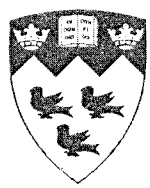
Wireless GINI: An Emulator for Ad-hoc Wireless Local Area Networks

Sheng Liao



Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

October 2005

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering.

2005/10/26

Your file Votre référence
ISBN: 978-0-494-24985-7
Our file Notre référence
ISBN: 978-0-494-24985-7

Abstract

With the increasing popularity of the Internet, computer networking courses are becoming important elements of undergraduate and graduate curricula. Unfortunately, due to the security and cost worries, it is impractical to establish a real network environment for teaching and learning purposes. The GINI toolkit developed by Advanced Networking Research Lab of Mcgill University provides the solution for it. It is an overlay experimentation platform that allows running a virtual micro network on a single host.

However, wireless applications are not supported by GINI currently while the explosive growth of the wireless computing calls for a deep understanding of the wireless network. Therefore, in my thesis, an emulator for ad-hoc wireless local area network is implemented and plugged into the GINI toolkit. It is built on a virtual Linux kernel, called User-mode Linux, running on top of a central controlling instance. The central controlling instance, also known as wireless GINI center, provides efficient medium access control mechanisms, a set of complete propagation channel models, concrete frame corruption functions, and practical node mobility patterns. Generally, the wireless GINI can establish a software-based but realistic wireless environment, helping diverse experiments on new wireless network algorithms.

Résumé(Français)

Avec la popularité sans cesse croissante de l'Internet, les cours de réseautique deviennent des élements importants des connaissances des diplômés et non-Diplômés. Malheureusement, dû aux inquiétudes liées à la sécurité et aux coûts, il n'est pas facile d'établir un environement réseau pour but d'apprentissage et d'enseignement. La trousse d'outils GINI, développée par le Laboratoire de Recherche en Réseautique Avancée de l'université McGill, apporte une solution à ce problème. C'est une plateforme de recouvrement pour expérimentation, qui donne la possibilité de faire fonctionner un micro-réseau complet, sur un seul hôte.

Parcontre, les applications sans-fil ne sont pas encore incluses dans la trousse GINI alors que la accroissement explosif de la popularité de l'ordinateur portable demande une très profonde compréhension des réseaux sans-fil. Donc, dans ma thèse, un émulateur de réseau sans-fil ad-hoc est construit et greffé à la trousse d'outils GINI. Il est construit sur un noyau Linux virtuel, appelé Mode-Usager Linux, fonctionnant au plus haut des processus de contrôle centraux. La centrale de gestion des processus, aussi connue sous le nom de Centre GINI sans-fil, procure un excellent mécanisme d'accès moyen, un assortiment complet de modèles de canaux de propagation, de fonctions de corruption de trame concrets et de pratiques formulations de mobilité des branches. Généralement, la trouse Sans-fil GINI peut établir un environnement réseau sans-fil réaliste, existant seulement virtuellement, aidant à mener des expériences diverses sur les nouveaux algorithmes de réseau sans-fil.

Acknowledgments

I would like to thank Prof.Muthucumaru Maheswaran for his continuous guidance and supervision on my thesis research. His advice and suggestions are invaluable for the wireless GINI software development.

Thanks should be given to everyone in the Advanced Network Research Lab for their valuable comments. Special thanks to Balasubramaneyam Maniymaran, Arindam Mitra and Shah Asaduzzaman, who provided lots of help on the testing and debugging of my program, the general computer knowledge, and the thesis correcting. I'd like to acknowledge Sheng Lu and Yi Yang as well, who gave me some advice on the implementation of the software. Furthermore, I would like to thank Jean-Sébastien Légaré's unselfishness contribution on my thesis organization and content refinement.

I would like to express my appreciation to my mother Mianli Xu, my parents-in-law Jiayou Nie and Lamei Yang and my wife Lin Nie for their successively spirit support and encouragement all the time. Besides, deepest thanks and memorization to my father, Hexiang Liao, who passed away two years ago but lives in my heart forever. He is always the brightest light in my lifetime.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The objective of my thesis research is to develop software emulation model for Ad-Hoc wireless local area networks based on the IEEE 802.11 standard. This model is built as an extension of the existing *GINI* toolkit developed by the Advanced Network Research Lab in Mcgill University. The intent is to allow researchers and students to evaluate the performance of different instantiations of IEEE 802.11 networks.

GINI toolkit is an overlay experimentation platform for creating virtual micro Internet for teaching and learning based on the following principles: user-level deployment, scale reduction, and minimal abstraction. User-level deployment enables GINI to run on lab workstations without requiring kernel level changes, facilitates debugging, and makes the workstations immune to misbehaving or crashing GINI instances. Scale reduction enables GINI to emulate small-scale networks without loss of detail using a single workstation. However, larger networks can be handled by GINI by distributing the configurations over multiple workstations. Because GINI is designed as a teaching and learning tool, one of the goals is to expose the students to the underlying operations performed by the Internet. To achieve this goal, the abstractions introduced as part of GINI are kept to a bare minimum. The simple *application programming interfaces* (APIs) exported by GINI are part of that minimal abstraction set and they can be used by students to modify the functionalities provided by the micro-scale Internet. Further, the minimal abstraction principle ensures gentle learning curves.

Consider the small example network in Figure 1.1. Suppose GINI is used to emulate this network, the resulting configuration will look like Figure 1.2. The end nodes are

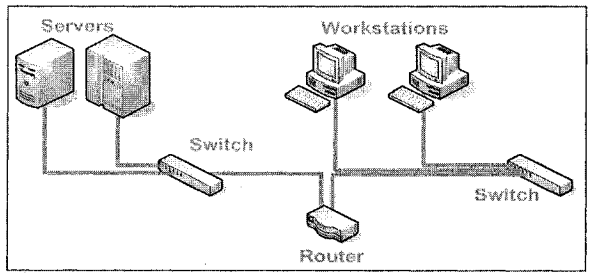


Fig. 1.1 An example network in GINI

represented by *User-mode Linux* (UML) instances and network elements are represented by GINI routers and switches. These elements are interconnected using modified inter-process communication routines to complete the emulator setup.

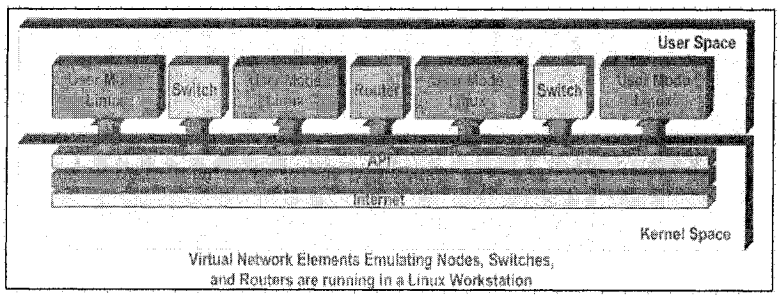


Fig. 1.2 The back-ends of the example network

1.1 Overview of the GINI toolkit

Network emulation within GINI has three phases: design, emulate, and visualize as shown in Figure 1.3. The student inputs the network topology using the graphical interface provided by the topology design tool shown in Figure 1.4.

It converts the graphical topology input into a XML marked up document defining the example network. A snippet of the XML topology specification is shown in Figure 1.5. The topology generator uses the definition to spawn various components such as UMLs, GINI routers, GINI switches, and interconnections to create the example topology. Once the topology is setup, a virtual network that looks and feels similar to a physical realization of the example network runs within the workstation. The student clicking on the icon that

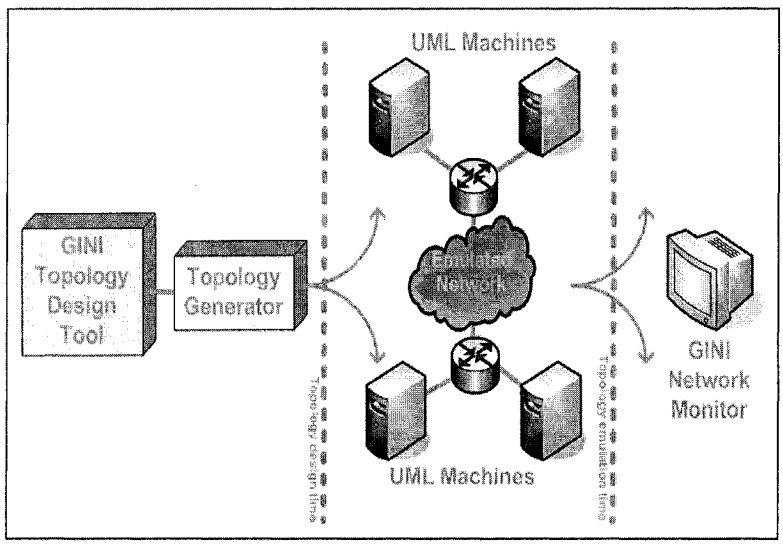


Fig. 1.3 Three phases in GINI emulation

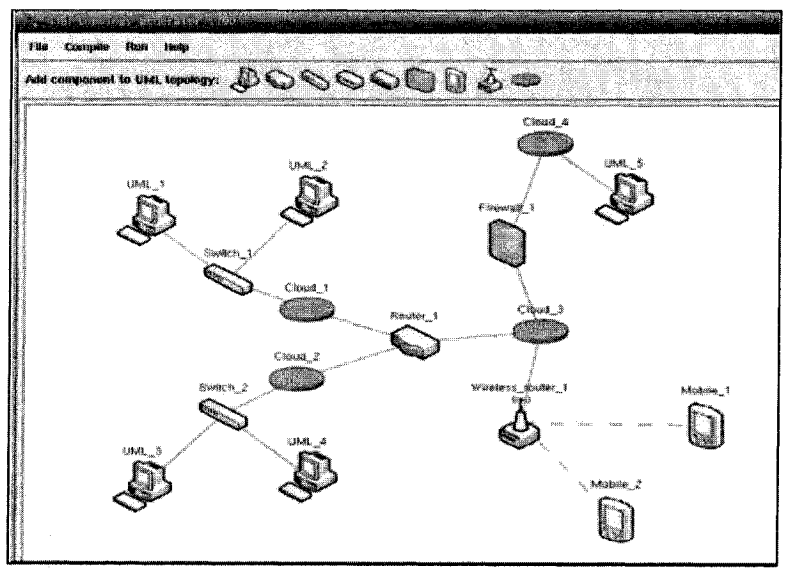


Fig. 1.4 The topology design tool in GINI

corresponds to the virtual device on the topology in Figure 1.4. Further, the icon changes color or shape to show the different states of the corresponding virtual device.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topgen SYSTEM "./topgen.dtd">
<topgen>
<vs id="1">
  <port>1115</port>
  <hub/>
</vs>
<vs id="2">
  <port>1116</port>
  <hub/>
</vs>

<vm name="uml1">
  <filesystem type="cow">root_fs.rh-6.2-full.pristine.20020312</filesystem>
  <if>
    <switch_id>1</switch_id>
    <mac>fe:fd:00:00:00:01</mac>
    <ipv4>10.10.1.25</ipv4>
    <route type="net" netmask="255.255.255.0" gw="10.10.1.1">10.10.2.0</route>
  </if>
</vm>
.....

