_LEN = round(PREAMBLE_LEN*1e-6/TIME_UNIT);
end

PLCP_HEADER_LEN=input('PLCP header length (ofdm:8us, fh:32us,
ds:48us) [8 us]? '); if 0==length(PLCP_HEADER_LEN)
  PLCP_HEADER_LEN = round(8e-6/TIME_UNIT);
else
  PLCP_HEADER_LEN = round(PLCP_HEADER_LEN*1e-6/TIME_UNIT);
end

SLOT_TIME=input('slot time (ofdm:16us, fh:50us, ds:20us) [16 us]?');
if 0==length(SLOT_TIME)
```

```
  SLOT_TIME = round(16e-6/TIME_UNIT);
else
  SLOT_TIME = round(SLOT_TIME*1e-6/TIME_UNIT);
end

SIFS_TIME=input('SIFS time (ofdm:32us, fh:28us, ds:10us) [32 us]?
'); if 0==length(SIFS_TIME)
  SIFS_TIME = round(32e-6/TIME_UNIT);
else
  SIFS_TIME = round(SIFS_TIME*1e-6/TIME_UNIT);
end

CW_MAX = 1023;
MAC_HDR_LEN = 272;  % bits

dateAndTime = now;
[outputFilePath, errmsg] = sprintf('./80211bc_%s.txt',
                                   datestr(dateAndTime, 30));
%fid = fopen(outputFilePath,'a');
%fprintf(fid, '\nPARA_LIST_START\n');
%fprintf(fid, 'current_date_time = %s\n', datestr(dateAndTime));
%fprintf(fid, 'time_unit (s) = %e\n', TIME_UNIT);
%fprintf(fid, 'simulation_duartion (time_unit) = %e\n', SIM_DURATION);
%fprintf(fid, 'number_of_simulation = %d\n', NUM_OF_SIM);
%fprintf(fid, 'preamble_len (time_unit) = %d\n', PREAMBLE_LEN);
%fprintf(fid, 'plcp_header_len (time_uint) = %d\n', PLCP_HEADER_LEN);
%fprintf(fid, 'PKT_LEN (bit) = %d\n', PKT_LEN);
%fprintf(fid, 'minimum_contention_window_size = %d\n', CW_MIN);
%fprintf(fid, 'maximum_contention_window_size = %d\n', CW_MAX);
%fprintf(fid, 'slot_time (time_unit) = %d\n', SLOT_TIME);
%fprintf(fid, 'SIFS_time (time_unit) = %d\n', SIFS_TIME);
%fprintf(fid, 'bit_error_rate = %e\n', BER);
%fprintf(fid, 'data_rate (bit/time_unit) = %e\n', DATA_RATE);
%fprintf(fid, 'express_way_len (m) = %d\n', EXPWAY_LEN);
%fprintf(fid, 'transmission_range (m) = %d\n', TX_RANGE);
%fprintf(fid, 'PARA_LIST_END\n');
```

```
%fprintf(fid, '\n\nS\t\tD\t\tST\t\tDR\t\tDensity\t\tarrival_rate\n');
%fclose(fid);

fprintf('\n---------- sim starts ----------\n\n');

fprintf('current_date_time = %s\n', datestr(dateAndTime));
fprintf('time_unit (s) = %e\n', TIME_UNIT);
fprintf('simulation_duartion (time_unit) = %e\n', SIM_DURATION);
fprintf('number_of_simulation = %d\n', NUM_OF_SIM);
fprintf('preamble_len (time_unit) = %d\n', PREAMBLE_LEN);
fprintf('plcp_header_len (time_uint) = %d\n', PLCP_HEADER_LEN);
fprintf('PKT_len (bit) = %d\n', PKT_LEN);
fprintf('e_CW_size_lower_bound = %d\n', E_CW_L);
fprintf('e_CW_size_upper_bound = %d\n', E_CW_U);
fprintf('p_CW_size_lower_bound = %d\n', P_CW_L);
fprintf('p_CW_size_upper_bound = %d\n', P_CW_U);
fprintf('slot_time (time_unit) = %d\n', SLOT_TIME);
fprintf('SIFS_time (time_unit) = %d\n', SIFS_TIME);
fprintf('bit_error_rate = %e\n', BER);
fprintf('data_rate (bit/time_unit) = %e\n', DATA_RATE);
fprintf('express_way_len (m) = %d\n', EXPWAY_LEN);
fprintf('transmission_range (m) = %d\n', TX_RANGE);

%for trafficArrivalRate=TRAFFIC_ARRIVAL_RATE
  for terDensity=TER_DENSITY
    clear perform;

  fprintf('\nterminal_density (#/m) = %f\n', terDensity);
%  fprintf('traffic_arrival_rate (#/s) = %f\n', trafficArrivalRate);

  for i=1:NUM_OF_SIM
    clear phy mac within simPara prtclPerformSumStru

    % resetting of the random table
    rand('state',sum(100*clock));
```

```
%init corresponding protocol
simPara.ber = BER;
simPara.dataRate = DATA_RATE;
simPara.preambleLen = PREAMBLE_LEN;
simPara.plcpHeaderLen = PLCP_HEADER_LEN;
simPara.mpduLen = MAC_HDR_LEN + PKT_LEN;
simPara.sifsTime = SIFS_TIME;
simPara.slotTime = SLOT_TIME;
simPara.txRange = TX_RANGE;
simPara.expWayLen = EXPWAY_LEN;
simPara.eCwLrBd = E_CW_L;
simPara.eCwUpBd = E_CW_U;
simPara.pCwLrBd = P_CW_L;
simPara.pCwUpBd = P_CW_U;
simPara.terDensity = terDensity;  % # per meter

% pkt per second
simPara.pTrafficArrivalRate = P_TRAFFIC_ARRIVAL_RATE;

% pkt per second
simPara.eTrafficArrivalRate = E_TRAFFIC_ARRIVAL_RATE;
simPara.timeUnit = TIME_UNIT;
simPara.simDura = SIM_DURATION;
[phy mac within actualExpWayLen numOfTerminals] =
  satu_bc_80211_init(simPara);

prtclPerformSumStru = satu_bc_80211(phy, mac, within, simPara.simDura);

perform.eS(i) = PKT_LEN / DATA_RATE *
                prtclPerformSumStru.eNumOfGoodPkt /
                prtclPerformSumStru.actualSimDura;
perform.eD(i) = prtclPerformSumStru.eTotalSlotTime /
                prtclPerformSumStru.eNumOfGoodPkt;
perform.eST(i) = prtclPerformSumStru.eTotalSlotTime2 /
                 prtclPerformSumStru.eNumOfGoodPkt;
perform.eDR(i) = prtclPerformSumStru.eNumOfGoodPkt /
```

```
                    prtclPerformSumStru.eTotalPktSent;
    perform.pS(i) = PKT_LEN / DATA_RATE *
                    prtclPerformSumStru.pNumOfGoodPkt /
                    prtclPerformSumStru.actualSimDura;
    perform.pD(i) = prtclPerformSumStru.pTotalSlotTime /
                    prtclPerformSumStru.pNumOfGoodPkt;
    perform.pST(i) = prtclPerformSumStru.pTotalSlotTime2 /
                     prtclPerformSumStru.pNumOfGoodPkt;
    perform.pDR(i) = prtclPerformSumStru.pNumOfGoodPkt /
                     prtclPerformSumStru.pTotalPktSent;

     if 0 == perform.eDR(i)
       prtclPerformSumStru
       plot(phy.status.geoPosition, 2, 'kp');
       pause
     end
   end

   fprintf('\n\neS\t\teD\t\teST\t\teDR\t\t
            Density\t\te_arrival_rate\n');
   fprintf('%f\t%f\t%f\t%f\t%f\t%f\n', mean(perform.eS),
            mean(perform.eD),  ...
            mean(perform.eST), mean(perform.eDR), ...
            terDensity, E_TRAFFIC_ARRIVAL_RATE);
   fprintf('\n\npS\t\tpD\t\tpST\t\tpDR\t\tDensity\t\t
            p_arrival_rate\n');
   fprintf('%f\t%f\t%f\t%f\t%f\t%f\n',
            mean(perform.pS), mean(perform.pD),  ...
            mean(perform.pST), mean(perform.pDR), ...
            terDensity, P_TRAFFIC_ARRIVAL_RATE);

end

fprintf('\n---------- sim ends ----------\n\n');

%graph(outputFilePath);
```

```
return;
```

B.2 Initialization

```
% satu_bc_80211_init.m
%
% saturation broadcast 802.11 initialization
%
% by Xianbo Chen
%
% Copyright 2006~2007. All right reserved

function [phy mac within actualExpWayLen numOfTerminals] =
satu_bc_80211_init(simPara)

phy.attrib.timeUnit = simPara.timeUnit;
phy.attrib.ber = simPara.ber;  % channel bit error rate
phy.attrib.dataRate = simPara.dataRate; % bit per us
phy.attrib.sifs = simPara.sifsTime; phy.attrib.difs =
phy.attrib.sifs + 2*simPara.slotTime; phy.attrib.slotTime =
simPara.slotTime;
phy.attrib.txRange = simPara.txRange;  % meter;
phy.attrib.preambleLen = simPara.preambleLen;
phy.attrib.plcpHeaderLen = simPara.plcpHeaderLen;
phy.attrib.propDelay = 0; %us

% position distribution
gapLenBtwNodes = (1/simPara.terDensity); % meter
numOfTerminals = 1; phy.status.geoPosition(numOfTerminals) =
ceil(-gapLenBtwNodes*log(1-rand));

%while (phy.status.geoPosition(numOfTerminals) < simPara.expWayLen)
while (numOfTerminals < ceil(simPara.expWayLen/gapLenBtwNodes))
  numOfTerminals = numOfTerminals + 1;
```

```
  phy.status.geoPosition(numOfTerminals) =
    phy.status.geoPosition(numOfTerminals-1) +
    ceil(-gapLenBtwNodes*log(1-rand));
end

within = zeros(numOfTerminals, numOfTerminals); for
i=1:numOfTerminals
    for j=i:numOfTerminals
        if (abs(phy.status.geoPosition(i) - phy.status.geoPosition(j)) <=
            phy.attrib.txRange)
            within(i, j) = 1;
            within(j, i) = 1;
        end
    end
%    within(i,i) = 0;
end

actualExpWayLen = phy.status.geoPosition(numOfTerminals);

% calculating sampling area
centerOfExpWay = actualExpWayLen/2;
mac.statistics.lowBoundOfSamplingArea =
  centerOfExpWay - phy.attrib.txRange;
mac.statistics.upBoundOfSamplingArea = centerOfExpWay +
  phy.attrib.txRange;
mac.statistics.minSamplingStaId = 0;

mac.statistics.maxSamplingStaId = 0;

for i=1:numOfTerminals
  if phy.status.geoPosition(i) >=
     mac.statistics.lowBoundOfSamplingArea
    mac.statistics.minSamplingStaId = i;
    break;
  end
end
```

```
for j=mac.statistics.minSamplingStaId:numOfTerminals
  if (j-mac.statistics.minSamplingStaId) >=
      ceil((mac.statistics.upBoundOfSamplingArea -
      mac.statistics.lowBoundOfSamplingArea)/gapLenBtwNodes)
    mac.statistics.maxSamplingStaId = j;
    break;
  end
end

fprintf('\n\n%d\t%d\n\n', mac.statistics.minSamplingStaId,
                          mac.statistics.maxSamplingStaId);

if j==numOfTerminals
  mac.statistics.maxSamplingStaId = numOfTerminals;
end

phy.status.numOfTxSignal = zeros(1, numOfTerminals);
phy.status.signalIntegrity = zeros(1, numOfTerminals);

%mac.attrib.qInterval = (1/simPara.trafficArrivalRate)/phy.attrib.timeUnit;
mac.attrib.pArrivalInterval =
(1/simPara.pTrafficArrivalRate)/phy.attrib.timeUnit;
mac.attrib.eArrivalInterval =
(1/simPara.eTrafficArrivalRate)/phy.attrib.timeUnit;
mac.status.mpduLen = simPara.mpduLen;

mac.attrib.propGoodPkt = (1-phy.attrib.ber)^(mac.status.mpduLen +...
                         round((phy.attrib.preambleLen +
                         phy.attrib.plcpHeaderLen)*simPara.dataRate));
mac.attrib.CwLrBd(1) = simPara.eCwLrBd; mac.attrib.CwUpBd(1) =
simPara.eCwUpBd; mac.attrib.CwLrBd(2) = simPara.pCwLrBd;
mac.attrib.CwUpBd(2) = simPara.pCwUpBd;


mac.status.pPktBornTime = -mac.attrib.pArrivalInterval*log(1-rand(1,
```

```
numOfTerminals)); mac.status.ePktBornTime =
-mac.attrib.eArrivalInterval*log(1-rand(1, numOfTerminals));
mac.status.pktBornTime = min(mac.status.ePktBornTime,
mac.status.pPktBornTime); mac.status.pktType =
(mac.status.ePktBornTime > mac.status.pPktBornTime) + 1;
mac.attrib.PKT_TYPE_E = 1; mac.attrib.PKT_TYPE_P = 2;

%mac.status.backOffSlotCnt = floor(rand(1, numOfTerminals) *
                                   mac.status.contentionWindowSize);
%mac.status.backOffSlotCnt = ceil(rand(1, numOfTerminals) *
                             mac.status.contentionWindowSize +
                             mac.status.pktType*
                             (mac.status.contentionWindowSize+1));
%mac.status.backOffSlotCnt = ceil(rand(1, numOfTerminals) *
                             mac.status.contentionWindowSize);
for i=1:numOfTerminals
  mac.status.backOffSlotCnt(i) = ceil(rand *
                                 (mac.attrib.CwUpBd(mac.status.pktType(i)) -
                                 mac.attrib.CwLrBd(mac.status.pktType(i))) +
                                 mac.attrib.CwLrBd(mac.status.pktType(i)));
end

mac.status.syncTime = mac.status.pktBornTime;
mac.status.pktSrvStartTime = mac.status.pktBornTime;
mac.status.nxtEvtTime = mac.status.syncTime + phy.attrib.difs +
mac.status.backOffSlotCnt * phy.attrib.slotTime;
mac.status.nxtEvtType = zeros(1, numOfTerminals);
mac.status.dsrcTimer = 0;

mac.statistics.eNumOfPktSent = zeros(1, numOfTerminals);
mac.statistics.eNumOfGoodPkt = zeros(1, numOfTerminals);
mac.statistics.eTotalSlotTime = zeros(1, numOfTerminals);
mac.statistics.eTotalSlotTime2 = zeros(1, numOfTerminals);
mac.statistics.pNumOfPktSent = zeros(1, numOfTerminals);
mac.statistics.pNumOfGoodPkt = zeros(1, numOfTerminals);
mac.statistics.pTotalSlotTime = zeros(1, numOfTerminals);
```

```
mac.statistics.pTotalSlotTime2 = zeros(1, numOfTerminals);

return;
```

B.3 Protocol Processing

```
% satu_bc_80211.m
%
% 802.11 saturation broadcast simulation function
%
%
% by Xianbo Chen
%
% Copyright 2006~2007. All right reserved

function prtclPerformSumStru = satu_bc_80211(phy, mac, within,
simDuration)

percentage = 1; s = '';

while (mac.status.dsrcTimer < simDuration)
  if percentage == floor(100*mac.status.dsrcTimer/simDuration)
    for j=1:length(s)
      fprintf('\b');
    end
    [s, errmsg] = sprintf('%%%d finished', percentage);
    fprintf('%s', s);
    percentage = percentage + 1;
  end

  nxtEvtTime = min(mac.status.nxtEvtTime);
  mac.status.dsrcTimer = nxtEvtTime;

  % deal with evt_end_tx:1 first, then evt_start_tx:0
  idxNext2EndTx = find(mac.status.nxtEvtTime == nxtEvtTime
```

```
                    & 1 == mac.status.nxtEvtType);
for i = 1 : length(idxNext2EndTx)
  terId = idxNext2EndTx(i);

  % according to varibles within and nxtEvtType to decide
  %  the reaction of each terminal within the range to
  %  this event.
  IdxNeighbor = find(1 == within(terId, :));

  % if it has no neighbors, do nothing
  if ~isempty(IdxNeighbor)
    if mac.attrib.PKT_TYPE_E == mac.status.pktType(terId)
      mac.statistics.eNumOfPktSent(terId) =
       mac.statistics.eNumOfPktSent(terId) + 1;
    else
      mac.statistics.pNumOfPktSent(terId) =
       mac.statistics.pNumOfPktSent(terId) + 1;
    end

    phy.status.numOfTxSignal(IdxNeighbor) =
     phy.status.numOfTxSignal(IdxNeighbor) - 1;

    if (all(0==phy.status.signalIntegrity(IdxNeighbor))
     & all(0==phy.status.numOfTxSignal(IdxNeighbor)))
      % no collision, this tx is a good one, do statistics
      if rand < mac.attrib.propGoodPkt
        if mac.attrib.PKT_TYPE_E == mac.status.pktType(terId)
          mac.statistics.eTotalSlotTime2(terId) =
            mac.statistics.eTotalSlotTime2(terId) +
            mac.status.dsrcTimer -
            mac.status.pktSrvStartTime(terId);
          mac.statistics.eTotalSlotTime(terId) =
            mac.statistics.eTotalSlotTime(terId) +
            mac.status.dsrcTimer - mac.status.pktBornTime(terId);
          mac.statistics.eNumOfGoodPkt(terId) =
            mac.statistics.eNumOfGoodPkt(terId) + 1;
```

```
    else
      mac.statistics.pTotalSlotTime2(terId) =
        mac.statistics.pTotalSlotTime2(terId) +
        mac.status.dsrcTimer -
        mac.status.pktSrvStartTime(terId);
      mac.statistics.pTotalSlotTime(terId) =
        mac.statistics.pTotalSlotTime(terId) +
        mac.status.dsrcTimer -
        mac.status.pktBornTime(terId);
      mac.statistics.pNumOfGoodPkt(terId) =
        mac.statistics.pNumOfGoodPkt(terId) + 1;
    end
  end

  % medium idle detected by all its neighoring terminals,
  %  next event for these terminals is to send if any
  %  packet ready.
  tmpIdx = mac.status.dsrcTimer >=
    mac.status.pktBornTime(IdxNeighbor);
  mac.status.syncTime(IdxNeighbor(tmpIdx)) =
    mac.status.dsrcTimer;
  tmpIdx = mac.status.dsrcTimer <
    mac.status.pktBornTime(IdxNeighbor);
  mac.status.syncTime(IdxNeighbor(tmpIdx)) =
    mac.status.pktBornTime(IdxNeighbor(tmpIdx));
  mac.status.nxtEvtType(IdxNeighbor) = 0;
  mac.status.nxtEvtTime(IdxNeighbor) =
    mac.status.syncTime(IdxNeighbor) +
    phy.attrib.difs +
    mac.status.backOffSlotCnt(IdxNeighbor) *
    phy.attrib.slotTime;
else
  for j = 1 : length(IdxNeighbor)
    neighborId = IdxNeighbor(j);
    if (0 == phy.status.numOfTxSignal(neighborId))
      if (0 ~= phy.status.signalIntegrity(neighborId))
```