</topgen>
```

Fig. 1.5 The XML topology specification of the example network

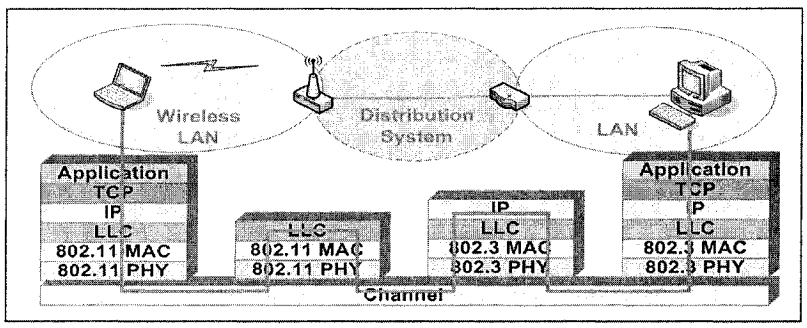


Fig. 1.6 Emulation of a wired and wireless scenario

1.2 Thesis contribution and organization

My thesis focuses on the wireless extension of the existing GINI toolkit [2]. The MAC and physical layers defined by the 802.11 standard and the wireless channel have been implemented. It allows the seamless integration of the wireless network with the existing

wired one in the GINI toolkit. Figure 1.6 shows an emulation scenario that includes both wireless and wired components. As shown in Figure 1.4, the topology design tool allows students to specify arbitrary wireless or wired network configurations.

This thesis is organized as follows: Chapter 2 presents some related works on the network simulation and emulation. Chapter 3 focuses on the overview of the background knowledge, which includes the IEEE 802.11 standard, the User Mode Linux and the existing GINI system. Chapter 4 gives a general description of the design idea behind the wireless GINI while Chapter 5 focuses on the concrete implementation of the functional layers and modules. Chapter 6 shows some detailed analysis of the data collected from different experimental scenarios.

Chapter 2

Related works

2.1 Description of simulators

Due to the high costs of the networking equipment and maintenance, simulators are the most affordable solutions for experiments related to teaching networks. The flexibility of the simulator enables the whole experiment to take place in a single powerful computer. The network structures are reconfigurable and reproducible, which makes simulations a better approach even in the long run. In addition, simulation results can be easily analyzed because important information can be extracted via log files in predefined format.

Currently, several simulators are used in practical research and teaching. The *NS-2 simulator* [3] developed by UC Berkeley and the *Java simulator* [4] by the Ohio State University and University of Illinois are the most popular simulators. Both of these simulators are designed to adopt to different kinds of network types, i.e, wired networks, wireless networks and satellite networks. Other simulators, such as the *Simple Ad-hoc siMulator* (SAM) [5] and the *A Simple Ad Hoc Network Simulator* (SANS) [6] are designed specifically for Ad-hoc wireless networks. These simulators contain simulation models of some main protocols and mechanisms and may be easily extended to support new protocols.

Simulations are however only an approximation of real networks, because the traffic running on the simulator is usually not the one that can be transferred in the real world. The simulator just executes the commands that predefined in some specific order organized by the scheduler and the time interval between two consecutive instantaneous events is totally ignored.

In a network simulation, only the configuration at the input side and the data analysis

at the output side is visible. The intermediate processes are totally transparent. The mistakes caused by the users, such as configuration errors, cannot be monitored and will eventually result in failure to run. From the points mentioned above, simulators are not the most appropriate tools for increasing the understanding of networks.

2.2 Description of emulators

The shortcomings of the simulator can be addressed by emulators. An emulator for wireless communication networks, called *UML emulator* [7], developed by Université Catholique de Louvain is built on a UML kernel. The UML is extended to support the wireless functions by an attached software based wireless card driver, called hostap driver. All the UMLs with a wireless driver are interconnected by a physical layer emulator, which is responsible for the packet error calculation based different propagation medium. In this way, it allows running highly realistic emulations on a completely virtual environment.

Another emulator specified for mobile ad-hoc networks, called *Microkernel-based Network Emulator* [8], is a system based on multiple Linux instances running on top of the Fiasco microkernel [9]. Fiasco is a second-generation microkernel that only implements a basic functional operating system and allows a non-privileged system mode. The interconnections between the Linux instances or to physical networks are managed by a special multiplexer component. With the control of the multiplexer, the movements of mobile stations and the channel conditions can be traced all over the time to provide the useful information for the emulator calculations.

2.3 Wireless GINI emulator

The wireless GINI implemented in my thesis is a UML-based emulator for the *wireless local area network* (WLAN) written in C. It has the following features:

- It is seamlessly integrated with the virtual wired network emulated by GINI. The final goal of wireless GINI is not only to emulate ad-hoc WLAN, but also to provide infrastructure for WLAN that can connect to other higher level networks, such as 802.x networks.

- The wireless GINI has an independently running central control module that ex-

changes packets with UMLs through TCP sockets. Its independence makes it possible to make the wireless GINI as the basis for all possible virtual wireless devices, such as a new implementation of GINI wireless router. Moreover, centralized control is convenient for the management of the virtual network.

- The independence of wireless GINI enables the traffic control mechanism in the wireless GINI adaptor, whereas normally traffic flow is uncontrollable in other emulators.
- A set of wireless channels are implemented to provide a entirely software-based but realistic wireless environment.
- A global system time is provided in wireless GINI. It helps to handle the time critical events, especially for the frame collision, which is not implemented in other emulators to the best of my knowledge.

The details of the design and implementation of the wireless GINI are given in Chapter 4 and Chapter 5.

Chapter 3

Background knowledge

Unlike the wired network emulation, which is already implemented in the existing GINI toolkit, the wireless network emulation faces more challenges due to the unpredictable wireless propagation medium. To make the GINI toolkit support the wireless applications, a deeper understanding of three components is necessary: the IEEE 802.11 standard, the User-mode Linux, and the existing GINI system.

3.1 Overview of the 802.11 standard

IEEE 802.11 Standard [1] defines the protocols deployed in the operation of the wireless LAN and uses the *carrier sense multiple access protocol with collision avoidance* (CSMA/CA) mechanism to control the medium sharing. The *medium access control* (MAC) is applicable for the network under the control of the *access point* (AP) as well as that composed by several independent stations. The protocol includes authentication, association, and reassociation services, an optional encryption/decryption procedure, power management to reduce power consumption in mobile stations, and a point coordination function for time-bounded transfer of data. The infrared implementation of the physical layer supports $1Mbps$ data rate with an optional $2Mbps$ extension. The radio implementations of the *Physical layer* (PHY) specify either a *frequency-hopping spread spectrum* (FHSS) supporting $1Mbps$ and an optional $2Mbps$ data rate or a *direct sequence spread spectrum* (DSSS) supporting both 1 and $2Mbps$ data rates.

3.1.1 Architecture of wireless local area networks

An 802.11 LAN is based on a cellular architecture where the system is subdivided into cells, each of which is called a *Basic Service Set* (BSS). The 802.11 standard defines two modes: *Ad- hoc* and *infrastructure.*

The Ad-hoc mode is similar to a *peer-to-peer* network in which there is no central management server responsible for controlling the network. The WLAN in ad-hoc mode can be described as an *Independent Basic Service Set* (IBSS). All nodes in the IBSS WLAN can communicate with others directly without the requirement of tunneling the traffic through a centralized server. Normally, Ad-hoc WLAN can only offer the capacity of communication in a relatively small area because of the limitation of the full-mesh network topology and provides no access to the higher level networks.

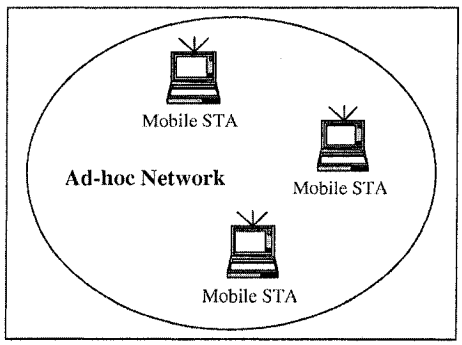


Fig. 3.1 Sketch of an Ad-hoc WLAN

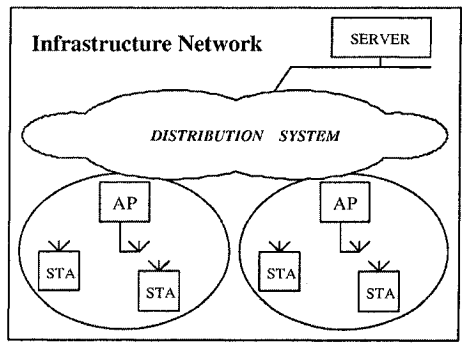


Fig. 3.2 Sketch of an infrastructure WLAN

The WLAN of infrastructure WLAN is composed of at least one access point and a number of wireless mobile stations. The AP is analogous to the base station in a cellular mobile network, acting as a management server for the corresponding BSS. Communication traffic between two nodes in the same BSS should be relayed by AP but cannot be transferred directly. The development of the AP allows the extension of the communication range. A set of BSSs connected to the same *distribution system* (DS) through APs can form a subnetwork called *Extended Service Set* (ESS). Both the single-channel and multi-channel mode for the ESS configuration are supported by the IEEE 802.11 standard based on different network requirements.

The distribution system is not defined by the 802.11 standard, therefore, it may be 802.11 compliant or non-standard. Any data traffic between the a IEEE 801.11 WLAN

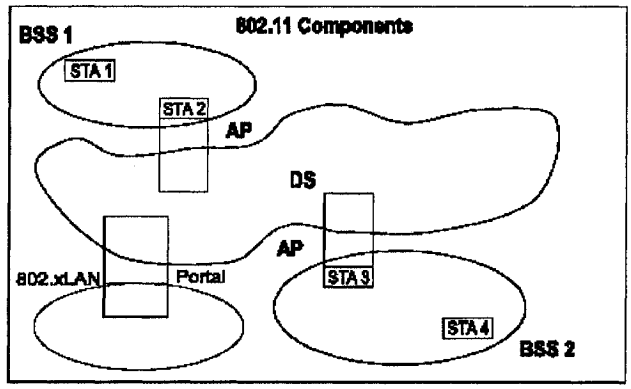


Fig. 3.3 The communication between the WLAN and the 802.x network [1]

and a non-IEEE 802.11 network should be processed by a logical point which is called *portal*. It is responsible for providing the seamless integration between different protocols, acting as a frame translator.

3.1.2 IEEE 802.11 physical layer

At the physical layer, three physical techniques for WLAN are defined by IEEE 802.11 standard: IR, FHSS and DSSS.

Infrared Technology(IR)

IR utilizes the wavelength range from 850 to 950 nm for signal transmission. It can only be used for the indoor communication. Its implementation allows the reception of the *line-of-sight* (LOS) and reflected signals. Two basic access rates can be offered by this technique, which are $1Mbps$ by using 16-pulse position modulation (PPM) and $2Mbps$ by using 4-PPM modulation [10].

Frequency Hopping Spread Spectrum Technology (FHSS)

Frequency Hopping Spread Spectrum [1] transmits the base-band signal carried by the frequency-changeable narrow band carrier. There are twenty-two hop patterns defined by the IEEE 802.11 standard and all of them are selected from the 2.4GHz *Industrial, Scientific, and Medical* (ISM) band (2.4000-2.4835 GHz) band. The bandwidth for each

individual channel is 1MHz. FHSS technology allows the frequency of the carrier used to modulate the signal shift from one to another at a fixed hop rate. This speciality can prevent the signal from interfering with specified frequency and being decoded by others who do not know the rate and the sequence of the frequency shifts. This security protection was adopted for military purpose. FHSS can support data rates from 1.2 to 2 Mbps with maximum distance of 620 miles. The 2.4 GHz band is divided into 75 one-MHz sub-channels and the hopping pattern can be described as a sequence of sub-channels, over which the *Transmitter-Receiver* (T-R) pair has an agreement to transmit the signals.

Direct Sequence Spread Spectrum (DSSS)

The DSSS [1] is another secure wireless technology using the 2.4 GHz ISM frequency band. The bandwidth for each channel is 11MHz with the DSSS instead of the 1MHz used by the FSSS. The difference is caused by the introduction of a 11-bit chip sequence, also known as the Barker code, to modulate the data stream in DSSS to provide the complete security. The signal spectrum is spread by the modulation, which makes it appear as wide band noise to the sniffers. Another advantage of DSSS technology is that the spreading codes can be so useful on recovering the original signal from some bit corruptions that retransmissions can be avoided. Both $1Mbps$ and $2Mbps$ data access rates are supported by DSSS mode by *Differential Binary Phase Shift Keying* (DBPSK) and *Differential Quadrature Phase Shift Keying* (DQPSK) modulation methods, respectively.

3.1.3 IEEE 802.11 MAC layer

The MAC layer provides various services to manage authentication, de-authentication, association, de-association, privacy, data transfer and power management.

The most important responsibility that the MAC layer takes is the medium access control. CSMA/CA mechanism is used in the 802.11 standard while the *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD) mechanism is implemented in the 802.3 Ethernet network. Unlike the wired ethernet, the mobile node can not keep listening to the channel while transmitting because its own signal strength at the antenna is much stronger than the signal from anyone else. This half-duplex characteristic prevents the implementation of collision detect in wireless networks. And the bandwidth of the wireless channel is much lower than that of the wired line, so the wasted overhead caused by the collision

frames can not be ignored as in the wired network. To make the right decision on the channel state and improve the efficiency of the wireless networks, two kinds of coordination functions are introduced by the 802.11 standard: the *Distributed Coordination Function* (DCF) and the optional *Point Coordination Function* (PCF). The MAC architecture can be described as shown in Figure 3.4 as providing the PCF through the services of the DCF.

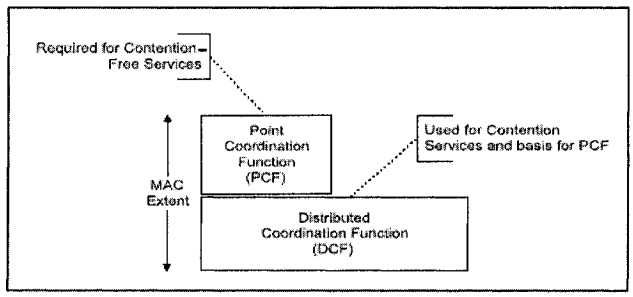


Fig. 3.4 The MAC layer architecture [1]

Distributed Coordination Function (DCF)

Distributed Coordination Function, known also as the CSMA/CA mechanism, is one of the basic medium access control methods. The DCF can be used in both IBSS and infrastructure network. As a transmitter, the mobile node should sense the medium first and can only proceed with the transmission only if the medium is determined to be free for a fixed time duration specified by the 802.11 standard. Once the medium is detected as busy, the node should defer until the end of the current transmission. After the deferral or a successful transmission, instead of attempting to transmit again, a back-off process should be invoked with a randomly selected back-off interval. In order to reduce the bandwidth waste caused by collisions, short control frames (*request to send* (RTS) and *clear to send* (CTS) frames) are exchanged prior to the real data transmission. The details about the RTS/CTS exchange, transmission deferral, and back-off process are described in Section 5.3.2.

Point Coordination Function (PCF)

Point Coordination Function (PCF) is an optional access method in the 802.11 standard, which can only be used in infrastructure mode. In PCF, the AP takes the role of the

polling master and determine which mobile station can access the medium by the point coordination function embedded in itself. An access priority mechanism (e.g, *Round Robin Mechanism*) is normally used as a polling mechanism in PCF. The transaction of the Beacon management frames enable collision-free data transmission by forcing nodes other than the receiver to keep silent for the time period marked in the beacon frames. In addition, the *inter-frame space* (IFS) is defined as the medium free time period prior to the frame transmission. The IFS specified in PCF is smaller than that in DCF, which implies the higher priority of the traffic controlled the PCF compared with DCF. With this polling algorithm, PCF can be described as a contention-free access method with shorter transmission delay, which make it be suitable for the time critical services, like voice or video transmission. In order to adopt to different kinds of services, DCF and PCF often co-exist by alternating the coordination functions in different time periods, called *contention period* (CP) and *contention-free period* (CFP) respectively. The cooperation of these two functions is illustrated as Figure 3.5.

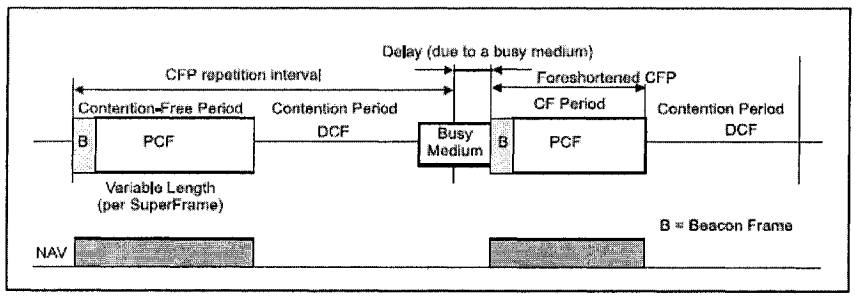


Fig. 3.5 The alternation of DCF and PCF functions [1]

3.2 Overview of User-Mode Linux (UML)

UML [11] is a way to run Linux in user space. Users can engage in various privileged experiments within a UML. This feature is useful for running buggy software or engaging in potentially invasive experiments. The UML can also provide a virtual machine with more hardware and software virtual resources than the physical machine. The virtual disk storage is only a single file located in the host and the hardware access right can be totally controlled with proper configurations.

3.3 Overview of the existing GINI toolkit

The already existing GINI toolkit has the capacity to establish a virtual wired network, which is composed of UML terminals, the *UML Virtual Switches* (UVSs) and GINI routers. The GINI router is virtual device composed of a set of modules [2].

Chapter 4

Design of the Wireless GINI

The major focus of this thesis is to develop a wireless extension on top of the existing wired GINI system. One of the major design issues with wireless emulation is synchronization. With introduction of the MAC and physical layers, synchronization becomes a crucial factor for efficient network emulation. The situation is further complicated because GINI follows a software-only approach to emulate the wireless protocols. This makes a precise implementation of the synchronization harder in the emulator.