```
            phy.status.signalIntegrity(neighborId) = 0;
          end

          % medium idle detected by this neighoring terminal,
          %  next event for this terminal is to send if any
          %  packet ready
          if mac.status.dsrcTimer >=
            mac.status.pktBornTime(neighborId)
            mac.status.syncTime(neighborId) =
              mac.status.dsrcTimer;
          else
            mac.status.syncTime(neighborId) =
              mac.status.pktBornTime(neighborId);
          end

          mac.status.nxtEvtType(neighborId) = 0;
          mac.status.nxtEvtTime(neighborId) =
            mac.status.syncTime(neighborId) +
            phy.attrib.difs +
            mac.status.backOffSlotCnt(neighborId) *
            phy.attrib.slotTime;
        end
      end
    end

    mac.status.nxtEvtType(terId) = 0;
%    mac.status.pktBornTime(terId) =
%      mac.status.pktBornTime(terId) -
%      mac.attrib.qInterval*log(1-rand);
%%   for saturation
%    mac.status.pktBornTime(terId) = mac.status.dsrcTimer;
    if mac.attrib.PKT_TYPE_E == mac.status.pktType(terId)
      mac.status.ePktBornTime(terId) =
        mac.status.ePktBornTime(terId) -
        mac.attrib.eArrivalInterval*log(1-rand);
```

```
    else
      mac.status.pPktBornTime(terId) =
      mac.status.pPktBornTime(terId) -
      mac.attrib.pArrivalInterval*log(1-rand);
    end

    if mac.status.ePktBornTime(terId) <=
      mac.status.dsrcTimer
      mac.status.pktBornTime(terId) =
        mac.status.ePktBornTime(terId);
      mac.status.pktType(terId) = 1;
    else
      mac.status.pktBornTime(terId) =
        min(mac.status.ePktBornTime(terId),
            mac.status.pPktBornTime(terId));
      mac.status.pktType(terId) =
        (mac.status.ePktBornTime(terId) >
         mac.status.pPktBornTime(terId)) + 1;
    end

%    mac.status.backOffSlotCnt(terId) =
%      floor(rand * mac.status.contentionWindowSize); % CFP
%    mac.status.backOffSlotCnt(terId) =
%      ceil(rand * mac.status.contentionWindowSize);
    mac.status.backOffSlotCnt(terId) = ceil(rand *
      (mac.attrib.CwUpBd(mac.status.pktType(terId)) -
      mac.attrib.CwLrBd(mac.status.pktType(terId)))
      + mac.attrib.CwLrBd(mac.status.pktType(terId)));

    if mac.status.dsrcTimer >= mac.status.pktBornTime(terId)
      mac.status.syncTime(terId) = mac.status.dsrcTimer;
    else
      mac.status.syncTime(terId) = mac.status.pktBornTime(terId);
    end

    mac.status.pktSrvStartTime(terId) = mac.status.syncTime(terId);
```

```
    mac.status.nxtEvtTime(terId) =mac.status.syncTime(terId) +
      phy.attrib.difs +
      mac.status.backOffSlotCnt(terId) * phy.attrib.slotTime;
  end

  % propagation delay might be considered.
  idxNext2StartTx = find(mac.status.nxtEvtTime <=
    (nxtEvtTime+phy.attrib.propDelay) &
    0 == mac.status.nxtEvtType);
  for i = 1 : length(idxNext2StartTx)
    terId = idxNext2StartTx(i);

    % according to varibles within and nxtEvtType to decide
    %  the reaction of each terminal within the range to
    %  this event.
    IdxNeighbor = find(1 == within(terId, :));
    phy.status.numOfTxSignal(IdxNeighbor) =
      phy.status.numOfTxSignal(IdxNeighbor) + 1;

    for j = 1 : length(IdxNeighbor)
      neighborId = IdxNeighbor(j);
      if (0 == phy.status.signalIntegrity(neighborId))
        if (2 <= phy.status.numOfTxSignal(neighborId))
          % signal interference, integrity destroyed
          phy.status.signalIntegrity(neighborId) = 1;
        end

%       % not in the tx idx
%       if (0 == length(find(neighborId == idxNext2StartTx))) &
%        (Inf ~= mac.status.nxtEvtType(neighborId))
        if (isempty(find(neighborId == idxNext2StartTx))) &
          (Inf ~= mac.status.nxtEvtType(neighborId))
          % deal with the possible backoff counter decreament
          if (0 == mac.status.nxtEvtType(neighborId))
            numOfSlotPassed = floor((mac.status.dsrcTimer -
              mac.status.syncTime(neighborId) -
```

```
              phy.attrib.difs)/phy.attrib.slotTime);
            if (numOfSlotPassed > 0)
              mac.status.backOffSlotCnt(neighborId) =
                mac.status.backOffSlotCnt(neighborId) -
                numOfSlotPassed;
            end
          end

          if (1 == mac.status.nxtEvtType(neighborId))
            wrong;
          end

          mac.status.nxtEvtType(neighborId) = Inf;
          mac.status.nxtEvtTime(neighborId) = Inf;
        end
      end
    end

    mac.status.nxtEvtType(terId) = 1;
    mac.status.nxtEvtTime(terId) = mac.status.nxtEvtTime(terId) +
      mac.status.mpduLen/phy.attrib.dataRate +
      phy.attrib.preambleLen + phy.attrib.plcpHeaderLen;
  end
end

%for i=1:length(mac.statistics.numOfPktSent)
%    while mac.status.pktBornTime(i) < mac.status.dsrcTimer
%        mac.status.pktBornTime(i) =
%          mac.status.pktBornTime(terId) -
%          mac.attrib.qInterval*log(1-rand);
%        mac.statistics.numOfPktSent(i) =
%          mac.statistics.numOfPktSent(i);
%    end
%end

prtclPerformSumStru.eTotalPktSent =
```

```
  sum(mac.statistics.eNumOfPktSent(
  mac.statistics.minSamplingStaId:
  mac.statistics.maxSamplingStaId));
prtclPerformSumStru.eTotalSlotTime =
  sum(mac.statistics.eTotalSlotTime(
  mac.statistics.minSamplingStaId:
  mac.statistics.maxSamplingStaId));
prtclPerformSumStru.eTotalSlotTime2 =
  sum(mac.statistics.eTotalSlotTime2(
  mac.statistics.minSamplingStaId:
  mac.statistics.maxSamplingStaId));
prtclPerformSumStru.eNumOfGoodPkt =
  sum(mac.statistics.eNumOfGoodPkt(
  mac.statistics.minSamplingStaId:
  mac.statistics.maxSamplingStaId));
prtclPerformSumStru.pTotalPktSent =
  sum(mac.statistics.pNumOfPktSent(
  mac.statistics.minSamplingStaId:
  mac.statistics.maxSamplingStaId));
prtclPerformSumStru.pTotalSlotTime =
  sum(mac.statistics.pTotalSlotTime(
  mac.statistics.minSamplingStaId:
  mac.statistics.maxSamplingStaId));
prtclPerformSumStru.pTotalSlotTime2 =
  sum(mac.statistics.pTotalSlotTime2(
  mac.statistics.minSamplingStaId:
  mac.statistics.maxSamplingStaId));
prtclPerformSumStru.pNumOfGoodPkt =
  sum(mac.statistics.pNumOfGoodPkt(
  mac.statistics.minSamplingStaId:
  mac.statistics.maxSamplingStaId));
prtclPerformSumStru.actualSimDura = mac.status.dsrcTimer;

%totalPkt1to25 = sum(mac.statistics.numOfPktSent(1:25));
%goodPkt1to25 = sum(mac.statistics.numOfGoodPkt(1:25));
%fprintf('\n\nS1~25=%f \t\t D1~25=%f\n', 200 * 8 /
```

```
%  phy.attrib.dataRate
%  * goodPkt1to25 / prtclPerformSumStru.actualSimDura,
%  goodPkt1to25 / totalPkt1to25);

return ;
```

APPENDIX C

MatLab Code Conducting Event-driven 802.11 MAC layer Simulation

```
% ignore the fact that a station may not backoff when
%  transmitting its first packet after power-up becuase
%  it may achieve its first DIFS without finding the
%  medium was ever busy.

% propagation delay is ignored

function ao2dot11

clear;

% *** MAC state mechine
MAC_STATE_IDLE              = 0;

MAC_STATE_WAIT_FOR_NAV      = 1;

MAC_STATE_WAIT_FOR_DIFS     = 2;

MAC_STATE_BACK_OFF          = 3;

MAC_STATE_TX_BC             = 4;

MAC_STATE_TX_UC             = 5;

MAC_STATE_SIFS_AFTER_TX_UC  = 6;

MAC_STATE_WAIT_FOR_ACK      = 7;

MAC_STATE_RX_ACK            = 8;

MAC_STATE_TX_RTS            = 9;
```

```
MAC_STATE_WAIT_FOR_CTS      = 10;

MAC_STATE_TX_CTS            = 11;

MAC_STATE_WAIT_FOR_DATA     = 12;

MAC_STATE_SIFS_AFTER_TX_RTS = 13;

MAC_STATE_RX_CTS            = 14;

MAC_STATE_RX_UC             = -1;

MAC_STATE_SIFS_AFTER_RX_UC  = -2;

MAC_STATE_TX_ACK            = -3;

MAC_STATE_RX_BC             = -4;

MAC_STATE_RX_RTS            = -5;

MAC_STATE_SIFS_AFTER_RX_RTS = -6;

MAC_STATE_SIFS_AFTER_TX_CTS = -7;

MAC_STATE_SIFS_AFTER_RX_CTS = -8;


% *** message type
MSG_TP_NULL                  = 0;

MSG_TP_TRANSIT_2_IDLE        = 1;

MSG_TP_SOMEONE_TX            = 2;

MSG_TP_SOMEONE_STOP_TX       = 3;
```

```
MSG_TP_DIFS_ACHEIVED         = 4;

MSG_TP_TX_UC_DATA_START      = 5;

MSG_TP_TX_UC_DATA_END        = 6;

MSG_TP_READY_TO_RX_ACK       = 7;

MSG_TP_ACK_TIME_OUT          = 8;

MSG_TP_RX_ACK_START          = 9;

MSG_TP_RX_ACK_END            = 10;

MSG_TP_RX_UC_DATA_START      = 11;

MSG_TP_RX_UC_DATA_END        = 12;

MSG_TP_TX_ACK_START          = 13;

MSG_TP_TX_ACK_END            = 14;

MSG_TP_TX_BC_DATA_START      = 15;

MSG_TP_TX_BC_DATA_END        = 16;

MSG_TP_RX_BC_DATA_START      = 17;

MSG_TP_RX_BC_DATA_END        = 18;

MSG_TP_TX_RTS_START          = 19;

MSG_TP_RX_RTS_START          = 20;

MSG_TP_TX_RTS_END            = 21;
```

```
MSG_TP_RX_RTS_END            = 22;

MSG_TP_TX_CTS_START          = 23;

MSG_TP_RX_CTS_START          = 24;

MSG_TP_TX_CTS_END            = 25;

MSG_TP_RX_CTS_END            = 26;

MSG_TP_READY_TO_RX_CTS       = 27;

MSG_TP_CTS_TIME_OUT          = 28;

MSG_TP_READY_TO_RECEIVE_DATA = 29;

% *** event type
EVT_TP_TRANSIT_2_IDLE        = MSG_TP_TRANSIT_2_IDLE;

EVT_TP_DIFS_ACHEIVED         = MSG_TP_DIFS_ACHEIVED;

EVT_TP_UC_TX_DATA_START      = MSG_TP_TX_UC_DATA_START;

EVT_TP_UC_TX_DATA_END        = MSG_TP_TX_UC_DATA_END;

EVT_TP_READY_TO_RX_ACK       = MSG_TP_READY_TO_RX_ACK;

EVT_TP_ACK_TIME_OUT          = MSG_TP_ACK_TIME_OUT;

EVT_TP_TX_ACK_START          = MSG_TP_TX_ACK_START;

EVT_TP_TX_ACK_END            = MSG_TP_TX_ACK_END;

EVT_TP_BC_TX_DATA_START      = MSG_TP_TX_BC_DATA_START;

EVT_TP_BC_TX_DATA_END        = MSG_TP_TX_BC_DATA_END;
```

```
EVT_TP_READY_TO_RX_CTS       = MSG_TP_READY_TO_RX_CTS;

EVT_TP_TX_RTS_START          = MSG_TP_TX_RTS_START;

EVT_TP_TX_RTS_END            = MSG_TP_TX_RTS_END;

EVT_TP_CTS_TIME_OUT          = MSG_TP_CTS_TIME_OUT;

EVT_TP_TX_CTS_START          = MSG_TP_TX_CTS_START;

EVT_TP_TX_CTS_END            = MSG_TP_TX_CTS_END;

EVT_TP_READY_TO_RECEIVE_DATA = MSG_TP_READY_TO_RECEIVE_DATA;

% *** packet type
PKT_TP_UC_BAS = 0;    % unicast with basic access mechanism

PKT_TP_BC     = 1;    % broadcast

PKT_TP_UC_RTS = 2;    % unicast with RTS/CTS

% *** other constant
MIN_NUM_OF_STA   = 5;

NUM_OF_STA_STEP  = 5;

MAX_NUM_OF_STA   = 50;

TIME_UNIT        = 1e-6;                    % micro second

SIM_DURATION     = 100 / TIME_UNIT;         % TIME_UNIT

DATA_RATE        = 1e6;                     % bps

SLOT_SIZE        = 50;                      % TIME_UNIT
```

```
SIFS             = 28;                        % TIME_UNIT

DIFS             = SIFS + 2 * SLOT_SIZE;      % TIME_UNIT

MIN_CW_STAGE     = 7;

MAX_CW_STAGE     = 10;

ACK_TIMEOUT      = 300;                       % ACK TIME_UNIT

CTS_TIMEOUT      = 300;                       % CTS TIME_UNIT

DATA_PAYLOAD_LEN = 8184;                      % bit

MAC_HEADER_LEN   = 272;                       % bit

PHY_HEADER_LEN   = 128;                       % bit

DATA_PKT_LEN     = DATA_PAYLOAD_LEN + ...
                   MAC_HEADER_LEN + ...
                   PHY_HEADER_LEN;            % bit

ACK_PKT_LEN      = 112 + PHY_HEADER_LEN;      % bit

RTS_PKT_LEN      = 160 + PHY_HEADER_LEN;      % bit

CTS_PKT_LEN      = 112 + PHY_HEADER_LEN;      % bit

numOfLoop        = 0;

% & execution &
for numOfSta = MIN_NUM_OF_STA : NUM_OF_STA_STEP : MAX_NUM_OF_STA
  % *** parameters initialization
  curTime = 0; % TIME_UNIT
```

```
% *** event scheduler initialization
for staId = 1 : numOfSta
  evt.occurTime(staId) = curTime;
  evt.evtType(staId)   = EVT_TP_TRANSIT_2_IDLE;
  evt.para1(staId)     = 0;
end

% *** message box initializaion
msgP1Counter        = 0;
msgP1.staId         = repmat(0, 1, numOfSta);
msgP1.msgType       = repmat(MSG_TP_NULL, 1, numOfSta);
msgP1.para1         = repmat(0, 1, numOfSta);
msgP2Counter        = 0;
msgP2.staId(1)      = 0;
msgP2.msgType(1)    = MSG_TP_NULL;
msgP2.para1(1)      = 0;

% *** station (sta) initialization
for staId = 1 : numOfSta
  sta.phy.status.medState(staId)        = 0;
  sta.mac.status.ucpktBornTime(staId)   = 0;
  sta.mac.status.bcpktBornTime(staId)   = 0;
  sta.mac.status.bcTxStatus(staId)      = 0;
  sta.mac.status.state(staId)           = MAC_STATE_IDLE;
  sta.mac.status.bakOffCnter(staId)     = 0;
  sta.mac.status.bakOffStartTime(staId) = 0;
  sta.mac.status.cwStage(staId)         = MIN_CW_STAGE;

  tempVar = staId;
  while tempVar == staId
    tempVar = ceil(rand * numOfSta);
  end

  sta.mac.status.dstStaId(staId) = tempVar;
  sta.mac.status.srcStaId(staId) = 0;
  sta.mac.status.pktType(staId) = PKT_TP_UC_RTS;
```

```
  sta.statistics.NumOfUcpktBasSentSuccessfully(staId) = 0;
  sta.statistics.ucpktBasTotalDelay(staId) = 0;
  sta.statistics.NumOfBcpktSentSuccessfully(staId) = 0;
  sta.statistics.bcpktTotalDelay(staId) = 0;
  sta.statistics.NumOfUcpktRtsSentSuccessfully(staId) = 0;
  sta.statistics.ucpktRtsTotalDelay(staId) = 0;
end

% *** run
while (curTime <= SIM_DURATION)
  % event handling, jump to the nearest time point
  nearestTimePoint = min(evt.occurTime);

  if (inf == nearestTimePoint)
    fprintf('WRONG, nearest time point is inf');
    pause;
  else
    staIdIdx = find(nearestTimePoint == evt.occurTime);
  end

  % notify corresponding stations the event by messaging
  for i = 1 : length(staIdIdx)
    staId = staIdIdx(i);
    para1 = evt.para1(staId);
    switch evt.evtType(staId)
      case EVT_TP_TRANSIT_2_IDLE
        msgP1Counter = msgP1Counter + 1;
        msgP1.staId(msgP1Counter) = staId;
        msgP1.msgType(msgP1Counter) =
          MSG_TP_TRANSIT_2_IDLE;

      case EVT_TP_DIFS_ACHEIVED
        msgP1Counter = msgP1Counter + 1;
        msgP1.staId(msgP1Counter) = staId;
        msgP1.msgType(msgP1Counter) =
          MSG_TP_DIFS_ACHEIVED;
```

```
case EVT_TP_UC_TX_DATA_START
  msgP1Counter = msgP1Counter + 1;
  msgP1.staId(msgP1Counter) = staId;
  msgP1.msgType(msgP1Counter) =
    MSG_TP_TX_UC_DATA_START;
  tempVar = msgP2Counter + 1;
  msgP2Counter = msgP2Counter + numOfSta - 1;
  msgP2.staId(tempVar:msgP2Counter) =
    [1:(staId-1) (staId+1):numOfSta];
  msgP2.msgType(tempVar:msgP2Counter) =
    MSG_TP_SOMEONE_TX;

  % in order to align
  msgP2.para1(tempVar:msgP2Counter) = staId;
  if para1<staId
    msgP2.msgType(tempVar + para1 - 1) =
      MSG_TP_RX_UC_DATA_START;
  else
    msgP2.msgType(tempVar + para1 - 2) =
      MSG_TP_RX_UC_DATA_START;
  end

case EVT_TP_UC_TX_DATA_END
  msgP1Counter = msgP1Counter + 1;
  msgP1.staId(msgP1Counter) = staId;
  msgP1.msgType(msgP1Counter) =
    MSG_TP_TX_UC_DATA_END;
  tempVar = msgP2Counter + 1;
  msgP2Counter = msgP2Counter + numOfSta - 1;
  msgP2.staId(tempVar:msgP2Counter) =
    [1:(staId-1) (staId+1):numOfSta];
  msgP2.msgType(tempVar:msgP2Counter) =
    MSG_TP_SOMEONE_STOP_TX;