4.1 Layered structure of the GINI system

For the wireless GINI system to handle all possible events that can happen in the WLAN, proper layers specified by the wireless networking standards should be developed, including the MAC layer and the physical layer, and wireless propagation channels. The purpose of this design is to integrate the UMLs, the already existing virtual network devices, and the newly developed wireless extension altogether in a layered structure.

To stay consistent with the layering paradigm used in TCP/IP model, the wireless GINI treats the propagation channel as the primary layer of the system as well. The system is organized as a set of *basic functional layers* with associated *modules*. The basic functional layers contain common functions that are shared by the specific functions running within the modules.

The wireless GINI involves three basic functional layers: the *MAC layer*, the *physical layer*, and the *channel*. Nine modules implements the complete functionalities of an ad-hoc WLAN, across these three layers. An overview of the overall structure is given in Figure

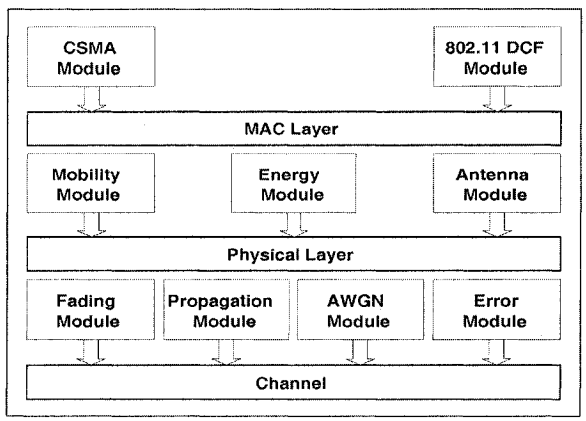


Fig. 4.1 The layered structure of the wireless GINI

4.1.

4.2 Components of the GINI system

Figure 4.2 shows the GINI achitecture in more detail. The basic GINI elements are *GINI virtual physical layer interface* (VPL interface), *GINI internal stack*, *GINI external stack*, and *GINI channel*. The non-basic GINI elements are formed by different combinations of GINI elements and are divided into two categories: the *End Nodes* that act as the terminal stations in the network and the *GINI Nodes* which take the role of the virtual network devices. An End node is composed of the UML and the GINI external stack bridged by the VPL interface, whereas a GINI node only consists of an independent GINI internal stack.

4.2.1 The VPL interface of the GINI system

Currently UML can support only the MAC layer and above. Therefore, the wireless GINI provides an external stack to emulate the functions of the lower layers as needed to make the End node fully functional. The VPL interface establishes a connection between the UML and GINI external stack, allowing these two independent processes to exchange packets.

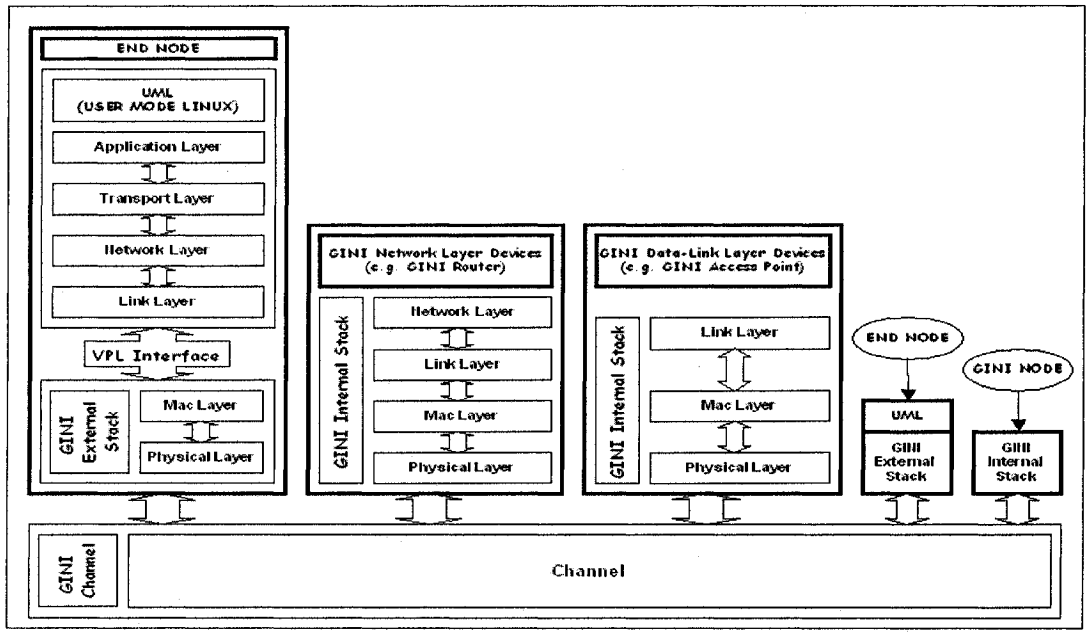


Fig. 4.2 Components of the GINI system

4.2.2 The external stack of the GINI system

As a part of the End node, the external stack mainly provides MAC and physical layer functions to handle the traffic from the UMLs. The basic functions embedded in both layers are used for time management, frame encapsulation and decapsulation, and control of the *service access points* (SAPs) to the adjacent layers. The modules offered by the GINI toolkit or developed by the users can be plugged into the basic functional layers for the purpose of a specific network experiment. A more complete description of the modular structure is given in Chapter 5.

4.2.3 The internal stack of the GINI system

The internal stack of the GINI is the backbone of the GINI node. The layers and the modules embedded in the GINI internal stack can be different depending on the virtual network device the GINI node represents, shown as Figure 4.2, it could be totally different when it acts as a network layer device such as GINI router, or a data link layer device such as the GINI access point. Any new GINI virtual device can be easily obtained by the new

combination of the layers and the modules. This flexibility allows application extensibility in the future.

4.2.4 The GINI channel

As the shared propagation medium of the traffic from the End Nodes and GINI nodes, the channel performs the propagation delay calculation for every individual physical frame shipped in the tunnel. The actual physical features of the different propagation mediums are defined in the corresponding modules. Different functions will be called while the signals are transmitted over the medium with different propagation characteristics.

4.3 Structure of the emulated virtual WLAN in GINI

In the wired GINI system, the UML acts as the wired terminal and it can only handle the functions above MAC layer currently. Therefore, instead of being directed to the lower layers, the traffic flow is terminated at the *logic link control* (LLC) layer, as shown in Figure 4.3. In order to support a complete wireless network emulation, a *wireless GINI management center* is introduced. But the management center and the links between the different processes are totally invisible to users. Figure 4.4 gives a clear view of all virtual components in a emulated WLAN.

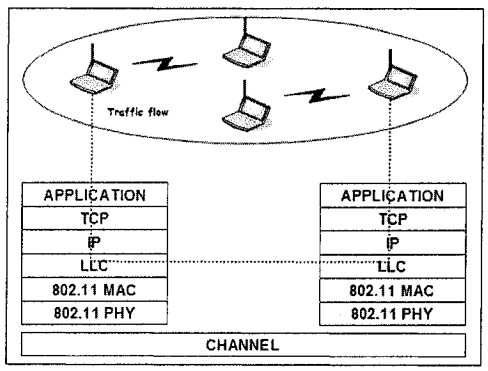


Fig. 4.3 Traffic flow in the wired GINI system without the external stack

A WLAN in GINI is composed of UMLs, UVSs, VPL interfaces, and a wireless GINI management center. The UVSs (Section 3.3) is used to forward the traffic from the UML

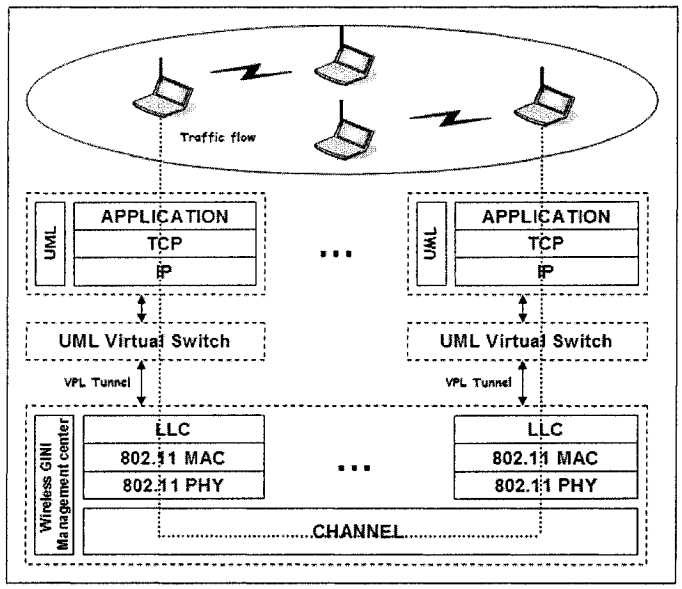


Fig. 4.4 The typical WLAN structure in wireless GINI

to the WGN management center and the VPL interface connects the virtual switch to the management center to provide a TCP tunnel between the two processes.

The WGN center consists of the shared wireless GINI channel, the wireless GINI external stacks (Section 4.2.2), and the LLC layers of all nodes. The external stack of each node is independent but centrally managed by the center. The LLC layer is a sub layer of data link layer and in charge of the frame encapsulation, decapsulation , the *address resolution protocol* (ARP), and the buffer management functions.

Currently only 802.3 ethernet interfaces are supported by the UML, IP packets from the network layer, therefore, are encapsulated with 802.3 frame header by the UML. In order to make the UML compatible with the wireless network, a wireless LLC layer that follows the 802.11 standard is developed in the wireless GINI. The detailed implementation is described in Section 4.6. The ARP function can be directly taken from the LLC layer of the UML, because there is no difference between the wired and wireless networks at this point. Furthermore, two buffers called *LLC to MAC* and *LLC to UML* are embedded in the LLC layer for scheduling frame transmissions. They stores frames from LLC layer to MAC layer and from LLC layer to the UML, respectively. The buffer sizes can be changed for different experimental purposes. Each buffer has its own mutex to prevent the buffer

from read or write collisions. The internal structure of the LLC layer is illustrated in Figure 4.5.

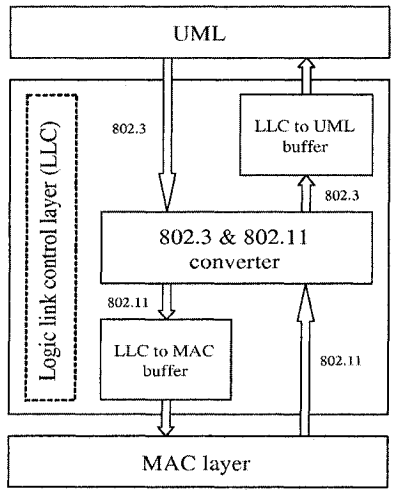


Fig. 4.5 The internal structure of the logic link control layer

4.4 Time management of the GINI system

4.4.1 The maintenance of the GINI system time

In wireless GINI, a event-based scheduler (Section 4.5) is implemented to handle all possible events. A uniform time is needed by the scheduler so that the events generated by different UMLs can be executed at the right time. However, each UML in the virtual network has its own clock with different time accuracy. Therefore, a global clock is deployed to maintain the system time for all virtual mobile stations and network devices.

The system time is defined as a data structure with two fields: the seconds field and the minimum resolution field. Normally the default value of the minimum resolution is ten nanoseconds. The minimum resolution of the timer represents the maximum possible error. Its value can be reset by changing the macro variable of *MIN_RES_FACTOR* defined in the program, but an unsuitable selection of this value may cause low efficiency or high error problems.

4.4.2 Time control by different timers

Six kinds of timers are implemented in wireless GINI for different controlling purpose, which are *Back-off timer*, *Defer timer*, *NAV timer*, *Interface timer*, *Tx timer*, and *Rx timer* [3]. They are used for tracking a variety of state transitions of the whole system. In GINI toolkit, two kinds of data structures are established, one is *back-off timer* used by the Back-off timer and the other, called *normal timer*, is shared by the rest of the timers. The back-off timer is distinguished from others because of its different functions and operations.

Table 4.1 The data structure of back-off timer

Fields of back-off Timer	Description
switcher	shows the on/off status of the timer
pause	shows whether the timer is "paused" or not
start_time	shows the start time of the timer
remain_time	shows the remaining time of the back-off process
wait_time	shows the time period the node should wait before decreasing the back-off time counter
timeout_time	the time to stop the back-off timer

Table 4.2 The data structure of normal timer

Fields of normal timer	Description
switcher	shows the on/off status of the timer
start_time	shows the start time of the timer
end_time	shows the end time of the timer

The detailed deployments of these timers are given in Chapter 5.

4.5 Event management of the GINI system

The software-based emulation of the MAC and the physical layers is a big challenge because it may cause a long processing delays. Unlike the hardware which can be informed of the channel states indicated by the changeable signal power, the software can only do it by checking whether there is some frame or not in the corresponding sending or receiving

buffer. For extremely small minimum time resolution, continuous sensing of the buffers can bring unacceptable delay. Experiments showed that the processing time in software with the default resolution of $10ns$ costs seventy times more than that in hardware.

In order to reduce this gap between the software emulation and the hardware solution, an central event scheduler was developed in the wireless GINI. It is obvious that the system state will not be changed without the execution of any event. Therefore, instead of polling system states at regular intervals, the checking procedure is only activated by the scheduler once some event is executed, and thus boosts the overall efficiency significantly.

4.5.1 Management of the event queue

A global event queue, called *event list*, is used by the event scheduler to centrally manage the events generated by all nodes in the network. The events in this queue are sorted by the associated execution time; that is, the event with the earliest execution time will be placed at the head of the queue. Three basic functions are offered by the event scheduler:

Add a new event: A new event can be created by any virtual device. Once an individual event is created, it is inserted at the right place of the event queue.

Delete an event: The event is deleted from the queue after the execution.

Destroy an event: The event is destroyed without execution.

All the events operations are controlled by an *event-queue-mutex* to prevent the event queue from the potential queue chaos problem. No process or thread is allowed to change the content of the event queue until it obtains the mutex.

4.5.2 Individual event generation

An event data structure is defined by the wireless GINI with a set of fields to identify an event from others. Its fields describe the event characteristics and provide sufficient information when the event is handled by the system. The following are the available fields:

id is the identifier of the mobile station which is the executor of the event.

Event_type is the type of the event that can result in different handling process.

Event_Time shows the event execution time.

Event_Object is a pointer pointing to the void format object (e.g. frame) attached with the event. In the case of the absence of this field, a NULL pointer can be passed as the input value.

Next_Event_Ptr is the pointer pointing to the next event.

4.5.3 Enumeration and description of the event types

All the available event types provided by wireless GINI toolkit are defined in the program. The users are allowed to add new event types for their own purpose. The following are the available event types:

BFTimeOut: This event happens once the back-off process is completed. It stops the back-off timer and invokes the back-off timeout event handling process.

DFTimeOut: This event happens once the deferral process is completed. It stops the defer timer and invokes the deferral timeout event handling process.

NAVTimeOut: This event happens once the *Network Allocation Vector* (NAV) process is completed. It stops the NAV timer and invokes the NAV timeout event handling process.

IFTimeOut: This event happens once the frame transmission process is completed. It stops the interface timer and execute the NAV timeout event handling process.

TXTimeOut: This event happens when the transmitter get no response from the receiver after waiting for a fixed time period. It stops the TX timer and invokes the TX timeout event handling process.

RXTimeOut: This event happens once the frame receiving process is completed. It stops the Rx timer and invokes the Rx timeout event handling process.

INTTimeOut: This event happens once the corresponding interference expires. It invokes the interference timeout event handling process.

START-RX: This event happens once the first bit of the frame arrives at the receiver. It invokes the receiving process of the frame attached with the event.

4.5.4 The capacity estimation of the event scheduler

If we ignore the collision events, the total number of the events generated by the virtual network is proportional to the number of the frames created by the mobile nodes. For a bandwidth-fixed channel, the total throughput is limited by the channel capacity. Therefore the maximum number of the frames that can be handled in a specific time period remains on a relative stationary level no matter how many mobile stations are running simultaneously.

Based on the current network scalability, the rate of collisions stays low. Therefore, it can be predicted that the event scheduler will not be overloaded with the increased number of mobile stations.

4.6 Design of the protocol converter

As described in Section 4.3, a *protocol converter* is needed to exchange the frames between the 802.3 ethernet and the 802.11 wireless network standards. In the future, the GINI toolkit will upgrade the kernel of the UML to one that supports the 802.11 standard and can run in ad-hoc mode. This upgrade will allow frames to use the 802.11 format directly and the converter can be discarded. The 802.11 format frame generated by the protocol converter, called *virtual 802.11 frame* is not the same as the real 802.11 one. The virtual 802.11 frame is composed of the frame header and the frame body with its own data structure. The virtual frame header includes all the information embraced in the real one. In addition, the information that will be used for the virtual frame handling process is contained as well, such as the transmitted and received power of the frame, the propagation delay, and the frame error information. The frame body is the IP layer packet which can be easily obtained by decapsulating the received 802.3 frame which is from the UML.

It should be noticed that the MAC layer and physical layer share the same virtual frame header. Instead of adding the frame headers one layer by the other, GINI just sets the relevant fields in the shared virtual frame header at the corresponding layer. The whole frame format transform procedure is illustrated in Figure 4.6.

4.7 Statistical calculation of the GINI system

Data analysis is important for evaluating the performance of the networks. Therefore, a powerful statistics module is provided by the wireless GINI. It has the capability to record

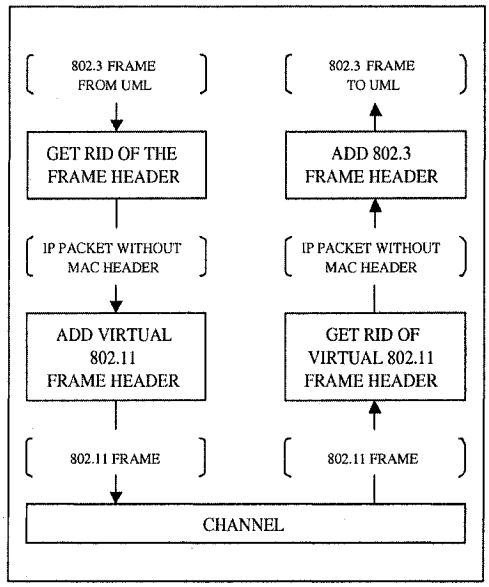


Fig. 4.6 Conversion steps taken between the two standards

the important data during the traffic transmission process and make some calculations. All the statistical experiment results can be output as a file, which can be in different format based on the requirement. The commands to generate the related output file can be found in the documentation provided with the GINI toolkit.

Chapter 5

Implementation of the wireless GINI

The purpose of this chapter is to give a detailed description of the implementation of the wireless GINI. Based on the design presented in Chapter 4, my thesis research focused on the implementation of the external GINI stack of the wireless node and the GINI propagation channel. On top of the external stack and the channel a set of functional modules are developed to make it possible to emulate a complete WLAN working in the ad-hoc mode.

5.1 Implementation of the GINI channel

The channel plays a crucial role in wireless network emulation. Unlike wired channels that present predictable performance characteristics, the performance of the wireless channels heavily depend on the environment through which the radio signals travel. The environment can distort or fade the radio signals at different levels based on the wireless channel condition. Following the GINI layered structure described in Section 4.1, The GINI channel consists of a basic functional channel and four modules: *Propagation module*, *Fading module*, *AWGN module*, and *Error module*.

5.1.1 Implementation of basic functional channel

The basic functional channel is used to handle the general propagation delay of signals. Whatever the channel type, the propagation delay of the transmitted signal can always be