  % in order to align
```

```
  msgP2.para1(tempVar:msgP2Counter) = staId;
  if para1<staId
    msgP2.msgType(tempVar + para1 - 1) =
      MSG_TP_RX_UC_DATA_END;
    else
      msgP2.msgType(tempVar + para1 - 2) =
        MSG_TP_RX_UC_DATA_END;
    end

case EVT_TP_READY_TO_RX_ACK
  msgP1Counter = msgP1Counter + 1;
  msgP1.staId(msgP1Counter) = staId;
  msgP1.msgType(msgP1Counter) =
    MSG_TP_READY_TO_RX_ACK;

case EVT_TP_ACK_TIME_OUT
  msgP1Counter = msgP1Counter + 1;
  msgP1.staId(msgP1Counter) = staId;
  msgP1.msgType(msgP1Counter) =
    MSG_TP_ACK_TIME_OUT;

case EVT_TP_TX_ACK_START
  msgP1Counter = msgP1Counter + 1;
  msgP1.staId(msgP1Counter) = staId;
  msgP1.msgType(msgP1Counter) =
    MSG_TP_TX_ACK_START;
  tempVar = msgP2Counter + 1;
  msgP2Counter = msgP2Counter + numOfSta - 1;
  msgP2.staId(tempVar:msgP2Counter) =
    [1:(staId-1) (staId+1):numOfSta];
  msgP2.msgType(tempVar:msgP2Counter) =
    MSG_TP_SOMEONE_TX;

  % in order to align
  msgP2.para1(tempVar:msgP2Counter) = staId;
  if para1<staId
```

```
    msgP2.msgType(tempVar + para1 - 1) =
      MSG_TP_RX_ACK_START;
  else
    msgP2.msgType(tempVar + para1 - 2) =
      MSG_TP_RX_ACK_START;
  end

case EVT_TP_TX_ACK_END
  msgP1Counter = msgP1Counter + 1;
  msgP1.staId(msgP1Counter) = staId;
  msgP1.msgType(msgP1Counter) =
    MSG_TP_TX_ACK_END;
  tempVar = msgP2Counter + 1;
  msgP2Counter = msgP2Counter + numOfSta - 1;
  msgP2.staId(tempVar:msgP2Counter) =
    [1:(staId-1) (staId+1):numOfSta];
  msgP2.msgType(tempVar:msgP2Counter) =
    MSG_TP_SOMEONE_STOP_TX;

  % in order to align
  msgP2.para1(tempVar:msgP2Counter) = staId;
    if para1<staId
      msgP2.msgType(tempVar + para1 - 1) =
        MSG_TP_RX_ACK_END;
    else
      msgP2.msgType(tempVar + para1 - 2) =
        MSG_TP_RX_ACK_END;
    end

case EVT_TP_BC_TX_DATA_START
  msgP1Counter = msgP1Counter + 1;
  msgP1.staId(msgP1Counter) = staId;
  msgP1.msgType(msgP1Counter) =
    MSG_TP_TX_BC_DATA_START;
  tempVar = msgP2Counter + 1;
  msgP2Counter = msgP2Counter + numOfSta - 1;
```

```
  msgP2.staId(tempVar:msgP2Counter) =
    [1:(staId-1) (staId+1):numOfSta];
  msgP2.msgType(tempVar:msgP2Counter) =
    MSG_TP_RX_BC_DATA_START;
  msgP2.para1(tempVar:msgP2Counter) = staId;

case EVT_TP_BC_TX_DATA_END
  msgP1Counter = msgP1Counter + 1;
  msgP1.staId(msgP1Counter) = staId;
  msgP1.msgType(msgP1Counter) =
    MSG_TP_TX_BC_DATA_END;
  tempVar = msgP2Counter + 1;
  msgP2Counter = msgP2Counter + numOfSta - 1;
  msgP2.staId(tempVar:msgP2Counter) =
    [1:(staId-1) (staId+1):numOfSta];
  msgP2.msgType(tempVar:msgP2Counter) =
    MSG_TP_RX_BC_DATA_END;
  msgP2.para1(tempVar:msgP2Counter) = staId;

case EVT_TP_TX_RTS_START
  msgP1Counter = msgP1Counter + 1;
  msgP1.staId(msgP1Counter) = staId;
  msgP1.msgType(msgP1Counter) =
    MSG_TP_TX_RTS_START;
  tempVar = msgP2Counter + 1;
  msgP2Counter = msgP2Counter + numOfSta - 1;
  msgP2.staId(tempVar:msgP2Counter) =
    [1:(staId-1) (staId+1):numOfSta];
  msgP2.msgType(tempVar:msgP2Counter) =
    MSG_TP_SOMEONE_TX;

  % in order to align
  msgP2.para1(tempVar:msgP2Counter) = staId;

  if para1<staId
    msgP2.msgType(tempVar + para1 - 1) =
```

```
      MSG_TP_RX_RTS_START;
    else
      msgP2.msgType(tempVar + para1 - 2) =
        MSG_TP_RX_RTS_START;
    end

case EVT_TP_TX_RTS_END
  msgP1Counter = msgP1Counter + 1;
  msgP1.staId(msgP1Counter) = staId;
  msgP1.msgType(msgP1Counter) =
    MSG_TP_TX_RTS_END;
  tempVar = msgP2Counter + 1;
  msgP2Counter = msgP2Counter + numOfSta - 1;
  msgP2.staId(tempVar:msgP2Counter) =
    [1:(staId-1) (staId+1):numOfSta];
  msgP2.msgType(tempVar:msgP2Counter) =
    MSG_TP_SOMEONE_STOP_TX;

  % in order to align
  msgP2.para1(tempVar:msgP2Counter) = staId;
  if para1<staId
    msgP2.msgType(tempVar + para1 - 1) =
    MSG_TP_RX_RTS_END;
  else
    msgP2.msgType(tempVar + para1 - 2) =
      MSG_TP_RX_RTS_END;
  end

case EVT_TP_READY_TO_RX_CTS
  msgP1Counter = msgP1Counter + 1;
  msgP1.staId(msgP1Counter) = staId;
  msgP1.msgType(msgP1Counter) =
    MSG_TP_READY_TO_RX_CTS;

case EVT_TP_CTS_TIME_OUT
  msgP1Counter = msgP1Counter + 1;
```

```
  msgP1.staId(msgP1Counter) = staId;
  msgP1.msgType(msgP1Counter) =
    MSG_TP_CTS_TIME_OUT;

case EVT_TP_TX_CTS_START
  msgP1Counter = msgP1Counter + 1;
  msgP1.staId(msgP1Counter) = staId;
  msgP1.msgType(msgP1Counter) =
    MSG_TP_TX_CTS_START;

  tempVar = msgP2Counter + 1;
  msgP2Counter = msgP2Counter + numOfSta - 1;
  msgP2.staId(tempVar:msgP2Counter) =
    [1:(staId-1) (staId+1):numOfSta];
  msgP2.msgType(tempVar:msgP2Counter) =
    MSG_TP_SOMEONE_TX;

  % in order to align
  msgP2.para1(tempVar:msgP2Counter) = staId;
  if para1<staId
    msgP2.msgType(tempVar + para1 - 1) =
      MSG_TP_RX_CTS_START;
  else
    msgP2.msgType(tempVar + para1 - 2) =
      MSG_TP_RX_CTS_START;
  end

case EVT_TP_TX_CTS_END
  msgP1Counter = msgP1Counter + 1;
  msgP1.staId(msgP1Counter) = staId;
  msgP1.msgType(msgP1Counter) =
    MSG_TP_TX_CTS_END;
  tempVar = msgP2Counter + 1;
    msgP2Counter = msgP2Counter + numOfSta - 1;
    msgP2.staId(tempVar:msgP2Counter) =
      [1:(staId-1) (staId+1):numOfSta];
```

```
        msgP2.msgType(tempVar:msgP2Counter) =
          MSG_TP_SOMEONE_STOP_TX;

      % in order to align
      msgP2.para1(tempVar:msgP2Counter) = staId;
      if para1<staId
        msgP2.msgType(tempVar + para1 - 1) =
          MSG_TP_RX_CTS_END;
        else
          msgP2.msgType(tempVar + para1 - 2) =
            MSG_TP_RX_CTS_END;
        end

    case EVT_TP_READY_TO_RECEIVE_DATA
      msgP1Counter = msgP1Counter + 1;
      msgP1.staId(msgP1Counter) = staId;
      msgP1.msgType(msgP1Counter) =
        MSG_TP_READY_TO_RECEIVE_DATA;

    otherwise
      fprintf('unknown event %d on sta # %d\n',
               evt.evtType(staId), staId);
      pause;
  end % end of "swtich evt.evtType(staId)"
end % end of "for i = 1 : length(staIdIdx)"

evt.occurTime(staIdIdx) = inf;     % clear event
curTime = nearestTimePoint;

% message processing
msgCounter = msgP1Counter + msgP2Counter;
msg.staId   = [msgP1.staId(1:msgP1Counter)
               msgP2.staId(1:msgP2Counter)];
msg.msgType = [msgP1.msgType(1:msgP1Counter)
               msgP2.msgType(1:msgP2Counter)];
msg.para1   = [msgP1.para1(1:msgP1Counter)
```

```
                msgP2.para1(1:msgP2Counter)];
msgP1Counter = msgP1Counter - msgP1Counter; % clear
msgP2Counter = msgP2Counter - msgP2Counter; % clear

for i = 1 : msgCounter
  staId = msg.staId(i);
  msgType = msg.msgType(i);
  para1 = msg.para1(i);

  if MSG_TP_SOMEONE_TX == msgType ||
     MSG_TP_RX_UC_DATA_START == msgType ||
     MSG_TP_RX_ACK_START == msgType ||
     MSG_TP_RX_BC_DATA_START == msgType || ...
     MSG_TP_RX_RTS_START == msgType ||
     MSG_TP_RX_CTS_START == msgType
    sta.phy.status.medState(staId) =
      sta.phy.status.medState(staId) + 1;
  end

  if MSG_TP_SOMEONE_STOP_TX == msgType ||
     MSG_TP_RX_UC_DATA_END == msgType ||
     MSG_TP_RX_ACK_END == msgType ||
     MSG_TP_RX_BC_DATA_END == msgType || ...
     MSG_TP_RX_RTS_END == msgType ||
     MSG_TP_RX_CTS_END == msgType
    sta.phy.status.medState(staId) =
      sta.phy.status.medState(staId) - 1;
  end

  switch sta.mac.status.state(staId)
    case MAC_STATE_IDLE
      switch msgType
        case MSG_TP_TRANSIT_2_IDLE
          sta.mac.status.state(staId) =
            MAC_STATE_WAIT_FOR_DIFS;
          evt.occurTime(staId) =
```

```
        curTime + DIFS;
      evt.evtType(staId) =
        EVT_TP_DIFS_ACHEIVED;  % set event
    otherwise
      fprintf('wrong msg %d with para %d
              in state %d on sta # %d\n',
              msgType, para1,
              sta.mac.status.state(staId),
              staId);
      pause;
  end

case MAC_STATE_WAIT_FOR_NAV
  switch msgType
    case MSG_TP_SOMEONE_TX
      %fprintf('Ignore message %d from
               station %d when I %d am
               in state %d\n', msgType,
               para1, staId,
               sta.mac.status.state(staId));
    case MSG_TP_SOMEONE_STOP_TX
      if 0 == sta.phy.status.medState(staId)
          sta.mac.status.state(staId) =
            MAC_STATE_WAIT_FOR_DIFS;
          evt.occurTime(staId) =
            curTime + DIFS;
          evt.evtType(staId) =
            EVT_TP_DIFS_ACHEIVED;
      end
    case MSG_TP_RX_UC_DATA_START
      if 1 == sta.phy.status.medState(staId)
        sta.mac.status.state(staId) =
          MAC_STATE_RX_UC;
        sta.mac.status.srcStaId(staId) =
          para1;
      end
```

```
case MSG_TP_RX_UC_DATA_END
  if 0 == sta.phy.status.medState(staId)
    sta.mac.status.state(staId) =
      MAC_STATE_WAIT_FOR_DIFS;
    evt.occurTime(staId) =
      curTime + DIFS;
    evt.evtType(staId) =
    EVT_TP_DIFS_ACHEIVED;  % set event
  end
case MSG_TP_RX_BC_DATA_START
  if 1 == sta.phy.status.medState(staId)
    sta.mac.status.state(staId) =
      MAC_STATE_RX_BC;
    sta.mac.status.srcStaId(staId) =
      para1;
  end
case MSG_TP_RX_BC_DATA_END
  if 0 == sta.phy.status.medState(staId)
    sta.mac.status.state(staId) =
      MAC_STATE_WAIT_FOR_DIFS;
    evt.occurTime(staId) =
      curTime + DIFS;
    evt.evtType(staId) =
      EVT_TP_DIFS_ACHEIVED;
  end
case MSG_TP_RX_RTS_START
  if 1 == sta.phy.status.medState(staId)
    sta.mac.status.state(staId) =
      MAC_STATE_RX_RTS;
    sta.mac.status.srcStaId(staId) =
      para1;
  end
case MSG_TP_RX_RTS_END
  if 0 == sta.phy.status.medState(staId)
    sta.mac.status.state(staId) =
      MAC_STATE_WAIT_FOR_DIFS;
```

```
        evt.occurTime(staId) =
          curTime + DIFS;
        evt.evtType(staId) =
          EVT_TP_DIFS_ACHEIVED;
      end
    otherwise
      fprintf('wrong msg %d with para %d in
              state %d on sta # %d\n',
              msgType, para1,
              sta.mac.status.state(staId),
              staId);
      pause;
  end

case MAC_STATE_WAIT_FOR_DIFS
  switch msgType
    case MSG_TP_SOMEONE_TX
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_NAV;
      evt.occurTime(staId) = inf;
    case MSG_TP_DIFS_ACHEIVED
      sta.mac.status.state(staId) =
        MAC_STATE_BACK_OFF;
      sta.mac.status.bakOffStartTime(staId) =
        curTime;

      % calculate the expecting backoff time
      if 0 ==
       sta.mac.status.bakOffCnter(staId)
        sta.mac.status.bakOffCnter(staId) =
          floor(rand * 2^sta.mac.status.cwStage(staId));
      end

      bckOffDuration = sta.mac.status.bakOffCnter(staId)
                       * SLOT_SIZE;
      evt.occurTime(staId) = curTime + bckOffDuration;
```

```
      if PKT_TP_UC_BAS == sta.mac.status.pktType(staId)
        evt.evtType(staId) = EVT_TP_UC_TX_DATA_START;
        evt.para1(staId) = sta.mac.status.dstStaId(staId);
      else
        if PKT_TP_BC == sta.mac.status.pktType(staId)
          evt.evtType(staId) = EVT_TP_BC_TX_DATA_START;
        else
          evt.evtType(staId) = EVT_TP_TX_RTS_START;
          evt.para1(staId) =
            sta.mac.status.dstStaId(staId);
        end
      end
    case MSG_TP_RX_UC_DATA_START
      sta.mac.status.state(staId) = MAC_STATE_RX_UC;
      sta.mac.status.srcStaId(staId) = para1;
      evt.occurTime(staId) = inf;     % clear event
    case MSG_TP_RX_RTS_START
      sta.mac.status.state(staId) = MAC_STATE_RX_RTS;
      sta.mac.status.srcStaId(staId) = para1;
      evt.occurTime(staId) = inf;     % clear event
    case MSG_TP_RX_BC_DATA_START
      sta.mac.status.state(staId) = MAC_STATE_RX_BC;
      sta.mac.status.srcStaId(staId) = para1;
      evt.occurTime(staId) = inf;     % clear event
    case MSG_TP_RX_BC_DATA_END
    otherwise
      fprintf('wrong msg %d with para %d in state %d
              on sta # %d\n', msgType, para1,
              sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_BACK_OFF
  switch msgType
    case MSG_TP_SOMEONE_TX
      sta.mac.status.state(staId) = MAC_STATE_WAIT_FOR_NAV;
```

```
  % calculate how many slots have passed
  sta.mac.status.bakOffCnter(staId) =
    sta.mac.status.bakOffCnter(staId) -
    floor((curTime -
          sta.mac.status.bakOffStartTime(staId)) /
          SLOT_SIZE);
  evt.occurTime(staId) = inf;     % clear event
case MSG_TP_TX_UC_DATA_START
  % clear backoff counter
  sta.mac.status.bakOffCnter(staId) = 0;
  sta.mac.status.state(staId) = MAC_STATE_TX_UC;
  time2SendDataPkt = DATA_PKT_LEN / DATA_RATE /
                     TIME_UNIT;
  evt.occurTime(staId) = curTime + time2SendDataPkt;
  evt.evtType(staId) = EVT_TP_UC_TX_DATA_END;
  evt.para1(staId) = sta.mac.status.dstStaId(staId);
case MSG_TP_TX_BC_DATA_START
  sta.mac.status.bakOffCnter(staId) = 0;
  sta.mac.status.state(staId) = MAC_STATE_TX_BC;
  time2SendDataPkt = DATA_PKT_LEN / DATA_RATE /
                     TIME_UNIT;
  evt.occurTime(staId) = curTime + time2SendDataPkt;
  evt.evtType(staId) = EVT_TP_BC_TX_DATA_END;
  evt.para1(staId) = sta.mac.status.dstStaId(staId);
case MSG_TP_TX_RTS_START
  sta.mac.status.bakOffCnter(staId) = 0;
  sta.mac.status.state(staId) = MAC_STATE_TX_RTS;
  time2SendDataPkt = RTS_PKT_LEN / DATA_RATE /
                     TIME_UNIT;
  evt.occurTime(staId) = curTime + time2SendDataPkt;
  evt.evtType(staId) = EVT_TP_TX_RTS_END;
  evt.para1(staId) = sta.mac.status.dstStaId(staId);
case MSG_TP_RX_UC_DATA_START
  sta.mac.status.state(staId) = MAC_STATE_RX_UC;
  sta.mac.status.srcStaId(staId) = para1;
  % calculate how many slots have passed
```

```
      sta.mac.status.bakOffCnter(staId) =
        sta.mac.status.bakOffCnter(staId) -
        floor((curTime -
              sta.mac.status.bakOffStartTime(staId)) /
              SLOT_SIZE);
        evt.occurTime(staId) = inf;
    case MSG_TP_RX_RTS_START
      sta.mac.status.state(staId) = MAC_STATE_RX_RTS;
      sta.mac.status.srcStaId(staId) = para1;
      % calculate how many slots have passed
      sta.mac.status.bakOffCnter(staId) =
        sta.mac.status.bakOffCnter(staId) -
        floor((curTime -
              sta.mac.status.bakOffStartTime(staId)) /
              SLOT_SIZE);
      evt.occurTime(staId) = inf;
    case MSG_TP_RX_BC_DATA_START
      sta.mac.status.state(staId) = MAC_STATE_RX_BC;
      sta.mac.status.srcStaId(staId) = para1;
      % calculate how many slots have passed
      sta.mac.status.bakOffCnter(staId) =
        sta.mac.status.bakOffCnter(staId) -
        floor((curTime -
        sta.mac.status.bakOffStartTime(staId)) /
        SLOT_SIZE);
      evt.occurTime(staId) = inf;
    otherwise
      fprintf('wrong msg %d with para %d in state %d
              on sta # %d\n', msgType, para1,
              sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_TX_UC
  switch msgType
    case MSG_TP_SOMEONE_TX
```