calculated based on the distance between of the T-R pair by using the equation below:

$$Propgation\_Delay = Distance/WaveSpeed \tag{5.1}$$

where *WaveSpeed* is the speed of the signal transmitted over the channel. Normally it equals to the speed of light, which is $3 \times 10^8 m/s$, for wireless channel and 2/3 of this for wired cable. In wireless GINI, a three-dimensional space called movement space is assumed to model the mobility of a mobile station. This space can be defined by setting the range value for the three coordinate axes. The mobility pattern and the current location for each mobile station is managed by the mobility module of the GINI system (Section 5.2.2) such that the value of the T-R distance can be easily obtained by consulting it. The receiver will start receiving a packet only after a period of time equal to the propagation delay has passed since the first bit of the corresponding packet was sent by the transmitter. The packet receiving procedure is activated by the START_RX event at the receiver side in wireless GINI. The START_RX event is created by the transmitter at the beginning of the packet transmission and the event execution time is set to the propagation delay after its creation.

5.1.2 Implementation of the channel modules

Four channel modules were developed in wireless GINI to emulate two typical wireless channels, which are *Additive White Gaussian Noise channel* (AWGN channel) and *Rayleigh Fading channel.* Signals transmitted over different channels are affected by different factors. Generally, the factors impacting the transmitted signal can be classified as the *large scale fading*, the *small scale fading* and the *environment noise.* The statistics of large-scale fading are used to estimate the path loss as a function of the T-R distance. It traditionally focuses on prediction of the average power strength of the received signal. The small-scale fading is caused by the interference of many scattered signals, called multi-path waves, arriving at an antenna at slightly different time. The combination of these waves can result in rapid and violent changes on the strength and the phase of the original signal. Therefore the small-scale fading always refers to the rapid fluctuation of the signal amplitude and phase. The environment noise is described as AWGN noise in GINI emulation. Different factors result in different power gains on the signal shown as below:

Large Scale Fading :

- *Distance dependent path loss gain*, represented as g_p
- *Shadowing gain*, represented as g_s

Small Scale Fading :

- The *multi-path fading gain*, represented as g_m

Environment Noise :

- *Additive white gaussian noise*, represented as $n(t)$

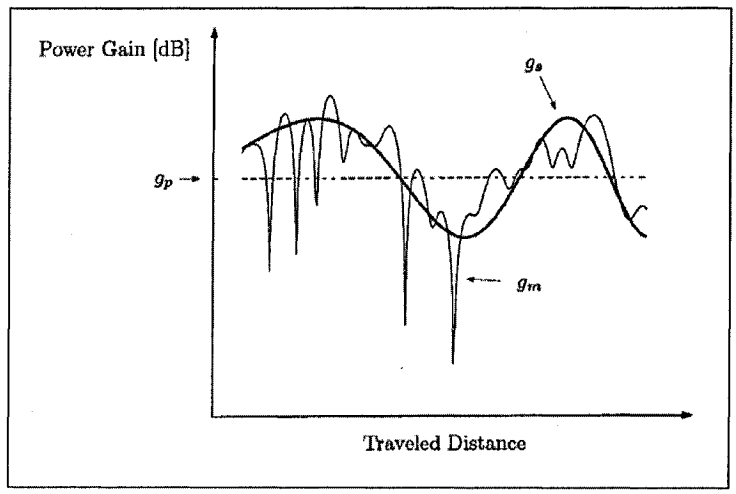


Fig. 5.1 The propagation power gain of the radio wave

In wireless GINI, the different channels can be described by different combinations of the power gain factors and it is the basis for dividing the channel modules. The propagation module is in charge of computation of large-scale fading while the fading module is responsible for the small scale fading process. The error module is used to provide the *bit error rate* (BER) caused by AWGN noise to the AWGN module, which handles frames affected by AWGN noise.

The AWGN channel model is defined as the addition of white gaussian noise to the transmitted signal, which can be described as the following:

$$r(t) = s(t) + n(t) = g_p g_s I(t) + n(t) \tag{5.2}$$

where the $r(t)$ and $s(t)$ are the received signal with and without the noise contribution, respectively. $I(t)$ is the original signal transmitted by the transmitter. From the above equation, it is clear that the AWGN channel can be emulated by the combination of the propagation module and AWGN module.

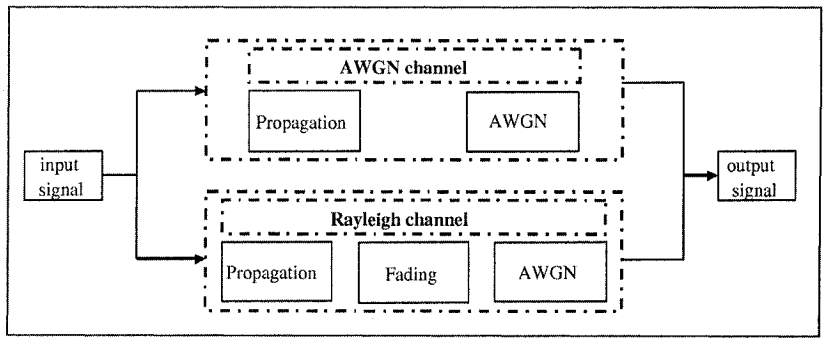


Fig. 5.2 Typical channel structures composed by different modules

The Rayleigh fading channel model takes the multi-path fading effect of transmitting over a distance into consideration and it is defined as a product of the transmitted signal with a complex fading coefficient, which can be represented as:

$$r(t) = ae^{j\theta}s(t) + n(t) = g_p g_s g_m I(t) + n(t) \tag{5.3}$$

where $ae^{j\theta}$ is the complex fading coefficient of the signal. By comparing the equations of two different channel models, it can be concluded that the only difference between them is that the Rayleigh fading channel model takes the multi-path fading gain into consideration. Therefore, it can be emulated by simply adding the fading module based on the AWGN channel.

(a) Propagation module

Three mathematical models provided by the propagation module are used to estimate large scale fading gain for the signal. These models are called large scale fading models because they can estimate the coverage area of the mobile transmitter and describe the changes of the average received signal strength between the T-R pair. Taking the transmitted power of the signal as the input parameter of the propagation module, the average received power can be calculated based on the distance derived from the mobility module (Section.5.2.2) corresponding to the power of $s(t)$ in Equation(5.2) and Equation(5.3). So far, three propagation models are implemented: the *free space model*, the *two-ray ground reflection model*, and the *shadowing model*.

Free space model

The free space propagation model is used to predict the average received signal power while the channel can be characterized as an unobstructed line-of-sight path.

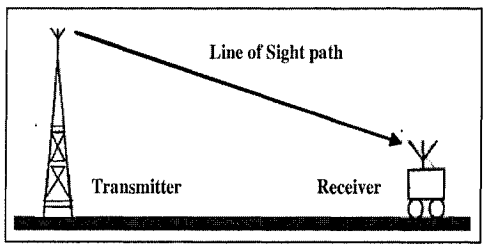


Fig. 5.3 Free space model

The received signal strength can be obtained by the following expression called *Friis free space equation* [12]:

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \tag{5.4}$$

where P_t is the transmitted signal power while P_r is the received signal power. G_t and G_r are the antenna gains of the transmitter and the receiver respectively. L is the system loss factor which is related to the physical characteristic of the antenna. λ is the wavelength in meters and d is the distance between the transmitter and the receiver. The value of L, G_t and G_r can be obtained by invoking the functions provided by antenna module (Section.5.2.2) for the specific type of antenna.

Two-ray ground reflection model

A single direct LOS path between two mobile nodes seldom provides an accurate representation of the physical radio channel. The two-ray ground reflection propagation model that considers both the direct and reflected propagation paths is more precise for the long T-R distance cases compared with the free space model. The received signal power can be expressed as below [12]:

$$P_r(d) = P_t G_t G_r \frac{h_t^2 h_r^2}{d^4 L} \tag{5.5}$$

where h_t and h_r are the antenna height of the transmitter and the receiver in meters respectively. L is a constant to make the equation consistent with the one for the free space model and it is always been set to 1.

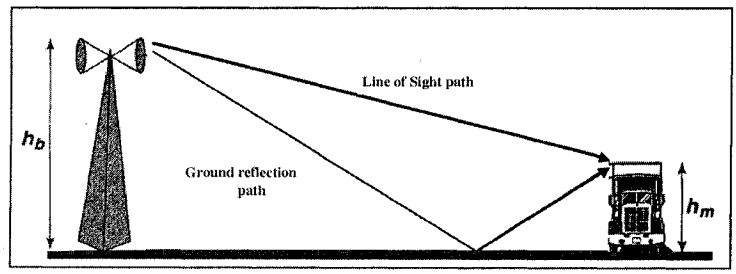


Fig. 5.4 Two-ray ground reflection model

However, the two-ray ground model cannot give a satisfying estimation of the received power for short distance because of the oscillation caused by the constructive and destructive combination of the two rays while the free space model is actually suitable for it. Thus, a cross-over distance, represented as d_c, is defined to separate the regions of applicability of two models and it is calculated as below [12]:

$$d_c = \frac{4\pi h_t h_r}{\lambda} \tag{5.6}$$

For the case of $d < d_c$, free space equation (Equation(5.4)) should be used, otherwise two-ray ground equation (Equation(5.5)) is used.

Shadowing model

Both of the above propagation models take the T-R distance as the only variable to predict the average received signal power which leads to an ideal circle communication range for each mobile node. Obviously this phenomenon seldom happens in real situations. The shadowing model takes the difference of surrounding environmental clutter with the same T-R distance into consideration, which results in the following equation [12]:

$$P_r(d)_{dB} = P_r(d_0)_{dB} - 10\beta \log_{10}(\frac{d}{d_0}) + X_{dB} \tag{5.7}$$

where β is called the path loss exponent. X_{dB} is a zero-mean gaussian distributed random variable with standard deviation of σ_{dB}, which is called the shadowing deviation. The values of both β and σ_{dB} are empirically determined by field measurement. d_0 is a close-in reference distance and it may be set to one meter for most cases.

The shadowing model adds some statistical features based on the ideal models, which results in the possibility of failing in real communications while the node moves to the edge of the radio range. The first term of the right side of the Equation(5.7) is the path loss gain while the second one is the shadowing gain in dB scale. For the two models mentioned before, only the path loss gain is taken into consideration and the shadowing gain is totally ignored.

(b) Fading module

The fading module is used to emulate the small-scale fading of the wireless channel. The small-scale fading can be described as Rayleigh fading if the multiple reflective paths are large in number without LOS signal component. In this case, the envelope of the received signal can be described as a Rayleigh distribution. It can be called as Rician fading if there is a presence of a dominant non-fading signal component, such as a LOS propagation path. And the envelope of the received signal can be described as a Rician distribution. In practical cases, the presence of a line-of-sight propagation path for the wireless signal transmission is rare, therefore, only Rayleigh fading is implemented in GINI toolkit.

In the Rayleigh fading channel, the stochastic model for the complex low-pass envelope

of the received signal is a complex gaussian process, which is shown as below [13]:

$$\mu(t) = \mu_1(t) + j\mu_2(t) = ae^{j\theta} \tag{5.8}$$

where $\mu_1(t)$ and $\mu_2(t)$ are independent zero-mean stationary Gaussian processes with identical variances:

$$\sigma^2 = \sigma_1^2 = \sigma_2^2$$

and a and θ are the amplitude and the phase of the received complex low-pass envelope respectively. Based on the assumption of the stationary gaussian process, the absolute value of the low-pass envelope of the received signal follows the Rayleigh distribution while the phase obeys uniform distribution, which can be represented as below [13]:

$$f_a(x) = \begin{cases} \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} & , x \geq 0 \\ 0 & , x < 0 \end{cases} \tag{5.9}$$

$$p_\theta(\alpha) = \frac{1}{2\pi} \quad , -\pi \leq \alpha \leq \pi \tag{5.10}$$

In the GINI emulation, Jakes' simulator is used to approximate the Rayleigh fading. The technique uses N_0 low-frequency offset oscillators plus one central carrier frequency generator to duplicate the mobile radio spectrum. Its structure can be shown as Figure 5.5. The expressions of the real and the imaginary parts of the low-pass envelope can be represented by the following equations [14]:

$$\mu_1(t) = 2\sum_{n=1}^{N_0} \cos\beta_n \cos\omega_n t + \sqrt{2}\cos\alpha\cos\omega_n t \tag{5.11}$$

$$\mu_2(t) = 2\sum_{n=1}^{N_0} \sin\beta_n \cos\omega_n t + \sqrt{2}\sin\alpha\cos\omega_n t \tag{5.12}$$

where

$$\omega_m = 2\pi\frac{\nu}{\lambda}, \quad \omega_n = \omega_m \cos\frac{2\pi n}{N}, \quad N_0 = \frac{1}{2}(\frac{N}{2} - 1), \quad \beta_n = \frac{n\pi}{N_0}$$

In the above equations, N is the number of the multiple paths considered by the emulation and N_0 is the number of the low-frequency oscillators in *Jakes' simulator*. The bigger

the value of N_0 is, the lower error difference this approach could be. ω_m is the maximum *Doppler shift* in terms of angular frequency and ω_n is the Doppler shift for each propagation path. By choosing $\alpha = 0$, we find that $\langle \mu_1 \mu_2 \rangle \equiv 0$ and $\langle \mu_1^2 \rangle = N_0$, $\langle \mu_2^2 \rangle = N_0 + 1$. Thus the simulated low-pass envelope is a narrow-band signal centered on a carrier frequency ω_c with the characteristics of Rayleigh fading.

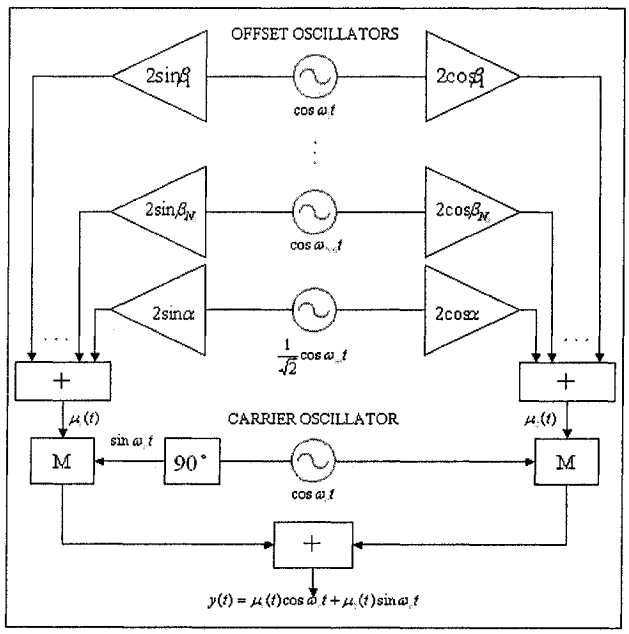


Fig. 5.5 Oscillator structure of Jakes' simulator

(c) AWGN noise module

The conceptions in AWGN module

Errors can occur in the frame demodulation procedure because of the interference of the AWGN noise. The AWGN module is responsible for the frame error rate checking in the AWGN channel. The mixture process of the signal and the AWGN noise is illustrated in Figure 5.6.

The $n(t)$ is the AWGN noise with zero mean and standard deviation of σ^2. The power

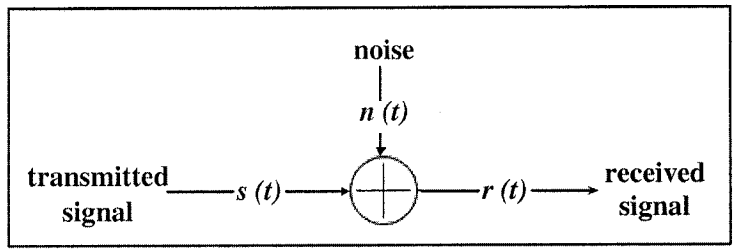


Fig. 5.6 Structure of the AWGN channel

spectrum density of the noise is a constant, which follows the relationship as below:

$$S_N(f) = \frac{N_0}{2} = \sigma^2 \tag{5.13}$$

where $S_N(f)$ is the power spectrum density of the noise and N_0 is a constant in scale of W/Hz. The value of σ can be changed anytime by the user.

The calculation of the frame error rate

The frame error rate depends on the noise power added into the signal and the modulation method which is adopted by the transmitter. In WLAN, the physical layer frame header is always transmitted on 1 Mbps using DBPSK, which has been specified by the 802.11 standard (Section 15.2.3 in [1]) and the *MAC protocol data unit* (MPDU) part is transmitted on the chosen transmission rate among 1, 2, 5.5 and 11 Mbps corresponding to DBPSK, DQPSK, CCK5.5 and CCK11 modulation scheme respectively. Based on the modulation information of the frame, the frame error probability can be determined. It is related to the bit error rate and the number of the bits that the frame embraces.

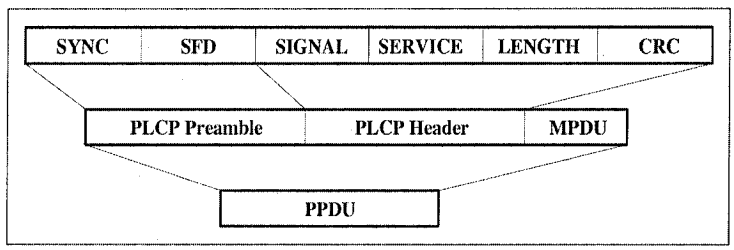


Fig. 5.7 Physical layer frame structure in the IEEE 802.11 standard

The modulation scheme used by the MPDU of a frame can be different from that used by the physical frame header. This results in the separate frame error probability calculation for these two parts. As shown in Figure 5.7, the physical frame header is composed of *Physical Layer Convergence Protocol* (PLCP) Preamble and the PLCP Header. So the error probability of the physical layer frame header, represented as $P_{e\_phy\_hdr}$, can be calculated using the following equation:

$$P_{e\_phy\_hdr} = 1 - (1 - P_{b\_DPBSK})^{LEN_{phy\_hdr}} \tag{5.14}$$

where P_{b_DPBSK} is the bit error probability for DPBSK modulation and LEN_{phy_hdr} is the total length of the physical layer frame header in bits. The value of LEN_{phy_hdr} is the sum of the PLCP preamble and PLCP header length. For the different spread spectrum methods used, FHSS or DSSS, the length of the frame header can all be different.

Table 5.1 Physical frame header length

Spread Spectrum Method	Preamble Length (bit)	PLCPHeader Length (bit)	Total Length (bit)
DSSS	144	48	192
FHSS	96	32	128

Table 5.2 MPDU length for different types of frame

Type	Mac Header Length (byte)	CRC Length (byte)	Frame Body Length (byte)	MPDU Length (byte)
DATA	30	4	payload length(unfixed)	34+payload length
RTS	16	4	0	20
CTS	10	4	0	14
ACK	10	4	0	14

The error probability of MPDU, represented as P_{e_MPDU} can be obtained by a similar way as well:

$$P_{e\_MPDU} = 1 - (1 - P_{b\star})^{LEN_{MPDU}} \tag{5.15}$$

where $P_{b\star}$ is the bit error probability for specific modulation used to transmit the MPDU of the frame. The LEN_{MPDU} is the length of the MPDU in bits, which is a variable depending

on the frame type. In the wireless GINI emulator, four types of frames are implemented, which are RTS, CTS, ACK ,and DATA frames. Their structures are shown as below:

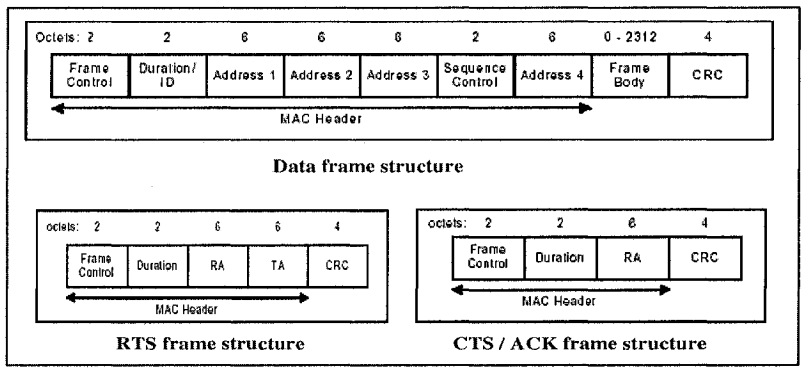


Fig. 5.8 MAC layer frame structure defined in standard 802.11

Based on the frame structure information, the MPDU length can be easily obtained as shown in Table 5.3. The payload that appears in this table is the IP packet received from the network layer. Using the analysis above, the probability of correct reception of a whole frame can be calculated as the successful rate that both the physical header and the MPDU are demodulated without corruption. Therefore, the frame error probability can be represented as [15]:

$$P_{e\_frame} = 1 - P_{successful} = 1 - (1 - P_{e\_phy\_hdr})(1 - P_{e\_MPDU}) \tag{5.16}$$

Frame error marking process in AWGN module

Following the Equation(5.16), the frame error rate caused by AWGN noise can be calculated by the AWGN module and a decision can be made on the correctness of the packet. This procedure follows four steps:

- Obtain the frame error rate from the AWGN module.
- Get a random number uniformly distributed over [0,1].
- Compare the value of the random number with the frame error rate.
- Mark the frame as erroneous if the random number happens to be less than the value of the error rate.

Error difference caused by AWGN module

There are two factors that can affect the performance of the AWGN module and cause error difference.

- The frame error checking procedure in AWGN module is implemented by calling the function, *UnitUniform()* provided by wireless GINI. It generates the uniformly distributed number over [0,1] by randomly selecting a number less or equal to 0x7fff and then divided by 0x7fff. Obviously it can only ensure the precision of $3.052 \times (10^{-5})$. It therefore cannot handle the cases with high SNR that is big enough to make the frame error rate below the minimum resolution.