```
      %fprintf('Ignore message %d from station %d
               when I %d am in state %d\n', msgType,
               para1, staId,
               sta.mac.status.state(staId));
    case MSG_TP_RX_UC_DATA_START
      %fprintf('Ignore message %d from station#%d
               when I %d am in state %d\n', msgType,
               para1, staId,
               sta.mac.status.state(staId));
    case MSG_TP_TX_UC_DATA_END
      sta.mac.status.state(staId) =
        MAC_STATE_SIFS_AFTER_TX_UC;
      evt.occurTime(staId) = curTime + SIFS;
      evt.evtType(staId) =
        EVT_TP_READY_TO_RX_ACK;
    otherwise
      fprintf('wrong msg %d with para %d in state %d
              on sta # %d\n', msgType, para1,
              sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_TX_BC
  switch msgType
    case MSG_TP_RX_BC_DATA_START
      sta.mac.status.bcTxStatus(staId) =
        sta.mac.status.bcTxStatus(staId) + 1;
    case MSG_TP_TX_BC_DATA_END
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_DIFS;
      evt.occurTime(staId) = curTime + DIFS;
      evt.evtType(staId) = EVT_TP_DIFS_ACHEIVED;

      if 0 == sta.mac.status.bcTxStatus(staId)
        sta.statistics.NumOfBcpktSentSuccessfully(staId) =
          sta.statistics.NumOfBcpktSentSuccessfully(staId)
```

```
          + 1;
        sta.statistics.bcpktTotalDelay(staId) =
          sta.statistics.bcpktTotalDelay(staId) +
          curTime - sta.mac.status.bcpktBornTime(staId);
      end

      sta.mac.status.bcpktBornTime(staId) = curTime;
      sta.mac.status.bcTxStatus(staId) = 0;
    otherwise
      fprintf('wrong msg %d with para %d in state %d on
              sta # %d\n', msgType, para1,
              sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_TX_RTS
  switch msgType
    case MSG_TP_SOMEONE_TX
      %fprintf('Ignore message %d from station %d when
                I %d am in state %d\n', msgType, para1,
                staId, sta.mac.status.state(staId));
    case MSG_TP_RX_RTS_START
      %fprintf('Ignore message %d from station#%d when
                I %d am in state %d\n', msgType, para1,
                staId, sta.mac.status.state(staId));
    case MSG_TP_TX_RTS_END
      sta.mac.status.state(staId) =
        MAC_STATE_SIFS_AFTER_TX_RTS;
      evt.occurTime(staId) = curTime + SIFS;
      evt.evtType(staId) = EVT_TP_READY_TO_RX_CTS;
    otherwise
      fprintf('wrong msg %d with para %d in state %d
               on sta # %d\n', msgType, para1,
              sta.mac.status.state(staId), staId);
      pause;
  end
```

```
case MAC_STATE_SIFS_AFTER_TX_UC
  switch msgType
    case MSG_TP_READY_TO_RX_ACK
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_ACK;
      evt.occurTime(staId) = curTime + ACK_TIMEOUT;
      evt.evtType(staId) = EVT_TP_ACK_TIME_OUT;
    case MSG_TP_RX_UC_DATA_END
      %fprintf('Ignore message %d from station#%d when
                I %d am in state %d\n', msgType, para1,
                staId, sta.mac.status.state(staId));
    case MSG_TP_SOMEONE_STOP_TX
      %fprintf('Ignore message %d from station#%d when
                I %d am in state %d\n', msgType, para1,
                staId, sta.mac.status.state(staId));
    otherwise
      fprintf('wrong msg %d with para %d in state %d
               on sta # %d\n', msgType, para1,
               sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_SIFS_AFTER_TX_RTS
  switch msgType
    case MSG_TP_READY_TO_RX_CTS
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_CTS;
      evt.occurTime(staId) = curTime + CTS_TIMEOUT;
      evt.evtType(staId) = EVT_TP_CTS_TIME_OUT;
    case MSG_TP_RX_RTS_END
      %fprintf('Ignore message %d from station#%d when
                I %d am in state %d\n', msgType, para1,
                staId, sta.mac.status.state(staId));
    case MSG_TP_SOMEONE_STOP_TX
      %fprintf('Ignore message %d from station#%d when
```

```
                I %d am in state %d\n', msgType, para1,
                staId, sta.mac.status.state(staId));
    otherwise
      fprintf('wrong msg %d with para %d in state %d on
               sta # %d\n', msgType, para1,
               sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_WAIT_FOR_ACK
  switch msgType
    case MSG_TP_SOMEONE_TX
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_NAV;
      if (MAX_CW_STAGE > sta.mac.status.cwStage(staId))
        % Enlarge contention window
        sta.mac.status.cwStage(staId) =
          sta.mac.status.cwStage(staId) + 1;
      end
      evt.occurTime(staId) = inf;
    case MSG_TP_ACK_TIME_OUT
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_DIFS;
      evt.occurTime(staId) = curTime + DIFS;
      evt.evtType(staId) = EVT_TP_DIFS_ACHEIVED;
      if (MAX_CW_STAGE > sta.mac.status.cwStage(staId))
        % Enlarge contention window
        sta.mac.status.cwStage(staId) =
          sta.mac.status.cwStage(staId) + 1;
      end
    case MSG_TP_RX_ACK_START
      if sta.mac.status.dstStaId(staId) ~= para1
        fprintf('dstStaId %d and para1 %d don''t match
                 on sta # %d\n',
                 sta.mac.status.dstStaId(staId),
                 para1, staId);
```

```
        pause;
      end
      sta.mac.status.state(staId) = MAC_STATE_RX_ACK;
      evt.occurTime(staId) = inf;
    case MSG_TP_RX_UC_DATA_START
      sta.mac.status.state(staId) = MAC_STATE_RX_UC;
      sta.mac.status.srcStaId(staId) = para1;
      if (MAX_CW_STAGE > sta.mac.status.cwStage(staId))
        % Enlarge contention window
        sta.mac.status.cwStage(staId) =
          sta.mac.status.cwStage(staId) + 1;
      end

      % clear event
      evt.occurTime(staId) = inf;
    otherwise
      fprintf('wrong msg %d with para %d in state %d
               on sta # %d\n', msgType, para1,
               sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_WAIT_FOR_CTS
  switch msgType
    case MSG_TP_SOMEONE_TX
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_NAV;
      if (MAX_CW_STAGE > sta.mac.status.cwStage(staId))
        % Enlarge contention window
        sta.mac.status.cwStage(staId) =
          sta.mac.status.cwStage(staId) + 1;
      end
      evt.occurTime(staId) = inf;     % clear event
    case MSG_TP_CTS_TIME_OUT
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_DIFS;
```

```
      evt.occurTime(staId) = curTime + DIFS;
      evt.evtType(staId) = EVT_TP_DIFS_ACHEIVED;
      if (MAX_CW_STAGE > sta.mac.status.cwStage(staId))
        % Enlarge contention window
        sta.mac.status.cwStage(staId) =
          sta.mac.status.cwStage(staId) + 1;
      end
    case MSG_TP_RX_CTS_START
      if sta.mac.status.dstStaId(staId) ~= para1
        fprintf('dstStaId %d and para1 %d don''t match on
                 sta # %d\n',
                 sta.mac.status.dstStaId(staId),
                 para1, staId);
        pause;
      end
      sta.mac.status.state(staId) = MAC_STATE_RX_CTS;
      evt.occurTime(staId) = inf;
    case MSG_TP_RX_RTS_START
      sta.mac.status.state(staId) = MAC_STATE_RX_RTS;
      sta.mac.status.srcStaId(staId) = para1;
      if (MAX_CW_STAGE > sta.mac.status.cwStage(staId))
        % Enlarge contention window
        sta.mac.status.cwStage(staId) =
          sta.mac.status.cwStage(staId) + 1;
      end
      evt.occurTime(staId) = inf;
    otherwise
      fprintf('wrong msg %d with para %d in state %d on
               sta # %d\n', msgType, para1,
               sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_RX_ACK
  switch msgType
    case MSG_TP_RX_ACK_END
```

```
  if sta.mac.status.dstStaId(staId) ~= para1
    fprintf('dstStaId %d and para1 %d don''t match
             on sta # %d\n',
             sta.mac.status.dstStaId(staId),
             para1, staId);
    pause;
  end
  sta.mac.status.state(staId) =
    MAC_STATE_WAIT_FOR_DIFS;
  evt.occurTime(staId) = curTime + DIFS;
  evt.evtType(staId) = EVT_TP_DIFS_ACHEIVED;

  % Restore contention window
  sta.mac.status.cwStage(staId) = MIN_CW_STAGE;

  % generate new destination station ID
  tempVar = staId;
  while tempVar == staId
    tempVar = ceil(rand * numOfSta);
  end
  sta.mac.status.dstStaId(staId) = tempVar;

  if PKT_TP_UC_BAS == sta.mac.status.pktType(staId)
sta.statistics.NumOfUcpktBasSentSuccessfully(staId) =
  sta.statistics.NumOfUcpktBasSentSuccessfully(staId) +
  1;
    sta.statistics.ucpktBasTotalDelay(staId) =
      sta.statistics.ucpktBasTotalDelay(staId) +
      curTime - sta.mac.status.ucpktBornTime(staId);
  else
sta.statistics.NumOfUcpktRtsSentSuccessfully(staId) =
  sta.statistics.NumOfUcpktRtsSentSuccessfully(staId) +
  1;
  sta.statistics.ucpktRtsTotalDelay(staId) =
    sta.statistics.ucpktRtsTotalDelay(staId) +
    curTime - sta.mac.status.ucpktBornTime(staId);
```