- In the calculation of the frame error probability, the BER is derived from the error module. The theoretical BER for specific modulation scheme is based on the assumption that there is no interference other than AWGN noise. The appearance of the receiving collision breaks the assumption and the statistical distribution of the total noise may change. Therefore, the BER provided by the error module is not applicable for this case. However, the collision seldom happens (normally order of 10^{-2}) under the control of the 802.11 DCF function. Even if it happens, the signal can only be accepted and demodulated while its power strength is larger than the total interference including noise and collision frames by a threshold (normally set to be $10dB$ or higher). Once this condition is satisfied, the BER caused by the SNR at this level is relative low (at the order of 10^{-5} for DBPSK and 10^{-3} for DQPSK) and can be ignored.

(d) Error module

The error module is used to provide the BER caused by AWGN noise based on the modulation method and the Signal-to-Noise Ratio (SNR). The additive noises mixed into the signal can impact the detection and demodulation of the received signal if it reaches a considerable level. The SNR per bit describes the relationship between the signal and the noise power defined by:

$$SNR = \frac{E_b}{N_0} = \frac{Average\_Energy\_Per\_Bit}{Density\_Spectral\_Power\_of\_Noise} \tag{5.17}$$

It is important to note that bits belonging to the same frame are assumed to have equal average power. The idea behind it is even with the maximum size (i.e. 2370 bytes for DSSS mode), the transmission delay of the physical frame is still less than $10^{-2}s$ with the channel bandwidth of $2Mbit/s$. And the maximum propagation delay is less than $10^{-6}s$ because of the extremely high wave speed ($3\times10^{8}m/s$) and the relatively small radio range (normally less than $250m$). A frame therefore only experiences a negligible delay from the beginning of transmission at the transmitter side to the complete reception at the receiver side. The power vibration of the noise can be ignored in such a small time interval. With higher bandwidth, the performance is even better because of the shorter transmission delay. Given the value of the SNR, the BER can be calculated following the literatures of [16, 17] shown as below:

Table 5.3 Bit error probability of demodulation

Modulation method	Bit Error Probability Coherent Demodulation	Bit Error Probability Non-coherent Demodulation
DBPSK	$2Q(\sqrt{\frac{E_b}{N_0}})$	$\frac{1}{2}e^{-\frac{E_b}{N_0}}$
DQPSK	–	$Q(\sqrt{\frac{E_b}{N_0}})$
BPSK	$Q(\sqrt{\frac{2E_b}{N_0}})$	–
FSK	–	$\frac{1}{2}e^{-\frac{E_b}{2N_0}}$

5.2 The physical layer implementation of the external GINI stack

The physical layer defined in the 802.11 standard can be divided into two sublayers, which are the *physical layer convergence protocol* (PLCP) sublayer and the *physical medium dependent sublayer* (PMD). The PLCP sublayer defines a method of mapping the IEEE 802.11 MPDUs into a framing format suitable for sending and receiving user data and management information. It simplifies the PHY service interface to the IEEE 802.11 MAC services while the PMD sublayer is busy with modulation. In the physical layer implementation of external GINI stack, the functions related to the modulation in PMD sublayer are ignored. But the PLCP sublayer which works with the DSSS mode are taken into account. During transmission, the MPDU shall be prepended with a PLCP preamble and header to create the *PLCP protocol data unit* (PPDU), which is called encapsulation, and at the

receiver side, the PLCP preamble and header are processed to facilitate in demodulation and delivery of the MPDU, which is called decapsulation.

Following the GINI layered structure described in Section 4.1, The physical layer consists of a basic functional physical layer and three modules: *Mobility module*, *Antenna module*, and *Energy module.*

5.2.1 Implementation of basic functional physical layer

The basic functional physical layer is responsible for the analysis of the physical layer frame header, the frame transmission and reception.

(a)Structure of the physical layer frame header

The physical layer frame structure, defined by the DSSS PLCP sublayer, is composed by the DSSS PLCP Preamble, the DSSS PLCP Header, and the MPDU. MPDU is the frame passed directly from the MAC layer and the remaining two parts, which are combined to the physical layer header, should be added. The PLCP Preamble contains the following fields: Synchronization (Sync) and Start Frame Delimiter (SFD). The PLCP Header contains the following fields: IEEE 802.11 Signaling (Signal), IEEE 802.11 Service (Service), LENGTH (Length), and CCITT CRC-16. The concrete structure is shown in Figure 5.9.

Fig. 5.9 The PLCP frame structure defined by DSSS PLCP sublayer

The definitions and the meanings for these fields and the counterpart variables in the pseudo-header of the GINI frame are described as below [1]:

PLCP Synchronization (SYNC): This field is used for synchronization and it is ignored by GINI implementation. The synchronization of GINI system is realized by a central system timer management (Section 4.4.1).

PLCP Start Frame Delimiter (SFD): It is used to indicate the start of the frame and ignored in GINI.

PLCP IEEE 802.11 Signal (SIGNAL): It indicates the modulation method that is used for transmission and reception of the MPDU. The real data rate equals to the field value multiplied by 100 kbit/s.

- 0x0A for 1 Mbit/s DBPSK
- 0x14 for 2 Mbit/s DQPSK

There is also a signal field in the pseudo-header of the GINI frame and its definition follows the 802.11 standard exactly. But so far only unique data rate is supported by GINI, it is therefore reserved for the future application.

PLCP IEEE 802.11 Service (SERVICE): It is reserved for future use and there is no implementation in GINI.

PLCP Length (LENGTH): It is an unsigned 16-bit integer that indicates the number of microseconds required to transmit the MPDU. There is a *mac-size* field that shows the length of MPDU in bytes in the pseudo-header of the GINI frame, the time that is needed to transmit the MPDU then can be obtained by dividing the mac-size by the data rate.

PLCP CRC (CCITT CRC-16): It is used for error checking. The basic object handled by the GINI system at the physical layer is the frame. Therefore, it is only relevant to know whether a frame has errors or not. Knowing where the error happens within the frame and which bit is mistakenly translated is irrelevant. In the GINI toolkit, instead of the bitwise calculation of the CRC field, the *error-mark checking procedure* is used and it is illustrated in the next section.

Transmission of the frame

Before transmitting the frame, the fields of the GINI pseudo-header of the physical layer frame should be filled. As mentioned in the previous section, instead of the real bits checksum calculation, the CRC process in GINI is replaced by the error-mark checking procedure. The frame will be marked with different types of error-marks indicating the occurrence of specific error condition. The error-marks can be classified as *colErrorSign*, *fadErrorSign*, and *noiseErrorSign*, which are listed as below:

Transmission collision error: Frame reception happens while the receiver is transmitting, the frame should be corrupted because of the collision. The colErrorSign should be set while transmission collision happens.

Reception collision error: Two or more frames are being received simultaneously. The frame should be corrupted by the collision. The MAC state is set to MAC_COLL while reception collision happens and no frame can be accepted when the MAC layer is in this status.

Fading error: The frame should be marked as fadErrorSign while its received average power is below the *reception threshold* (RX_Threshold), which is a physical parameter of the wireless network card.

Noise error: It is caused by the low SNR and the noiseErrorSign should be set.

In the GINI emulation, the fadErrorSign is set before the transmission while the remaining three error-marks are handled in the receiving process. The way to mark the fadErrorSign is based on the assumption that the propagation delay ($< 10^{-6}s$) can be ignored. The transmission process of the physical layer follows the procedures below:

- Duplicates a copy of the frame for each destination in the WLAN other than itself.
- Compute the predicted receiving average power of the frame based on the channel fading information and pass the result to the *rx_pwr* variable in its pseudo-header.
- If the value of rx_pwr is lower than the RX_Threshold of the wireless card, the fadErrorSign of the frame should be set.
- The frames sent onto the channel should stay for a propagation delay before starting the receiving process at the receiver.

Reception of the frame

On the receiver side, the frame is handled based on the principles below:

- The execution of the START_RX event can activate the reception of the frame.

- If the received signal power is lower than the carrier sense threshold, its power should be added into the interference and an INTTimeOut event will be generated in order to remove the interference after the time period of the transmission delay of the frame. The time delay can be easily obtained from the bandwidth of the wireless card interface and frame size. Then the total power of the interference should be rechecked and the receiving state of MAC layer may be changed to MAC_RECV if it becomes higher than the carrier sense threshold.

- If the received signal power is higher than the carrier sense threshold, the receiving state of MAC layer should be checked first. If the MAC state is MAC_RECV already, the reason for making it busy should be derived. If it is caused by the interference (the power not contributed by the frame in the receiving buffer should be treated as interference), the new incoming packet will be treated as interference as well and goes to the interference handling procedure mentioned in the previous step. If caused by a frame receiving process and the *capture effect option* (Section 6.1.3) is *off*, the MAC layer state will be switched to MAC_COLL. If the capture effect option is *on* and difference decibels between the signal power of receiving frame and that of the new incoming is bigger than the *capture threshold* (CP_Threshold), the new incoming packet will go to the interference handling procedure described in the previous step, while the receiving packet can be accepted correctly. Otherwise, the MAC layer state will be switched to MAC_COLL.

- Any frame other than the ones treated as interference will be passed to the MAC layer.

5.2.2 Implementation of physical layer modules

(a) Antenna module

There are several kinds of antennas in practical use. But only the simplest antenna named *Omni-directional antenna* has been implemented in GINI so far. The more complicated antennas, such as directional and adaptive ones, may be developed in the future. An omnidirectional antenna is an antenna system which radiates maximum power uniformly in all directions. The perfectly omnidirectional antenna is the isotropic antenna, a theoretical

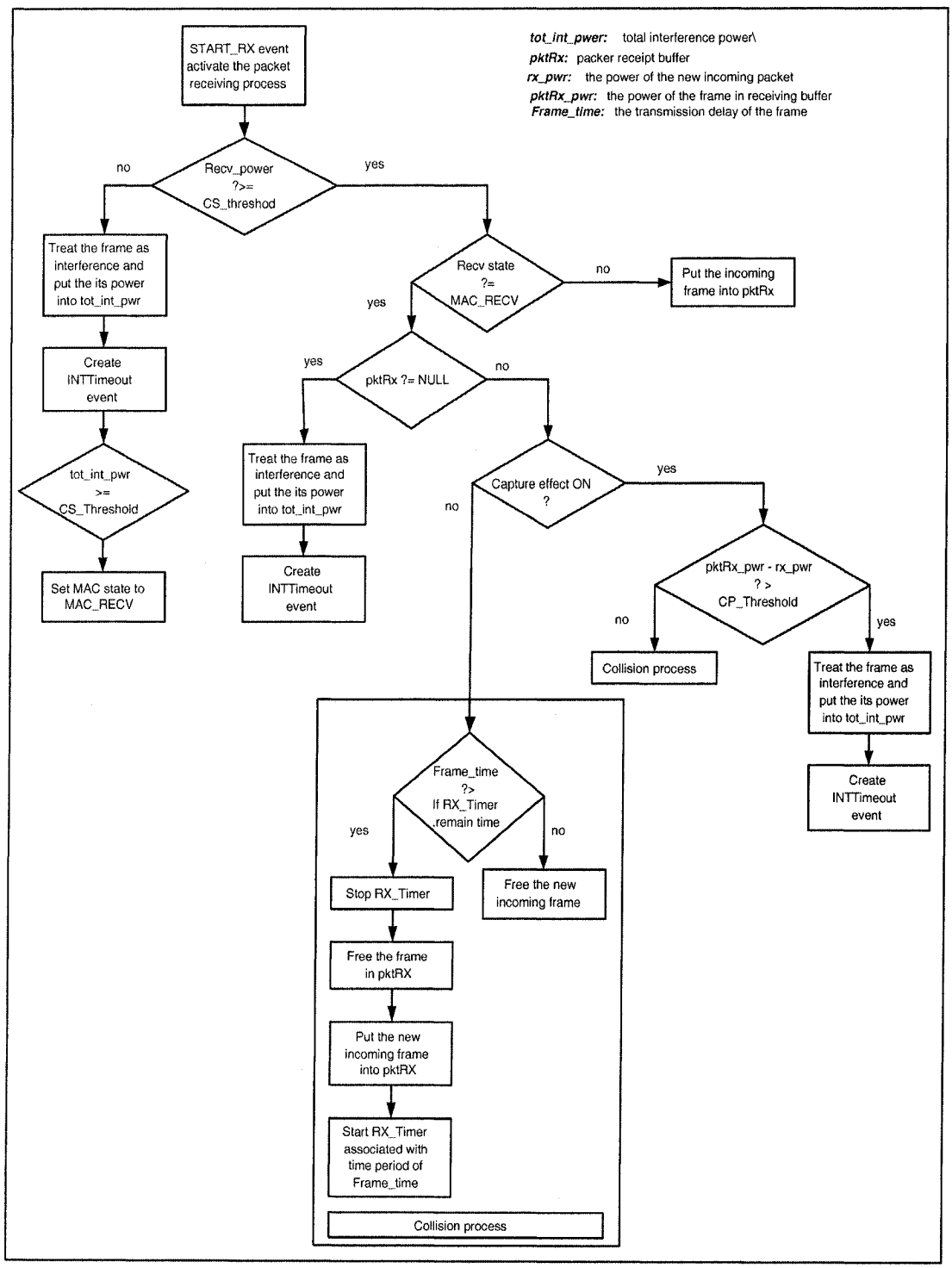


Fig. 5.10 The frame receiving procedures of the physical layer

construct derived from actual antenna radiation patterns and used as a reference for specifying antenna gain and radio system's effective radiated power. The key variables defined to describe the characters of the antenna in GINI include the height, the gain and the system loss. Their values can be configured by users freely and used by the propagation module to calculate the average frame reception power.

(b) Energy module

The energy problem is much more crucial in the wireless networks than in wired networks, because the mobile stations are unlikely to get a permanent power supply while a wired node can. Maintaining the longest battery life for the mobile node should be a vital problem in the wireless network. There are five states for each mobile node,including transmitting, receiving, idle, off, and sleeping. The exact amount of the power consumed in unit time depends on the state that the mobile station is actually in. The parameters of *Pt_consume*, *Pr_consume*, *P_idle*, *P_off*, and *P_sleep* in the data structure *nodeWCard* in GINI are defined to correspond to the power consumed during different states. The sleeping state is only used in the *power saving mode* (PSM). The basic motivation behind this mode is that the power wastage is dominated by keeping active in the idle state. Therefore, if the mobile node can "fall asleep" during the idle time, the energy can be greatly saved. However the implementation of this algorithm needs an efficient scheduler to centrally synchronize the whole network by some MAC-level management frame transactions, which may introduce the necessity of the AP [18].

This thesis only focuses on the Ad-hoc mode wireless network, in which there is no AP. But current version of the energy module can provide some rudimentary functions for power consumption calculation so that the extension can be easily implemented in the future.

(c) Mobility module

This module is used to emulate the random movement of the mobile station. In this module the *Random Way Point* mobility model [4] is used. In this model, a three-dimensional space is established with the maximum coordinator size defined by the user. The whole mobile process follows the procedures below:

- A node is allocated a random location coordinate that can be represented as (x, y, z) in the initialization of the network.
- A random speed is selected for each node from $[MINSPD\ MAXSPD]$, where MINSPD and MAXSPD are the minimum and maximum speed for the mobile node, respectively.
- Randomly choose a destination for the current movement and the node direct to it.
- The whole mobility system is updated periodically with a predefined time and the previous steps are repeated if the mobile node has reached the destination of the current movement.

5.3 The MAC layer implementation of the external GINI stack

Following the GINI layered structure described in Section 4.1, The MAC layer consists of a basic functional MAC layer and two modules: *802.11 DCF module* and *CSMA module.*

5.3.1 Implementation of basic functional MAC layer

The basic functional MAC layer is used to provide the buffer management function that is compatible for different types of medium access control mechanisms. Four one-frame buffers are created for different purposes. Two of them are for transmitting packets while others are for receiving packets.

RTS buffer: It stores the RTS frame during the frame transmission process while it works in RTS mode.

TX buffer: It stores the DATA frame during the frame transmission process.

RSP buffer: It stores the CTS and ACK frames that are response for the RTS and DATA frames from the transmitter, respectively, during the frame transmission process.

RX buffer: It stores all kinds of frames received from other mobile stations during the receiving process.

Other than these one-frame buffers, a multi-packet buffer, called MAC buffer, is implemented serving the retransmission mechanism. It is used to store all the correctly received frames for duplication checking purposes. Each incoming frame that cannot be found in the buffer will be simply inserted into the buffer if there is some empty space left or will overwrite the oldest frame which is already there. This buffer size can be freely defined by the user for different experimental purposes. The default value is set according to the 802.11 standard.

5.3.2 Implementation of MAC layer modules

(a) 802.11 DCF module

MAC states description

Taking the NS-2 simulator [3] as reference, a set of MAC layer states for the mobile node are defined in the GINI system to trace the frame transmission process:

Table 5.4 Classification of the MAC states

Node phase	Mac state	Description
sender	MAC_RTS	sending the RTS frame out and waiting for the CTS frame response from the receiver
	MAC_SEND	sending the DATA frame out and waiting for the ACK frame from the receiver
receiver	MAC_ACK	sending the ACK frame out
	MAC_CTS	sending the CTS frame out and waiting for the DATA frame from the sender
	MAC_COLL	collision happens while receiving the frames
sender | receiver	MAC_IDLE	no sending and receiving actions at the network interface

CSMA mechanism defined in 802.11 standard

Physical and virtual carrier-sense functions are used by the physical layer and MAC layer respectively to determine the state of the medium. The medium should be treated as busy

whenever either of the functions indicates the busy state of the medium.

The physical carrier-sense mechanism provided by the physical layer senses the signal strength of the frames sent to or received from other mobile node and other environmental or artificial interference.

Virtual carrier-sense mechanism provided by the 802.11 MAC layer can predict the future traffic on the medium based on the duration information that is announced in RTS/CTS frames prior the actual exchange of data, force the mobile station into the NAV process, and keep silent during the whole frame transmission process.

Carrier-sense is a combination of the two above mechanisms. It takes the NAV state and the physical channel state into consideration to determine the state of the medium.

In GINI implementation, the NAV timer should be turned on in NAV process. So the channel is announced to be free only under the condition that both the receiving and transmitting states are *idle* and the NAV timer is *off*.

MAC-level frame generator

The MAC-level frames, used for the medium access control, should be terminated at MAC layer while the received frames meant for the upper layers can be forwarded to the network layer by the SAP. So far, three types of MAC-level frames, which are RTS, CTS and ACK frames, can be created to facilitate the implementation of the RTS-CTS mechanism in wireless GINI.

Back-off procedure and implementation

The back-off procedure is used to reduce the chance of collisions among the packets injected by the mobile nodes and prevent the deterioration of the overall network performance. The back-off procedure can be invoked by a failure transmission or by the busy state of the medium while the mobile station is ready to send a frame out. The selection of the back off time should follow the equation:

$$Time = RandomNumber \times SlotTime \tag{5.18}$$

where the $RandomNumber$ is a pseudorandom integer drawn from a uniform distribution over the interval $[0, CW]$. The definitions of CW (contention window size) and the

SlotTime are described in Section 5.3.2.

In the GINI implementation, the back-off timer is used to control the back-off process. There are six fields in the data structure of the back-off timer which is described in Section 4.4.2. There are four different operations binding with the back-off timer: Start, Pause, Resume and Stop which may be fulfilled by the function group: *BFTimer_Start ()*, *BFTimer_Pause ()*, *BFTimer_Resume ()* and *BFTimer_Stop ()*.

BFTimer_Start() is used to start a back-off timer and set the fields of the timer data structure as Table 5.5. The back-off timer may be started after the medium is determined to be idle without interruption for a time period equal to *DCF interframe space* (DIFS) when the last frame detected on the medium was received correctly, or for a time period equal to *extended interframe space* (EIFS) when the last frame detected on the medium was received incorrectly. The random back-off time period shall be generated as an additional deferral time before transmitting, unless the back-off timer already contains a nonzero value. The selection of the random back-off time follows Equation(5.18).

Table 5.5 Back-off timer start configuration

Fields of Back-off Timer	Value
switcher	ON
pause	OFF
start_time	use GetTime() to get current system time
remain_time	randomly choose a time period based on the current CW
wait_time	0.0
timeout_time	start_time + remain_time

The back-off time can only be decremented while the channel is indicated as idle and the minimum decreasing time period is one SlotTime, called *back-off time slot.* Once the medium is determined to be busy at any time during a back-off slot, that slot shall not be decreased. The back-off procedure is then suspended and the function BFTimer_Pause() shall be called, which leads to the deletion of the previously established BFTimeOut event as well. The following equation is used to calculate the total reduced time to be counted out from the remain time of the back-off timer:

$$reduced\_time = current\_time - (start\_time + wait\_time) \tag{5.19}$$

But the last uncompleted time slot should not be counted into the decremented time, so the real decreased time should be:

$$real\_reduced\_time = floor(reduced\_time \div slottime) \times slottime \tag{5.20}$$

Please note that if the idle state of the channel is disturbed during the DIFS or EIFS waiting time period, as appropriate, the real_reduced_time could be a negative value. In this case, it may be set to be zero. After the pause action of the back-off timer, the parameters of the timer should be:

Table 5.6 Back-off timer pause configuration

Fields of back-off timer	Value
switcher	ON
pause	ON
start_time	N/A
remain_time	remain_time - real_reduced_time
wait_time	DIFS
timeout_time	N/A (timeout event is canceled)

Whenever the channel is indicated to be idle, the back-off timer should return to the back-off process by calling the function of BFTimer_Resume() and the related parameters shall be recalculated as in Table 5.7. The value of the wait_time depends on the immediately preceding medium-busy event. If it is caused by the detection of an erroneous frame, it may be set to be EIFS, otherwise DIFS. Each time the back-off timer is stopped by the reception of a new incoming frame, the function BFTimer_Pause() can be called, which may set the wait_time to the default value of DIFS after the computation of the back-off time period decrement. A complete received frame which is decided to be an erroneous one can result in changing the wait_time to be EIFS. Reception of an error-free frame during the EIFS can resynchronize the node to the actual busy/idle state of the medium, so the EIFS is terminated and the pause action activated by this error-free frame can reset the wait_time to be DIFS. In this way, normal medium access continues after the reception of that frame. BFTimer_Resume() may regenerate a BFTimeOut event based on the information of the timeout_time field of timer.

BFTimer_Stop() is a function that simply destroys the back-off timer and invokes the frame transmission process immediately.

Table 5.7 Back-off timer resume configuration

Fields of back-off timer	Value
switcher	ON
pause	OFF
start_time	use GetTime() to get current system time
remain_time	N/A
wait_time	DIFS / EIFS
timeout_time	start_time + remain_time + wait_time

Retransmission procedure

The retransmission mechanism is only applicable for the transmitter, not the receiver. This means the acknowledgement, which includes the CTS and ACK frames, can never be retransmitted while the RTS and Data frame can. The condition to invoke the retransmission

Table 5.8 Timeout period for different type of frame

Frame type	Timeout value
RTS	$RTS\_Time + CTS\_Time + 2 \times SIFSTime$
DATA	$Data\_Time + ACK\_Time + SIFSTime + DIFSTime$

is the absence of the response from the receiver after a specific timeout period (Table 5.8). The timeout event can be caused by the deep fading of the channel, out of communication range problems, or frame collisions. The value for the timeout period depends on the type of the frame that is about to be sent, which is shown in Table 5.8. In the table, The $SIFSTime$ and $DIFSTime$ are the short interframe space and DCF interframe space respectively, whose values are defined by the 802.11 standard (Section 9.2.10 in [1]) clearly.

$$SIFSTime = RxRFDelay + RxPLCPDelay + MACProcessingDelay + RxTxTurnaroundTime \tag{5.21}$$

$$DIFSTime = SIFSTime + 2 * SlotTime \tag{5.22}$$

$$SlotTime = CCATime + RxTxTurnaroundTime + AirPropagationTime + MACProcessingDelay \tag{5.23}$$

In the GINI implementation, the SIFSTime and the SlotTime of the DSSS mode are set to the values shown in the Table 5.9 (Section 15.3.3 in [1]).

Table 5.9 Time value selection in DSSS mode

Variable	Value (s)
SIFSTime	0.000020
SlotTime	0.000010

The RTS_Time, CTS_Time, Data_Time and ACK_Time represent the transmission delay of the RTS, CTS, Data and ACK frames respectively, which can be calculated as below [1]:

$$RTS\_Time = (PLCPPreambleLength + PLCPHeaderLength)/1Mbit/s + RTSLength/bandwidth \quad (5.24)$$

$$CTS\_Time = (PLCPPreambleLength + PLCPHeaderLength)/1Mbit/s + CTSLength/bandwidth \quad (5.25)$$

$$Data\_Time = (PLCPPreambleLength + PLCPHeaderLength)/1Mbit/s + DataLength/bandwidth \quad (5.26)$$

$$ACK\_Time = (PLCPPreambleLength + PLCPHeaderLength)/1Mbit/s + ACKLength/bandwidth \quad (5.27)$$

The *PLCPPreambleLength* and *PLCPHeaderLength* are the PLCP preamble length and the PLCP header length of the physical layer frame respectively and their values can be obtained from Table 5.1. The entire PLCP preamble and header is always transmitted by DBPSK modulated signal at the rate of 1 Mbit/s, which is specified by the 802.11 standard (Section 15.2.3 in [1]). The bit rate used for transmission and reception of the MPDU is indicated by the 8-bit IEEE 802.11 *signal* field in the PLCP header of the real physical

layer frame. The signal field is implemented by the "signal" variable of the pseudo-header of the GINI virtual frame. The real date bit rate is the signal field value multiplied by 100 kbit/s (Section 15.2.2.3 in [1]). In the GINI system, the retransmission process is invoked by TXTimeOut event. The concrete process is different based on the frame length and the timeout frame type, which can be RTS, DATA. The frame is indicated to be long or short one by taking the *RTS_threshold* to be the boundary. That is, it will be treated as long frame if longer than the threshold, otherwise taken as a short one.

There are two parameters defined in the 802.11 standard used to indicate the maximum number of the retry times for long and short frame respectively, which are called *long retry count* (LRC) and *short retry count* (SRC). Before the transmission becomes successful, or the relevant retry limit is reached, the retry counter may be incremented by one for each attempt of the retransmission. The related retry counter should be reset to be zero after the reception of a corresponding acknowledgement. This short retry count shall be reset when a MAC frame with the length less than or equal to RTS_Threshold succeeds for that *MAC service data unit* (MSDU) or *MAC management protocol data unit* (MMPDU) while the long retry count shall be reset when a MAC frame with length greater than RTS threshold succeeds for that MSDU or MMPDU. When either of the retry counters reaches the maximum value, the retry attempts shall cease, and the MSDU or MMPDU shall be discarded, and the following frame can be processed.

The retransmission is a useful recovery procedure to handle frame loss. But there should be a chance that the receiver sends the acknowledgement back but it failed to be accepted by the transmitter. In this case, the received frame duplication can be caused at the receiver side due to the retransmission. To solve this kind of problem, the duplicate detection facilitated by the *sequence control* field of the frame header is introduced by 802.11 standard. This field is composed of a sequence number and fragment number. However, because of the absence of the capacity to handle the fragmentation in current GINI toolkit, the fragment number field is ignored and only the *SourceAddress-SequenceNumber* information pair for each correctly received frame is needed to distinguish itself from others. For each incoming frame, the *fc_retry* field in the pseudo-header may be checked first and only the frames with this field value set to be one will be passed to the duplicate detection process. If its source MAC address and the sequence number matches any one of the entries recorded by the receiver, it will be treated as a duplicated one and dropped, otherwise it should be passed to the upper layer and its information will be inserted into the buffer as

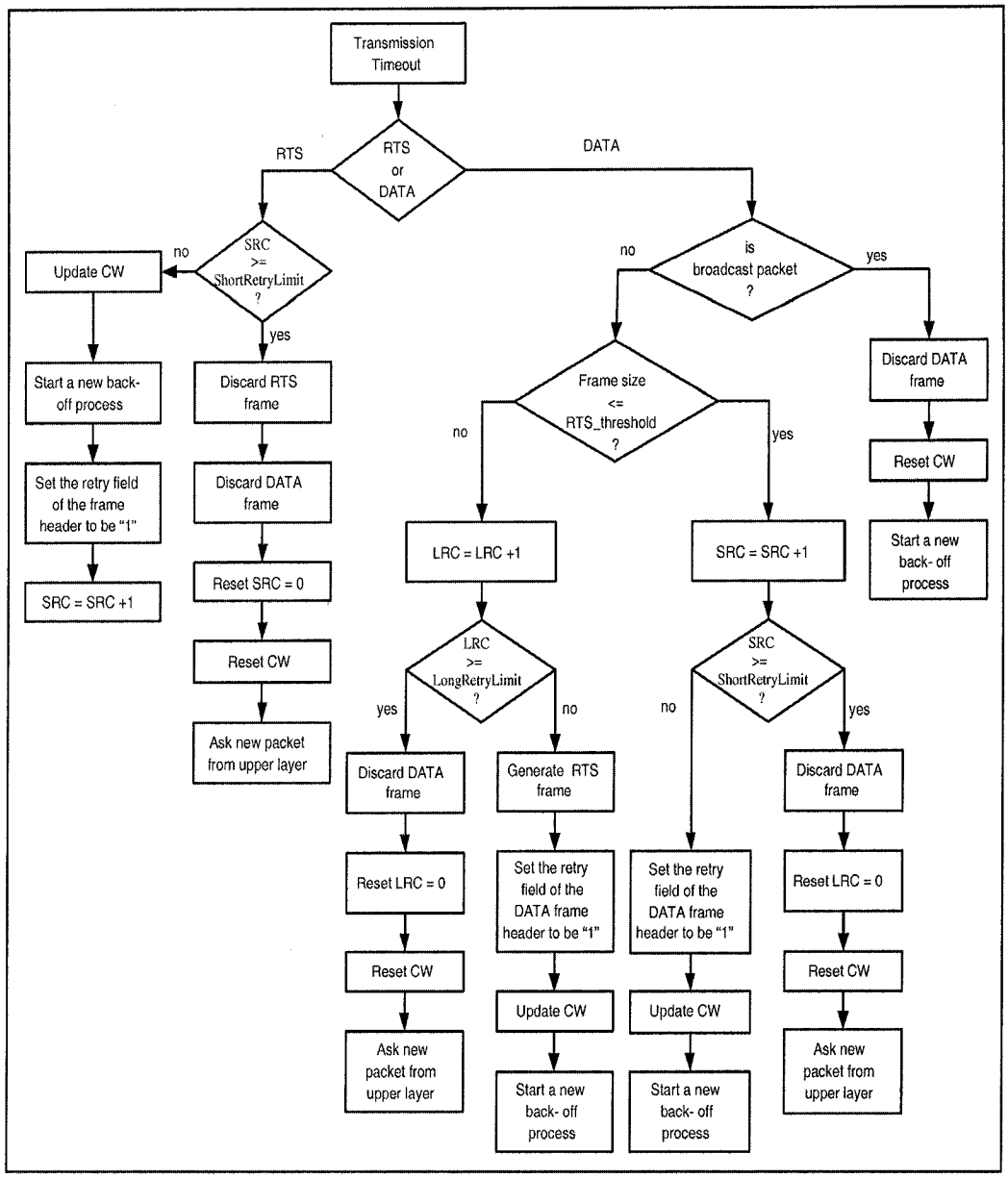


Fig. 5.11 The procedures of the retransmission mechanism

a new entry.

The *contention window size* (CW) is defined to be the upper bound of the random back-off time selection range. The control of the contention window size takes a vital role in the retransmission process. The retransmission event is a negative sign in the network because it is usually caused by the bad condition in the channel or the collision. Obviously a relatively big window size can minimize the probability of the collisions by preventing the mobile nodes from choosing the same back-off time and maximize the selection probability of a relative long back-off time period, which can provide sufficient time to avoid re-sending frames to the network when it is still in temporary overloaded or deep fading state. It can cause more frame loss or other negative impact on the network if the traffic flow keeps coming in before it is relieved. The variable *CW* in the *NodeMac* data structure is responsible for the contention window management in the GINI implementation and its initial value is *CWmin* which is defined by the 802.11 standard as the minimum contention window size. Its value can be different depending on the spread spectrum type shown as Table 5.10 [1]:

Table 5.10 Contention windows size

Emission type	CWmin	CWmax
DSSS	31	1023
FHSS	15	1023
IR	63	1023

The *CW* should be doubled for each unsuccessful attempt to transmit an MPDU until the *CW* reaches the *CW maximum value* (*CWmax*). Once it reaches the maximum value, the *CW* will stay at this value until it is reset. The *CW* is reset for each successful transmission of an MSDU or MMPDU, when LRC reaches *LongRetryLimit* (set to 7 in standard 802.11), or when SRC reaches *ShortRetryLimit* (set to 4 in standard 802.11). As the necessary components of a MSDU transmission, the RTS and CTS frames belong to the MAC-level control frame category, so the *CW* should not be reset after the reception of the CTS frame although it approves the successful transmission of the RTS frame.

Instead of simply repeating, retransmission can engage different handling procedures depends on the frame type. Normally, the failure of DATA frame with the MPDU size larger than the RTS threshold can result in the retransmission of itself associated with an new regenerated RTS frame while only the resending of the failed frame itself is needed

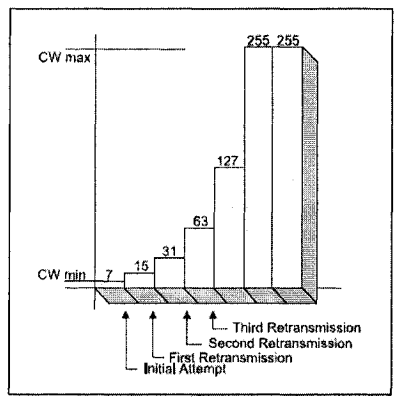


Fig. 5.12 An example of exponential increase of the window size [1]

in case that it is a RTS or the DATA frame with the MPDU size smaller than the RTS threshold.

NAV process description

The mobile node shall update its NAV timer based on the information in the *duration* field of the valid frame that does not address itself. The value of the duration field for different frame types can be calculated as shown in Table 5.11.

Table 5.11 Duration value for different frame

Frame type	Time duration
RTS	$SIFSTime \times 3 + CTS\_Time + Data\_Time + ACK\_Time$
CTS	$SIFSTime \times 2 + Data\_Time + ACK\_Time$
DATA	$ACK\_Time + SIFSTime$
ACK	0 (No fragmentation implementation)

The NAV process may force the node to keep silent without any carrier sense action to guarantee the current communication between the T-R pair as much as it can. Figure 5.13 below shows the impact of the NAV process on the networks.

Frame reception process

The *FCS* field in MAC layer frame is a 32-bit field containing a 32-bit CRC. It is used to check errors in the MAC layer frame. In the wireless GINI, the *MAC layer frame filter*

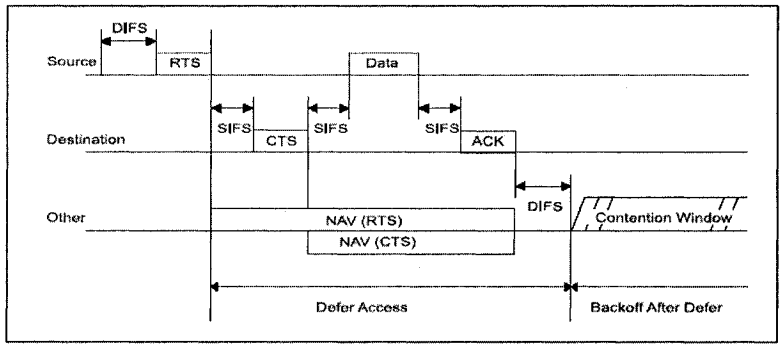


Fig. 5.13 The NAV process in WLAN [1]

is supposed to take the responsibility of the FCS field checking for the received frame and make a decision on whether it is a corrupted one or not. However, like the physical layer error checking process, instead of the instantaneous calculation of the FCS field, the error-mark checking procedure is used here as well. Any frame marked as erroneous by the physical layer will be treated as an incorrect one. The FCS field in the pseudo-header of GINI frame is always set to be zero without any use so far, but it can provide the interface a way to implement the real FCS checking function in the future.

Receiving a frame while transmitting shows a picture of the collision between the bi-directional frames caused by the overlap of the receiving process of the incoming frame and the transmission process of the outgoing frame. But the ending order of these two kinds of processes can lead to distinguished handling procedures although the incoming frame should be marked as colErrorSign in both cases. The difference is that the EIFS time deferral process will be invoked for the case that transmission process ends before the receiving process while not for the other case. The reason to set the deferral period after the reception of the corrupted frame is to provide the sufficient time for resynchronization of the receiver. According to the 802.11 standard, the EIFS interval deferral process shall begin once the medium becomes idle indicated by the physical layer after detection of the erroneous frame. Due to the simplex property of the wireless channel, the mobile station cannot be aware of the exiting of the received frame while it is transmitting in the latter case, no deferral process will be activated and the frame will then be silently discarded.

Errors can also occur in case of receiving more than one frame from different nodes at the same time. To handle this kind of situation, the MAC state may be set to MAC_COLL

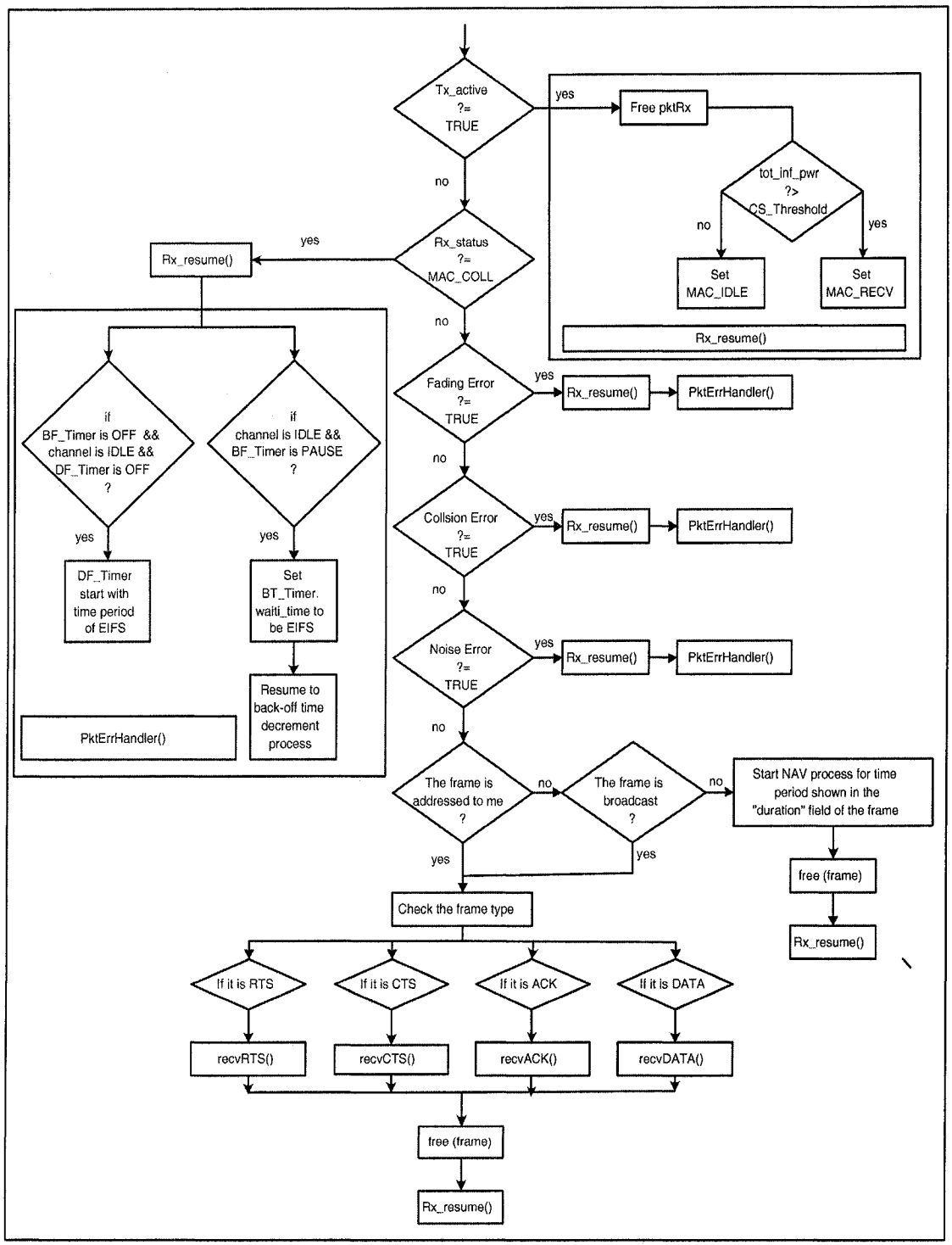


Fig. 5.14 The frame reception process at MAC layer

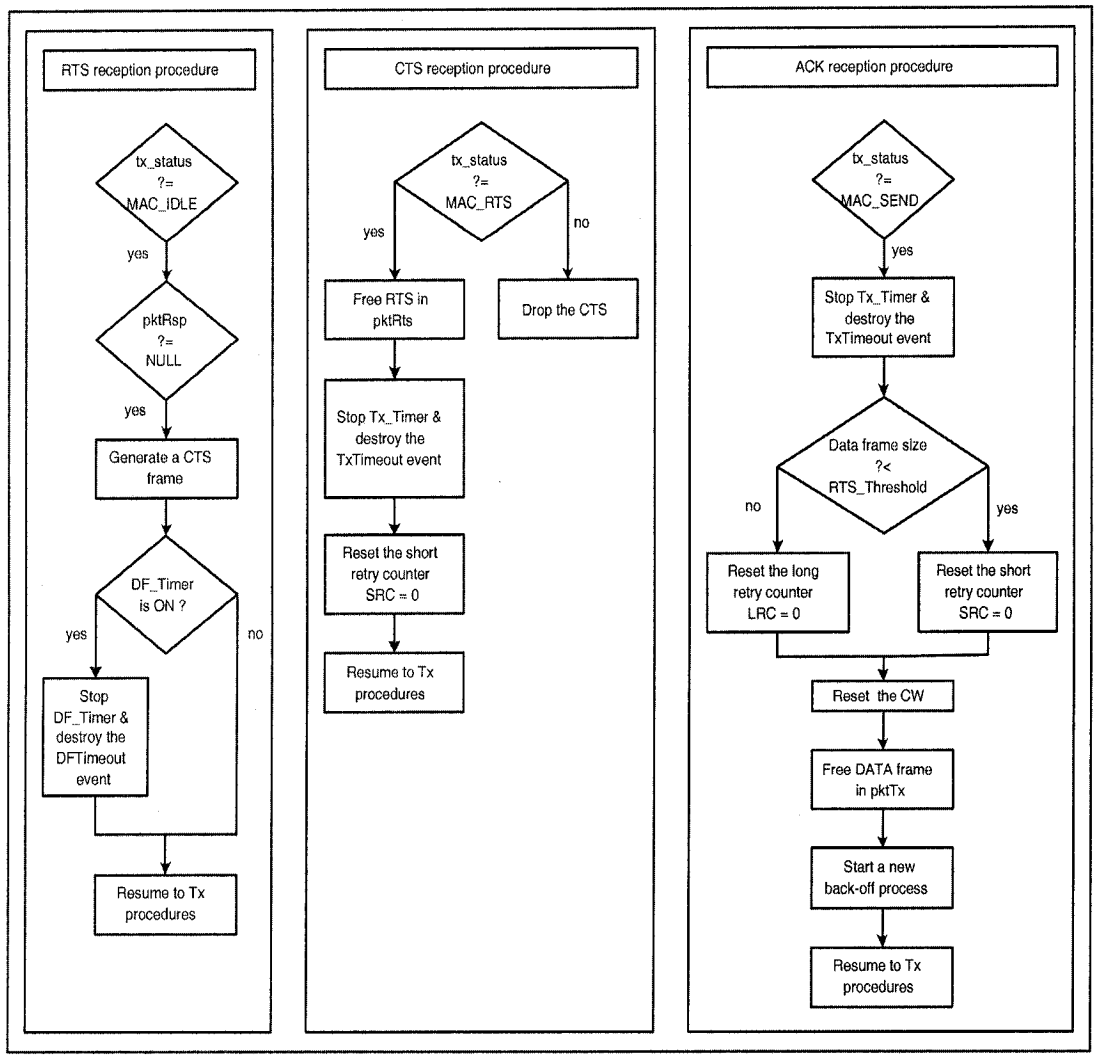


Fig. 5.15 The frame reception process at MAC layer (cont.1)

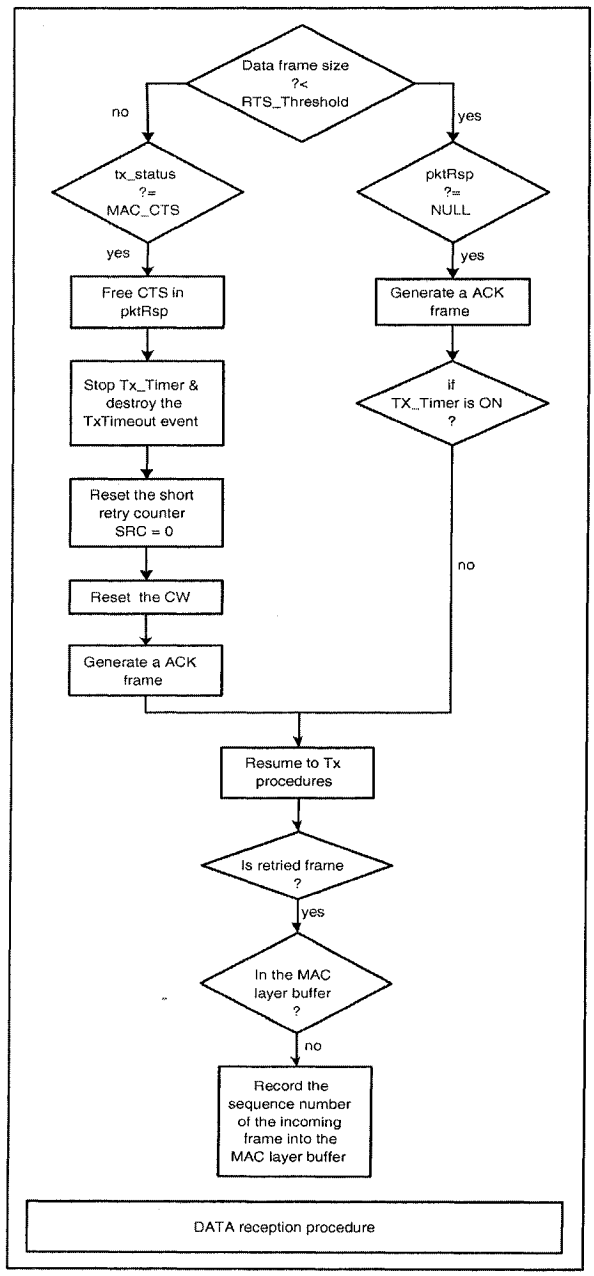


Fig. 5.16 The frame reception process at MAC layer (cont.2)

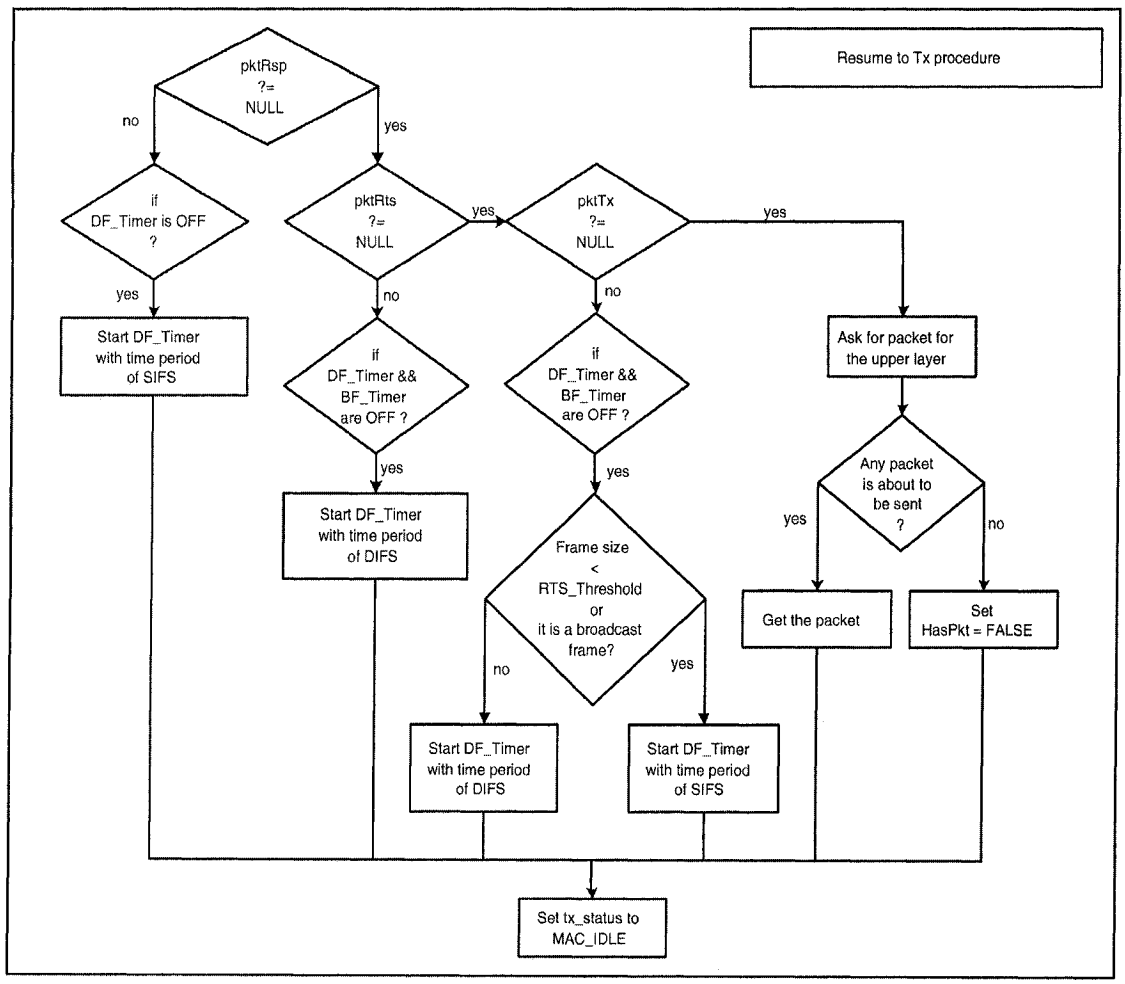


Fig. 5.17 The frame reception process at MAC layer (cont.3)

once the collision happens. Any frame received under this MAC state will be treated as a corrupted one. The latest frame which ends the MAC_COLL state should always be traced. All the collision frames are combined to be a bit stream and the last bit of the latest collision frame should be the starting point for the idle state of the medium, and the EIFS deferral process as well.