```
      end
      sta.mac.status.ucpktBornTime(staId) = curTime;
    otherwise
      fprintf('wrong msg %d with para %d in state %d
               on sta # %d\n', msgType, para1,
               sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_RX_CTS
  switch msgType
    case MSG_TP_RX_CTS_END
      if sta.mac.status.dstStaId(staId) ~= para1
        fprintf('dstStaId %d and para1 %d don''t
                 match on sta # %d\n',
                 sta.mac.status.dstStaId(staId),
                 para1, staId);
        pause;
      end
      sta.mac.status.state(staId) =
        MAC_STATE_SIFS_AFTER_RX_CTS;
      evt.occurTime(staId) = curTime + SIFS;
      evt.evtType(staId) = EVT_TP_UC_TX_DATA_START;
      % Restore contention window
      sta.mac.status.cwStage(staId) = MIN_CW_STAGE;
    otherwise
      fprintf('wrong msg %d with para %d in state %d
               on sta # %d\n', msgType, para1,
               sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_SIFS_AFTER_RX_CTS
  switch msgType
    case MSG_TP_TX_UC_DATA_START
      sta.mac.status.state(staId) =
```

```
        MAC_STATE_TX_UC;
      time2SendDataPkt = DATA_PKT_LEN / DATA_RATE /
        TIME_UNIT;
      evt.occurTime(staId) = curTime +
        time2SendDataPkt;
      evt.evtType(staId) = EVT_TP_UC_TX_DATA_END;
      evt.para1(staId) = sta.mac.status.dstStaId(staId);
    otherwise
      fprintf('wrong msg %d with para %d in state %d on
               sta # %d\n', msgType, para1,
               sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_RX_UC
  switch msgType
    case MSG_TP_SOMEONE_TX
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_NAV;
    case MSG_TP_RX_UC_DATA_START
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_NAV;
    case MSG_TP_RX_UC_DATA_END
      if sta.mac.status.srcStaId(staId) ~= para1
        fprintf('srcStaId %d and para1 %d don''t match
                 on sta # %d\n',
                 sta.mac.status.srcStaId(staId),
                 para1, staId);
        pause;
      end
      sta.mac.status.state(staId) =
        MAC_STATE_SIFS_AFTER_RX_UC;
      evt.occurTime(staId) = curTime + SIFS;
      evt.evtType(staId) = EVT_TP_TX_ACK_START;
      evt.para1(staId) = sta.mac.status.srcStaId(staId);
    otherwise
```

```
      fprintf('wrong msg %d with para %d in state %d on
               sta # %d\n', msgType, para1,
               sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_SIFS_AFTER_RX_UC
  switch msgType
    case MSG_TP_TX_ACK_START
      sta.mac.status.state(staId) =
        MAC_STATE_TX_ACK;
      time2SendAckPkt = ACK_PKT_LEN / DATA_RATE /
        TIME_UNIT;
      evt.occurTime(staId) = curTime + time2SendAckPkt;
      evt.evtType(staId) = EVT_TP_TX_ACK_END;
      evt.para1(staId) = sta.mac.status.srcStaId(staId);
    otherwise
      fprintf('wrong msg %d with para %d in state %d
               on sta # %d\n', msgType, para1,
               sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_TX_ACK
  switch msgType
    case MSG_TP_TX_ACK_END
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_DIFS;
      evt.occurTime(staId) = curTime + DIFS;
      evt.evtType(staId) = EVT_TP_DIFS_ACHEIVED;
    otherwise
      fprintf('wrong msg %d with para %d in state %d
               on sta # %d\n', msgType, para1,
               sta.mac.status.state(staId), staId);
      pause;
  end
```

```
case MAC_STATE_RX_BC
  switch msgType
    case MSG_TP_RX_BC_DATA_START
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_NAV;
    case MSG_TP_RX_BC_DATA_END
      if sta.mac.status.srcStaId(staId) ~= para1
          fprintf('srcStaId %d and para1 %d don''t
                   match on sta # %d\n',
                   sta.mac.status.srcStaId(staId),
                   para1, staId);
          pause;
      end
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_DIFS;
      evt.occurTime(staId) = curTime + DIFS;
      evt.evtType(staId) = EVT_TP_DIFS_ACHEIVED;
    otherwise
      fprintf('wrong msg %d with para %d in state
               %d on sta # %d\n', msgType,
               para1, sta.mac.status.state(staId),
               staId);
      pause;
  end

case MAC_STATE_RX_RTS
  switch msgType
    case MSG_TP_SOMEONE_TX
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_NAV;
    case MSG_TP_RX_RTS_START
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_NAV;
    case MSG_TP_RX_RTS_END
      if sta.mac.status.srcStaId(staId) ~= para1
```

```
        fprintf('srcStaId %d and para1 %d don''t
                 match on sta # %d\n',
                 sta.mac.status.srcStaId(staId),
                 para1, staId);
        pause;
      end
      sta.mac.status.state(staId) =
        MAC_STATE_SIFS_AFTER_RX_RTS;
      evt.occurTime(staId) = curTime + SIFS;
      evt.evtType(staId) = EVT_TP_TX_CTS_START;
      evt.para1(staId) = sta.mac.status.srcStaId(staId);
    otherwise
      fprintf('wrong msg %d with para %d in state %d on
               sta # %d\n', msgType, para1,
               sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_SIFS_AFTER_RX_RTS
  switch msgType
    case MSG_TP_TX_CTS_START
        sta.mac.status.state(staId) = MAC_STATE_TX_CTS;
        time2SendAckPkt = CTS_PKT_LEN / DATA_RATE /
          TIME_UNIT;
        evt.occurTime(staId) = curTime + time2SendAckPkt;
        evt.evtType(staId) = EVT_TP_TX_CTS_END;
        evt.para1(staId) = sta.mac.status.srcStaId(staId);
    otherwise
        fprintf('wrong msg %d with para %d in state %d
                 on sta # %d\n', msgType, para1,
                 sta.mac.status.state(staId), staId);
        pause;
  end

case MAC_STATE_TX_CTS
  switch msgType
```

```
    case MSG_TP_TX_CTS_END
      sta.mac.status.state(staId) =
        MAC_STATE_SIFS_AFTER_TX_CTS;
      evt.occurTime(staId) = curTime + SIFS;
      evt.evtType(staId) = EVT_TP_READY_TO_RECEIVE_DATA;
    otherwise
      fprintf('wrong msg %d with para %d in state %d on
               sta # %d\n', msgType, para1,
               sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_SIFS_AFTER_TX_CTS
  switch msgType
    case MSG_TP_READY_TO_RECEIVE_DATA
      sta.mac.status.state(staId) =
        MAC_STATE_WAIT_FOR_DATA;
    otherwise
      fprintf('wrong msg %d with para %d in state %d on
               sta # %d\n', msgType, para1,
               sta.mac.status.state(staId), staId);
      pause;
  end

case MAC_STATE_WAIT_FOR_DATA
  switch msgType
    case MSG_TP_RX_UC_DATA_START
      sta.mac.status.state(staId) = MAC_STATE_RX_UC;
    otherwise
      fprintf('wrong msg %d with para %d in state %d on
               sta # %d\n', msgType, para1,
               sta.mac.status.state(staId), staId);
      pause;
  end

otherwise
```

```
            fprintf('wrong state %d w/ message %d on sta # %d\n',
                    sta.mac.status.state(staId), msgType, staId);
            pause;
        end
      end
    end

    numOfLoop = numOfLoop + 1;
%   ucBasThroughput(numOfLoop) =
%       sum(sta.statistics.NumOfUcpktBasSentSuccessfully)*
%       DATA_PAYLOAD_LEN/DATA_RATE/TIME_UNIT/SIM_DURATION;
%   ucBasDelay(numOfLoop) =
%     sum(sta.statistics.ucpktBasTotalDelay)/
%     sum(sta.statistics.NumOfUcpktBasSentSuccessfully)*
%     TIME_UNIT;
%   bcThroughput(numOfLoop) =
%     sum(sta.statistics.NumOfBcpktSentSuccessfully)*
%     DATA_PAYLOAD_LEN/DATA_RATE/TIME_UNIT/SIM_DURATION;
%   bcDelay(numOfLoop) =
%     sum(sta.statistics.bcpktTotalDelay)/
%     sum(sta.statistics.NumOfBcpktSentSuccessfully)*
%     TIME_UNIT;
    ucRtsThroughput(numOfLoop) =
      sum(sta.statistics.NumOfUcpktRtsSentSuccessfully)*
      DATA_PAYLOAD_LEN/DATA_RATE/TIME_UNIT/SIM_DURATION;
    ucRtsDelay(numOfLoop) =
      sum(sta.statistics.ucpktRtsTotalDelay)/
      sum(sta.statistics.NumOfUcpktRtsSentSuccessfully)*
      TIME_UNIT;
    fprintf('numOfSta %d loop finished\n', numOfSta);
end % end of "for numOfSta = MIN_NUM_OF_STA ..."

%plot(MIN_NUM_OF_STA : NUM_OF_STA_STEP : MAX_NUM_OF_STA,
%       ucBasThroughput, 'g*-');
%plot(MIN_NUM_OF_STA : NUM_OF_STA_STEP : MAX_NUM_OF_STA,
%       bcThroughput, 'g*-');
```

```
plot(MIN_NUM_OF_STA : NUM_OF_STA_STEP : MAX_NUM_OF_STA,
     ucRtsThroughput, 'g*-'); axis([0 50 0.5 0.9]);
figure(2);
%plot(MIN_NUM_OF_STA : NUM_OF_STA_STEP : MAX_NUM_OF_STA,
%      ucBasDelay, 'ko-');
%plot(MIN_NUM_OF_STA : NUM_OF_STA_STEP : MAX_NUM_OF_STA,
%      bcDelay, 'ko-');
plot(MIN_NUM_OF_STA : NUM_OF_STA_STEP : MAX_NUM_OF_STA,
     ucRtsDelay, 'ko-');
axis([0 50 0 0.5])

ucRtsThroughput

ucRtsDelay
```