The frame may be marked as fadErrorSign while its received power is below the receiving threshold or by noiseErrorSign, depending on the calculating result obtained from the AWGN module based on the signal to noise ratio value. Both of these errors may result in the EIFS deferral process as well. Only the error-free frame can be recognized by the MAC layer.

The correctly received frames may be handled based on the information embraced in itself. The broadcast frames are always accepted as a data packet and passed to the upper layer. For the unicast frame, the first step is to check if is addressed to itself. A negative answer will invoke the NAV process and the frame may be dropped. Otherwise it should be passed to the next procedure based on the frame type.

The concrete handling steps for the whole reception process is described as in Figure 5.14 to 5.17.

(b) CSMA module

The CSMA mechanism implemented in this module is p-persistent CSMA, which means the frame can only be sent out with probability p while the channel is indicated as idle. If it happens to fall into the range outside of p, the mobile station should keep silent for the maximum propagation delay before sensing the channel again. The maximum propagation delay can be calculated as below:

$$Max\_Propagation\_Delay = \frac{\sqrt{X\_Width^2 + Y\_Width^2 + Z\_Width^2}}{WaveSpeed} \tag{5.28}$$

where X_Width, Y_Width and Z_Width are the maximum value for each lateral of the map respectively and the $WaveSpeed$ is the speed of the microwave, which is $3 \times 10^8 m/s$. The waiting process is called CSMA_NAV process in wireless GINI. After the waiting process, the mobile station will sense the channel again. The consecutive carrier sensing will be deployed if the channel is busy, otherwise the previous steps will be repeated. While a

frame transmission timeout event occurs, the resending procedure will be activated before the retry count number reaches the limit, otherwise the frame will be dropped.

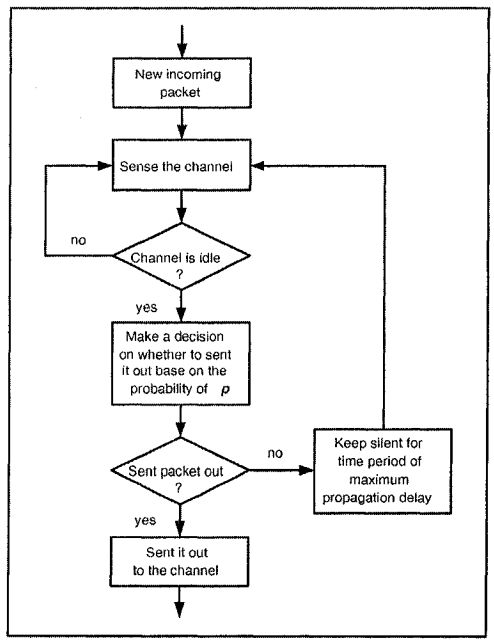


Fig. 5.18 The p-persistent CSMA process

Chapter 6

Validation of wireless GINI

6.1 Validation of MAC layer and the modules attached

6.1.1 Comparison of different medium access control mechanisms

(a)Scenario description

The configuration for this scenario described in Appendix B.1.1 satisfies the following conditions:

- The communication range for each node is limited to a fixed circle with a radius of $265m$ (no shadowing and fading affect experienced).
- Each node can communicate with each other during the whole testing period.
- The data packets may be sent out directly without RTS provided the RTS_Threshold is set to be big enough (in our scenario it is set to 10000); a RTS is attached with the data packet when the RTS_Threshold is 0.

(b)Report analysis

Generalized analysis based on the output file of the GINI can be concluded as follows:

- No packet is dropped due to *LOW RECV POWER* because all nodes are within the communication area of each other, without channel fading and noise affect.

- The collisions can be divided into transmission collisions (*TX Collision*) and reception collisions (*RECV Collision*). The transmission collision happens when the mobile node is sending and receiving frames simultaneously and the reception collision happens when the station is receiving more than one packet from different transmitters at the same time. In case there are only 2 nodes in the WLAN, no RECV Collision can happen because a receiver can not receive more than one frame from different transmitters simultaneously. But the TX Collision can happen because each node acts as both transmitter and receiver. As a file stream receiver, node 2 also sends data which is the TCP level ACK packets generated by TCP layer in response to each correctly received data packet from node 1. These TCP level ACK packets will be encapsulated as a DATA frame at MAC layer. Therefore, it is possible that node 1 receives the DATA frame from node 2 while it is transmitting, which results in TX Collison.

- In RTS mode (RTS_Threshold = 0), the type of all the collision unicast packet should almost be RTS but not DATA (for broadcast packet, RTS is never generated). This provides the evidence that RTS-CTS is working to prevent the nodes from DATA frame collisions.

- Under the 802.11 DCF control, all nodes almost equally share the channel and the transmitting rate of each node equals to the total throughput divided by the total number of the nodes that are using the channel simultaneously. This shows the fairness of the medium access control algorithm provided by the 802.11 standard.

The crucial factors to estimate the performances of the networks can be separated into several categories: throughput, collision rate, and retransmission rate. These aspects are described in the following tables of the experiment's results.

With different medium access control mechanisms, the WLAN performs different. The network throughput mentioned in this thesis is the pure data throughput, which takes the MAC-level management traffic out of consideration. There is a sharp decrement on the throughput for the CSMA and NONE modes when the total number of nodes increase from two to three due to the introduction of the receive collisions. For RTS-CTS and NO-RTS-CTS modes, the total throughput stays at a stable level, which shows the ability of the 802.11 DCF in preventing collisions in the WLAN.

Table 6.1 Total throughput of the WLAN

Mobile STA number	NONE	CSMA (kB/s)			RTS-CTS	NO-RTS-CTS
	(kB/s)	0.1	0.5	0.8	(kB/s)	(kB/s)
2	3.3	197.6	42.9	10.7	285.2	303.3
3	3.1	79.90	20.6	10.0	285.5	302.3
4	2.3	82.40	22.8	16.7	285.8	303.9
5	1.4	80.50	23.2	16.3	286.0	304.7
6	0.4	81.00	24.5	17.0	287.4	303.8
7	0.2	82.10	25.1	18.8	287.1	303.7

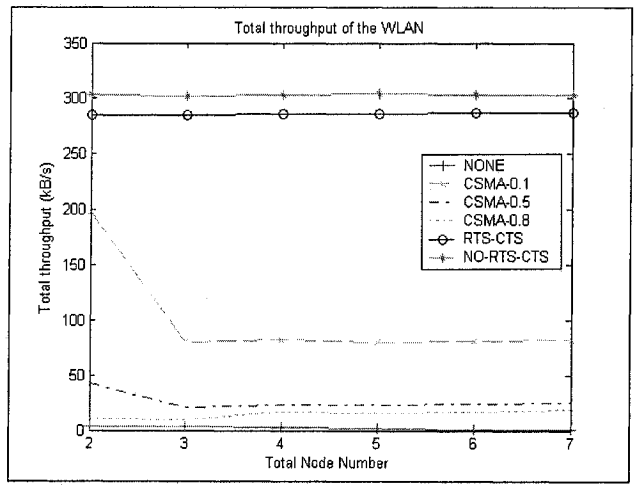


Fig. 6.1 Total throughput of the WLAN

In the scenario described Appendix B.1.1, the only receiver of the file data in this scenario is node 4 while all of others are transmitters. In order to get the most distinguished contrast for different MAC layer mechanisms, we focus on the analysis of the node 4 because it should have the most intensive incoming traffic flow and the highest probability of collisions.

Table 6.2 Transmission collision rate at the receiver

Mobile STA number	NONE	CSMA (%)			RTS-CTS	NO-RTS-CTS
	(%)	0.1	0.5	0.8	(%)	(%)
2	76.9999	3.5930	27.0744	60.3167	0.4236	0.8596
3	50.0063	9.3376	25.5808	33.9104	0.6556	1.3466
4	51.1090	8.2936	21.8712	26.5878	0.8698	1.6177
5	69.6663	7.9416	19.8842	23.1210	0.9171	1.8200
6	73.4319	7.9330	18.0764	21.2106	0.9553	1.8855
7	74.9504	7.7226	17.2541	19.9295	0.9726	1.8956

Table 6.3 Reception collision rate at the receiver

Mobile STA number	NONE	CSMA (%)			RTS-CTS	NO-RTS-CTS
	(%)	0.1	0.5	0.8	(%)	(%)
2	0.0000	0.0000	0.00000	0.00000	0.0000	0.0000
3	5.6272	9.0585	18.0795	21.1251	1.3947	2.7686
4	7.3981	7.9865	17.6327	16.2162	2.0539	4.1184
5	2.1364	8.6084	17.8106	16.3172	2.5276	4.7300
6	2.6610	8.2863	15.9919	14.5540	2.6127	4.9786
7	1.5873	8.3264	16.5570	13.8400	2.6761	5.1425

The retransmissions are invoked by the loss of a frame for some reason. Therefore, high collision rates are always associated with high retransmission rates. The retransmission rate goes up while the total node number in the WLAN is increased from two to three. This step corresponds to the sharp decrement of the total throughput of the network shown in Table 6.8.

NONE and CSMA mode comparison

NONE means no medium access control mechanism in the MAC layer and *CSMA* in our implementation refers to the *p-persistent CSMA* mechanism. Compared with NONE mode,

Table 6.4 Total collision rate at the receiver

Mobile STA number	NONE (%)	CSMA (%)			RTS-CTS (%)	NO-RTS-CTS (%)
		0.1	0.5	0.8		
2	76.9999	3.59300	27.0744	60.3167	0.4236	0.8596
3	55.6335	18.3962	43.6604	55.0356	2.0503	4.1152
4	58.5071	16.2801	39.5039	42.8040	2.9237	5.7361
5	71.8027	16.5500	37.6949	39.4382	3.4448	6.5499
6	76.0929	16.2193	34.0682	35.7647	3.5680	6.8642
7	76.5377	16.0490	33.8111	33.7695	3.6487	7.0480

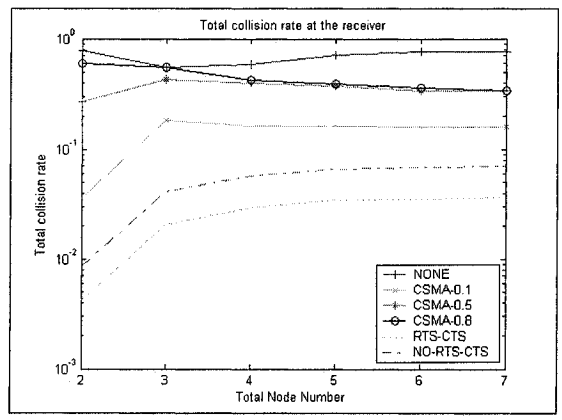


Fig. 6.2 Total collision rate at the receiver

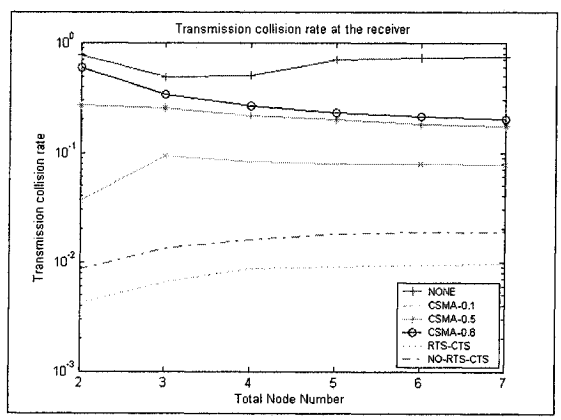


Fig. 6.3 Transmission collision rate at the receiver

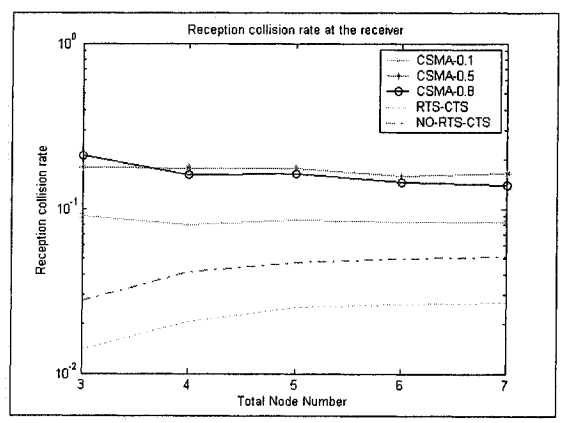


Fig. 6.4 Reception collision rate at the receiver

Table 6.5 Retransmission rate at the transmitter

Mobile STA number	NONE	CSMA (%)			RTS-CTS	NO-RTS-CTS
	(%)	0.1	0.5	0.8	(%)	(%)
2	76.1505	5.31580	36.9448	73.5436	0.6364	1.2899
3	85.1198	30.1996	70.9175	80.0362	3.0712	6.2878
4	81.6063	34.1068	74.4996	81.0239	4.6013	8.9883
5	78.2676	40.4036	75.2471	83.9342	5.2615	9.8640
6	78.7313	45.7211	78.1048	85.3708	5.4891	10.4640
7	78.1736	44.0555	77.5029	85.8092	5.5849	10.7481

Table 6.6 Retransmission rate at the receiver

Mobile STA number	NONE	CSMA (%)			RTS-CTS	NO-RTS-CTS
	(%)	0.1	0.5	0.8	(%)	(%)
2	82.0611	9.98000	50.2929	74.6501	1.2507	2.5123
3	83.1088	19.6735	46.4876	60.0841	1.9648	4.0246
4	80.2569	17.8658	46.7636	50.1244	2.6064	4.8599
5	82.7541	18.1883	46.5832	49.5002	2.7334	5.4557
6	83.4472	18.8428	46.5886	49.5404	2.8478	5.6125
7	83.3788	19.1492	46.7098	49.9825	2.9149	5.5752

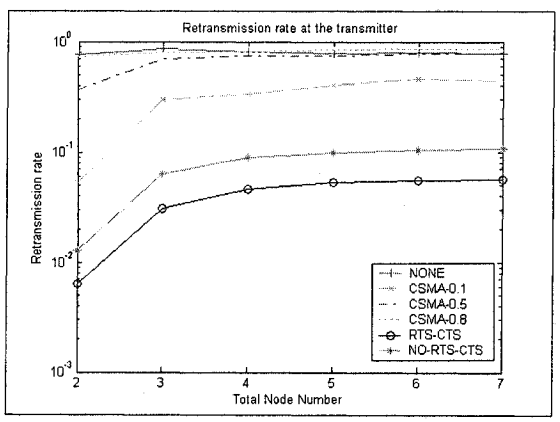


Fig. 6.5 Retransmission rate at the transmitter

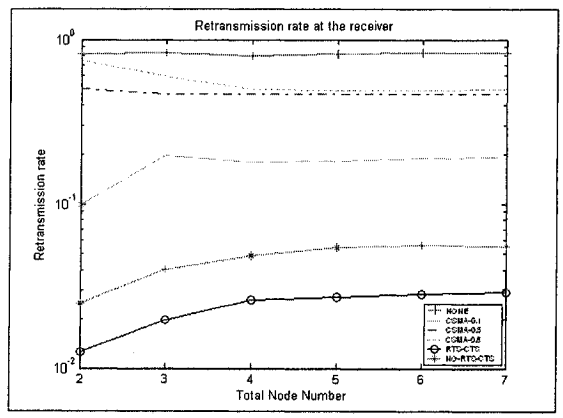


Fig. 6.6 Retransmission rate at the receiver

the CSMA has higher throughput, lower retransmission and collision rates (Table 6.8 - Table 6.6). The selection of the transmission probability value in CSMA mode can have great impact on the performance as well. Small transmission probability can reduce efficiency as well as collision rates. There is a tradeoff between collision and efficiency.

- In the NONE mode, all the nodes will sense the channel to be free at the same time if the propagation delay is ignored. In wireless networks the propagation delay can be ignored because of the relatively small radio range compared with the high microwave speed which equals to light speed. Therefore, all the nodes that have packets to send tend to ship the traffic onto the channel simultaneously, which may cause collision problems with high probability, especially under heavy network load conditions. In the CSMA mechanism, which realized this drawback, instead of sending the packet out directly once the channel is free, the mobile station will make the decision based on the transmission probability that is given (Section 5.3.2). Higher probability leads to more collisions. With low transmission probability the collision can definitely be reduced sharply because only a few nodes will send the packet at the same time. If the transmission probability is chosen to be too small, it's likely that the channel may keep idle even when some nodes have packet to send.

- The CSMA mode also introduces a mechanism which forces the node to be silent (stop sensing the channel) for a period of the maximum propagation delay after the failure of accessing the free medium.

CSMA and DCF-802-11-NO-RTS mode comparison

DCF-802-11-NO-RTS is the mode of 802.11 DCF without the RTS-CTS control mechansim. The DCF-802-11-NO-RTS can perform much better than the CSMA with very low transmission probability.

- For the DCF-802-11-NO-RTS mode, the back-off mechanism is introduced to prevent the nodes from transmitting packets at the same time. The back-off window size can be changed dynamically based on the traffic load and the channel condition.

- In DCF-802-11-NO-RTS mode, DIFS time period deferral is the counterpart of the maximum propagation delay waiting time in the CSMA mode.

- DCF-802-11-NO-RTS mode enables the mobile node to keep silent for EIFS time period when it realizes a corrupted packet has been received. It provides sufficient time for the node to resynchronize with the network.

DCF-802-11-RTS and DCF-802-11-NO-RTS mode comparison

DCF-802-11-RTS is the mode of 802.11 DCF with the RTS-CTS control mechansim.

- The collision and retransmission rate for NO-RTS mode is almost twice as that of the RTS mode. As we know, each independent DATA-ACK transaction in RTS mode is always associated with a RTS-CTS frame pair. It results in the total amount of the transmitted frames in RTS mode is as twice as those in the NO-RTS mode. Taking this into consideration, the performances of these two modes are almost equivalent.

- The introduction of the RTS-CTS mechanism can improve the wireless network performance, especially under heavy network load conditions. The bandwidth of the wireless channel is considerably narrow compared with wired networks, so the collision overhead can greatly impact on the network performance. There is no collision detection mechanism in wireless networks, so when collision happens, the bandwidth wasting for transmitting a RTS frame is obviously less than that of a DATA frame.

- In the network with the traffic load level generated in this scenario, the CSMA and the back-off mechanism provided by 802.11 DCF are sufficient to avoid most of the collisions. Therefore, the overhead induced by the RTS-CTS transaction outweighs the gain in throughput due to the low frequency of collisions. This can explain that the throughput in the RTS mode is slightly lower than that of the NON-RTS mode shown in Figure 6.1. But under more intensive load system with high collision rate, the RTS mode can definitely show its advantage on reducing the bandwidth wasting and improve the network efficiency.

6.1.2 Hidden problem scenario

The *hidden problem* in wireless network can be solved by the RTS-CTS mechanism. A typical hidden problem can be illustrated as Figure 6.7.

Without the RTS-CTS control, the wireless channel is indicated as free to node C even if node A is transmitting because node A is out of the *carrier sense range* (CS range) of

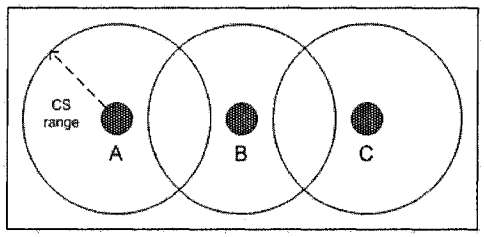


Fig. 6.7 Typical hidden problem scenario

node C. Therefore, node C may start transmitting while node A is also transmitting and a collision could happen. In this case, node A and node C are hidden node from each other.

If there is a RTS-CTS frame transaction prior to the data frame transmission, the data frame collision can be avoided. Node B is in the carrier sense area of both node A and node C. Therefore, the data frame transmission between node A and node B will be known by node C after it receives the CTS frame from node B. Based on the duration field in the CTS frame, node C will not interrupt the current frame transmission by setting NAV process (as explained in Section 5.3.2).

(a)Scenario description

The configuration for this scenario is described in Appendix B.1.2 satisfies the following conditions:

- The communication range for each node is limited to a fixed circle with a radius of $265m$ and the carrier sense radio range for each node is limited to a fixed circle with a different radius of $375m$.
- It's a typical hidden problem scenario. Nodes in a group will definitely not know the existence of other nodes in different groups. So RTS-CTS will take the most vital role here to avoid the collisions caused by *free-faking channel.*

(b)Report analysis

- In this scenario, group 1 and 2 are outside the carrier sense range of each other. As the unique receiver, node four, which is in communication circles of both groups can take the responsibility for reminding the presence of each group to the other by RTS-CTS mechanism. The CTS sent by the receiver can provide the time information to

all nodes other than the transmitter in the network and forces them to set the NAV process in order not to interrupt the current transmission.

- Without the control of the RTS-CTS mechanism, the traffic flow in the network will be totally random. The experimental results show that one group occupy most bandwidth of the channel all the time. This is because the failure of accessing the medium can lead to the increase of the contention window size. After several consecutive failures, the contention window size will stay with a big value, which can make the node with higher possibility of failure in another contention turn than the one with small contention window size. Eventually, this vicious circle can result in the low efficiency of the node.

- The nodes belonging to the same group can share the channel fairly because they can sense each other.

Table 6.7 Hidden problem scenario with RTS mechanism

Scenario Number	Group Number	Throughput (kB/s)	Retransmission Rate (%)
1	1	149.9	17.1315
	2	145.6	17.1270
2	1	139.6	28.4567
	2	142.6	28.4659
3	1	133.1	29.2422
	2	138.2	28.6033

The traffic is very intensive in the above experiments. We can use the *ping* command to generate some lighter traffic. The size of the *Internet Control Message Protocol* (ICMP) packet sent by this operation can be controlled by the command parameters. The increment of the size may elongate the vulnerable time of the packet, which can raise the collision rate. With the increasing of the size, the packet loss rate caused by collision inflates without the RTS-CTS control while there is almost no packet loss with the RTS-CTS control at this traffic load level.

Table 6.8 Hidden problem scenario without RTS mechanism

Scenario Number	Group Number	Throughput (kB/s)	Retransmission Rate (%)
1	1	304.7	3.27870
	2	2.6	80.8749
2	1	0.2	82.2222
	2	304.2	6.25900
3	1	0.1	85.2174
	2	303.9	8.44350

6.1.3 Capture effect scenario

(a)Scenario description

A phenomenon, in which only the stronger one of two received signals will be demodulated, is called *capture effect.* For a specified wireless NIC, the CP_Threshold parameter is defined as the minimum difference between two signals in dB scale that can lead to a successful capture effect. The configuration for this scenario is described in Appendix B.1.3 satisfies the following conditions:

- The CP_Threshold is $10dB$, so the distance configuration in this scenario satisfies the capture effect condition.

(b)Report analysis

- Node 3 can receive the packet from node 1 and node 2. But the received packet power from node 1 and node 2 could not satisfy the requirement of the capture because of the small power difference between the colliding frames. So, the collected data of node 3 can be totally ignored for the following discussion about the impact of the capture to the whole system.
- The capture effect does not affect the *TX COLLISION RATE*, because all the receiving packets will definitely be ruined while the node is transmitting.
- Table 6.9 shows that the *RECV COLLISION RATE* at node 1 was cut in half when the capture effect option is changed from *off* to *on* state. This provides the evidence

Table 6.9 Capture effect scenario

Node ID	Capture Option	Receive Collision Rate (%)	Retransmission Rate (%)
1	ON	0.4876	4.2656
	OFF	0.8694	6.6062
2	ON	2.9892	3.8483
	OFF	3.0663	3.9405
3	ON	0.8727	5.9596
	OFF	0.9432	6.3519

that the network performance benefits from the the capture effect. But for node 2, only a slight decrease is observed for the RECV COLLISION RATE. This is because the node 2 as a file data frame receiver receives much more frames than node 1 which is a transmitter. The receiving collision rate of node 2 is so low under the control of 802.11 DCF that the number of the captured frames is very small. It makes the captured frames occupy only a small proportion of the total received frames, leading to almost no vibration in the collision rate.

- From the analysis above, it can be predicted that capture option can improve the efficiency of networks and reduce the RECV COLLISION RATE.

6.2 Validation of the physical layer and the channel

For this part, a utilization tool package of wireless GINI is used. It is a separated unit from the main body of wireless GINI. It can facilitate the users to avoid complicated computation while doing some configurations of the networks. The source code for the testing section in this tool just duplicates the corresponding part from the wireless GINI. It makes the independent calculation and specific module testing more convenient.

6.2.1 The power gain of different propagation models

The programs in the propagation part of the utilization tool can be used as a testing tool for the power gain comparison of different propagation models. Given the required input

parameters, the values of the distance-power pair can be traced. The configuration for this scenario is described in Appendix B.2.1.

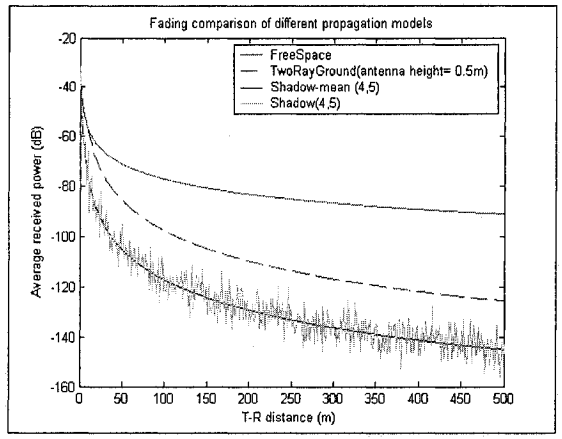


Fig. 6.8 Fading comparison of different propagation models

With the increasing of the T-R distance, the signal power keeps going down continuously. And the reason for the performances of these three models is explained in Section 5.1.2.

6.2.2 Total throughput of different propagation models

The configuration for this scenario is described in B.2.2.

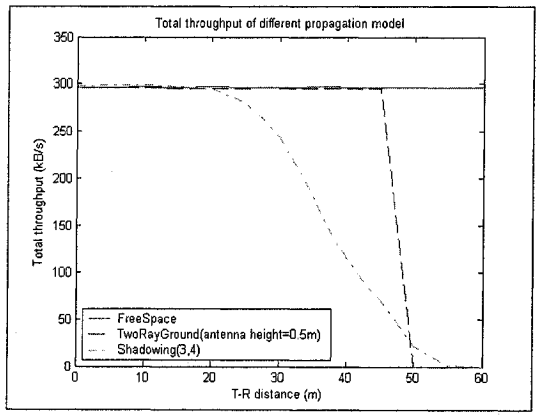


Fig. 6.9 Total throughput of different propagation models

This scenario shows that the two-ray ground reflection model assumes that the distance is the only factor that can affect the reception power of the signal. This makes the communication area an ideal circle. On the other side, some parameters, other than distance, are taken into consideration in the shadowing model so that no power step appears at a specific distance.

6.2.3 Frame error probability in AWGN channel

Mathematical bound emulated by the Error module of GINI

For this experiment, the utilization tool package is used and the configuration for this scenario is described in Appendix B.2.3.

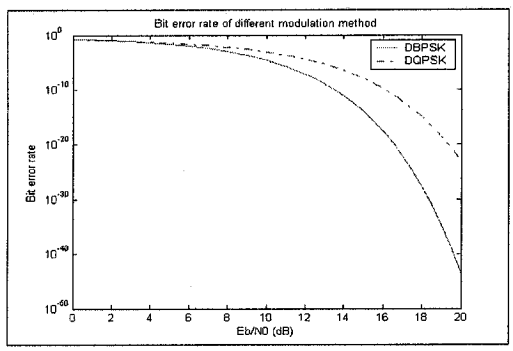


Fig. 6.10 Bit error rate of different modulation method

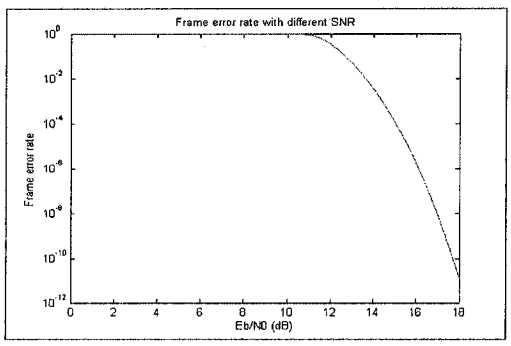


Fig. 6.11 Frame error rate with different SNR *(frame length = 1534 bytes)*

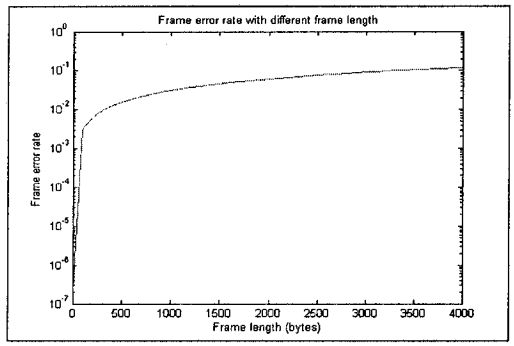


Fig. 6.12 Frame error rate with different frame length *(SNR = 13dB)*

The above figures show the relationship between the SNR per bit, the frame length and the frame error probability. Generally, the higher SNR and the smaller frame length results in lower frame error probability. The reason is described in the Section 5.1.2.

The frame error rate in AWGN channel

The configuration for this scenario is described in Appendix B.2.3.

Table 6.10 Frame error rate in AWGN channel with different SNR

SNR (dB)	Frame Error Rate (%)	Tx Retransmission Rate (%)	Throughput (kB/s)
11	81.7637	80.7777	2.700
12	25.3887	34.9272	229.4
13	3.13718	5.93140	288.3
14	0.22618	1.59970	296.4
15	0.01394	1.30910	296.4
16	0.00000	1.28310	296.9

The above results in Table 6.10 shows that the higher signal-to-noise ratio per bit leads to lower frame error probability.

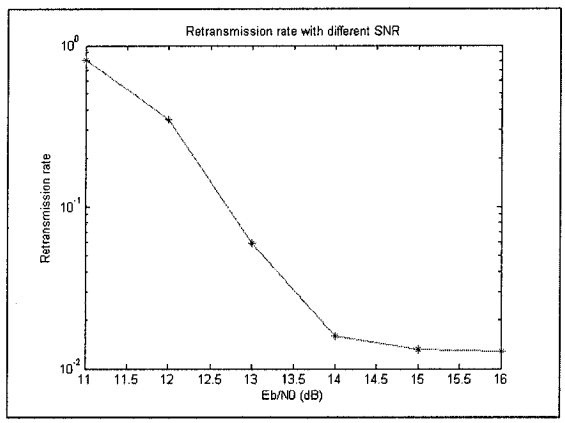


Fig. 6.13 Retransmission rate with different SNR

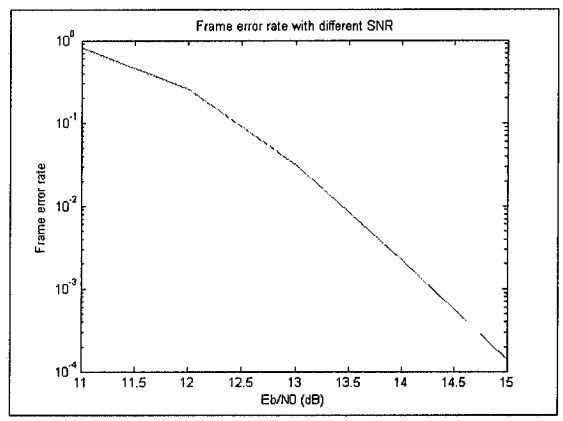


Fig. 6.14 Frame error rate with different SNR

6.2.4 Frame error rate in different channels

The configuration for this scenario is described in Appendix B.2.4. From the experimental results shown in Figure 6.15, we can see that the frame error rate in the Rayleigh fading channel can hardly go down while it decreases to a large extent in the AWGN channel with the increment of the SNR. This shows the negative impact on the performance of the networks by the Rayleigh fading.

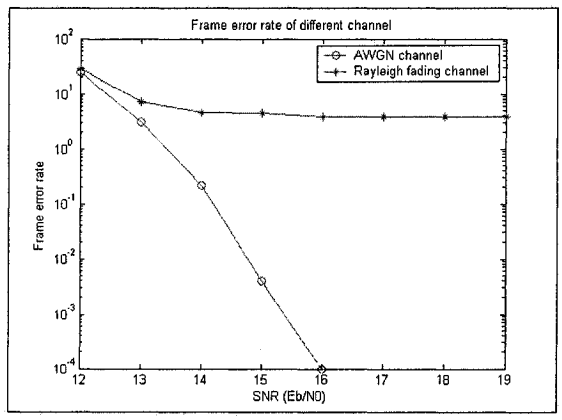


Fig. 6.15 Frame error rate in different channels

Chapter 7

Conclusion and future work

7.1 Conclusion of the wireless GINI implementation

With the wireless GINI system, virtual Ad-hoc WLANs can be created properly. The normal network operations such as ping, ssh, telnet, can be executed via the shell of the virtual terminals (UMLs). Based on the results of the validation experiments in Chapter 6, here is a summary of the features of the wireless GINI:

User-level deployment: The management center of the WLAN running at user-level handles the data communication function for all mobile stations and update the layers and channel states periodically. Thus user-level operation prevents the workstation from crashing due to the misbehavior of the users. It is also easily extensible by adding new modules in wireless GINI.

High performance: The wireless GINI integrates seamlessly with UMLs. The validations in Chapter 6 show that the implementation of the medium access control and the wireless physical layer functions as correct. The virtual WLAN emulated by the wireless GINI can provide highly realistic experiments within a totally software-based environment.

Flexibility of configuration: The configuration parameters of the wireless GINI can be changed easily to emulate various scenarios. Another convenient feature is that all the initial configuration commands can be put into a script and executed automatically at the initialization of the wireless GINI.

Extensibility and programmability: Based on the layered structure of the wireless GINI, a new protocol can be embedded into the system by simply adding a new module and then attaching it to the related layer. It provides a convenient application programming interface for the students.

Evaluation: The traffic of the whole WLAN can be monitored by the statistical module. Any time the system information is needed for analysis or debugging purpose, corresponding reports can be output in a friendly file format (shown in Appendix C).

7.2 Future work

Improve the capacity to handle large scale networks: In the simplest configuration, GINI runs all the virtual devices in a network on the workstation where the student launched the tool. Depending on the network size, this process can significantly load the workstation and increase the response times. To alleviate this problem, as shown in Figure 7.1, GINI allows the distribution of the virtual devices across different remote machines that can include Grid or hosting centers. In the remote mode, the design tool and visualization or monitoring tools run on the client workstations while the virtual devices are hosted on remote machines. If the remote machines are connected by wide-area network, actual network characteristics can be included in the GINI network emulation. Furthermore, GINI emulated networks can interface with actual networks through GINI routers that act as gateways. This requires administrative privileges in one of the workstations or servers hosting GINI elements.

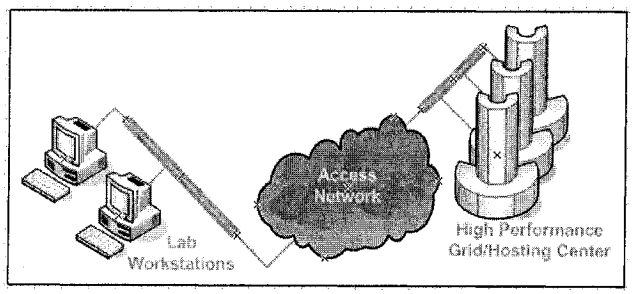


Fig. 7.1 Distributed deployment of a GINI network

Reduce the weight of the current UMLs: Currently, the UML kernel has received a minor amount of changes. In case of network emulations on a single workstation, the UMLs consume the most resources. However, in order to enable the host machine to support as many UMLs as possible, the UML kernel should be light-weighted.

Change the MAC frame encapsulation method: In current wireless GINI, instead of being created by UMLs directly, the IEEE 802.11 frames are obtained by the facilitation of the wireless GINI protocol converter. The efficiency of the system is compromised by the frame transform consuming. Another better solution is to change the source code of file "daemon.c" in the UML package, which is in charge of the encapsulation of the MAC layer frame, making UMLs conform to the 802.11 wireless networks.

Implement the access point functions: The access point functions are needed for the emulation of infrastructure network. The concrete modular structure is illustrated in Figure 7.2. The received frames will be classified by the traffic splitter and then forwarded to the corresponding destinations. The traffic to the 802.x (e.g., 802.3) network will be directed to the wireless GINI router, the ARP frames will be handled by the UML which is acting as an AP, and the traffic addressed to the same WLAN will be stored into the corresponding buffers. Each mobile node in the WLAN is allocated a buffer to store the received frames. The queuing model used by these buffers can be the simplest *Drop Tail* (DT) or the more complicated one called *Random Early Detection* (RED) depending on the different requirements [19]. The polling scheduler is used to select the mobile station to access the medium. The polling algorithm can be self-defined without the concern of the IEEE 802.11 standard. Normally the Round Robin algorithm is used. Better polling system [20] leads to better performance of the network.

Add routing functions for mobile node: In order to make the wireless GINI support the multiple-hop traffic forwarding, the wireless routing algorithm functions should be developed, such as *Ad hoc Multicast Routing protocol utilizing Increasing id-numberS* (AMRIS), *Dynamic Source Routing* (DSR), *Optimized Link State Routing* (OLSR) and so on.

Develop a packet arrival rate controller: For the research purpose, the traffic arrival

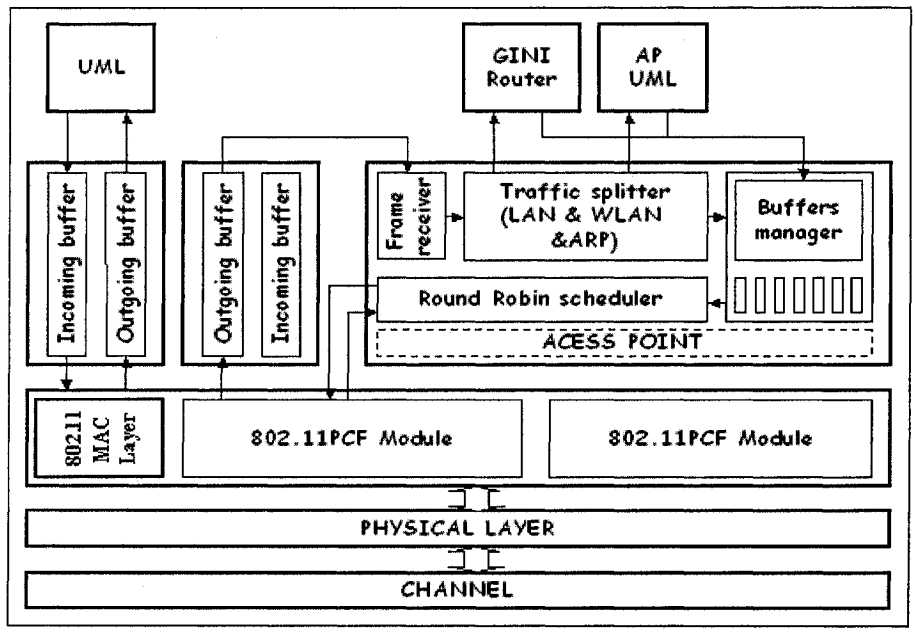


Fig. 7.2 Implementation of the PCF function

rate must be controllable. Therefore, a packet scheduler attached with a time interval generator is necessary. It will only inject the packet into the network based on the time interval sequence, which is created by the time interval generator and follows some specified distribution.

Enable handling the frame by individual bits: As mentioned in Chapter.5, the minimum object that can be handled by wireless GINI is frame. In order to handle the frame error checking procedure at the bit level, a decision rule at the receiver side is needed to recover the signal from the noise. As a simple example, the frame bits are mixed with the noise number sequence generated by the AWGN generator at the receiver side. For each received bit, it will be set to "1" only if its power is above the threshold defined by the decision rule, otherwise "0". This way the frame can be depicted as erroneous or not by the checksum calculation instead of the error-marking scheme implemented in the current wireless GINI.

Appendix A

Command List and Description

The wireless GINI is composed of the wireless GINI center and the facilitation tool. As the core part of wireless GINI, the wireless GINI center is used to create a virtual WLAN in network emulation. The facilitation tool helps users calculate some relevant parameters' values for the wireless GINI center based on the experimental scenario requirements so that the configuration procedure can be simplified.

A.1 Commands for the wireless GINI center

A detailed documentation of wireless GINI is available at:

$$http://www.cs.mcgill.ca/\sim anrl/projects/gini$$

A newer version can be downloaded once the documentation is updated. This documentation gives us a very concrete picture of wireless GINI center. It contains the description of all available commands and the configurations of relevant parameters.

A.2 Commands for the facilitation tool

The code files of the facilitation tool are located in the directory called "util" and the parameters' meanings for different programs are listed as below:

Free Space Test :

- "-p" is followed by the transmission power of the sender.
- "-f" is followed by the frequency of the signal divided by 10^6.
- "-s" is followed by the space between two samples. For instance, if we need to calculate the receiving power at the distance of 20, 40 and 60 meters away from the sender, this value should be set to be 20.
- "-n" is followed by the total number of the samples.
- "-d" is an optional selection and followed by nothing. The result may be shown in dB format with this option on, otherwise $Watt$ it is.

Two-ray Ground Reflection Test :

- "-p", "-f", "-s", "-n" and "-d" have the same meanings as those in free space testing configuration.
- "-t" is followed by the antenna height of the transmitter.
- "-r" is followed by the antenna height of the receiver.

Shadowing Test :

- "-p", "-f", "-s", "-n" and "-d" have the same meanings as those in free space testing configuration.
- "-r" is followed by the reference distance.
- "-b" is followed by path loss exponential parameter.
- "-v" is followed by shadowing deviation.

Awgn channel Test :

- "-b" is followed by the channel bandwidth.
- "-l" is followed by the frame length if the test mode is 2.
- "-r" is followed by the SNR if the test mode is 2.
- "-p" is followed by the value of the start point for the variable in the test.
- "-s" is followed by the sample space.
- "-n" is followed by the number of the samples.

- "-m" is followed by the test mode. Mode 1 is the test with fixed frame length but dynamically changed SNR while mode 2 is that with fixed SNR but dynamically changed frame length.

Appendix B

Configuration of Validation Experiments Scenarios

B.1 Scenarios for MAC layer validation

B.1.1 Comparison of different medium access control mechanism

(a)Configuration procedures

Fixed part :

- Put seven nodes into the network.
- Set all nodes to be stationary with locations of (0,0,0), (20,0,0), (40,0,0), (60,0,0), (80,0,0), (100,0,0), (120,0,0) respectively.
- Set the propagation model to "Free Space".
- Set "Rayleigh Fading" to "OFF".
- Set "AWGN" to be "OFF".
- Set "Capture Option" of each node to "OFF".

Unfixed part :

- Set MAC type.
  - set it to be "NONE"
  - set it to be "CSMA" with transmission probability of 0.1, 0.5 and 0.8.

  - set it to be "802_11_DCF"
- Set RTS_Threshold value (if MAC type is "802_11_DCF").
  - make it to be 0/10000.

Actions :

- Increasing the number of the nodes that transfer the file to node 4 one by one.

(b)Executable commands corresponding to the configuration

1. mov set node 1 switch off (e.g. node 1).
2. mov set node 1 location 0 0 0 (e.g. node 1).
3. ch set prop mode F.
4. ch set fad switch off.
5. ch set awgn switch off.
6. wcard set node 1 cap off (e.g. node 1).
7. mac set node 1 mode N/C/D11.
8. mac set node 1 txprob 0.1/0.5/0.8 (e.g. node 1).
9. mac set node 1 rtsthres 0/10000.

B.1.2 Hidden problem scenario

(a)Configuration procedures

Fixed part :

- Put seven nodes into the network.
- Set all nodes to be stationary with locations of (0,0,0), (20,0,0), (40,0,0), (240,0,0), (440,0,0), (460, 0, 0) and (480, 0, 0) respectively.
- Use the channel mode of "Free Space".
- Set "Rayleigh Fading" to be "OFF".

- Set "AWGN" to be "OFF".
- Set "Capture Option" to be "OFF".
- Set MAC layer type to be "802_11_DCF".
- The Carrier Sense Threshold Power is half as much as RX Threshold.

Unfixed part :

- Set RTS_Threshold to be 0/10000.

Actions :

- Divide the nodes into two groups. Node 1, 2 and 3 are in Group 1 while 5, 6, 7 belong to Group 2.
- Establish three scenarios, which are transferring a file from node 3 and 5 to node 4, transferring a file from node 2,3 and 5,6 to node 4 and transferring a file from node 1,2,3 and 5,6,7 to node 4 simultaneously. Assign a number for each of these scenarios from 1 to 3.

(b)Executable commands corresponding to the configuration

1. mov set node 1 switch off (e.g. node 1).
2. mov set node 1 location 0 0 0 (e.g. node 1).
3. ch set prop mode F.
4. ch set fad switch off.
5. ch set awgn switch off.
6. wcard set node 1 cap off (e.g. node 1).
7. mac set node 1 mode D11.
8. mac set node 1 rtsthres 0/10000.

B.1.3 Capture effect scenario

(a)Configuration procedures

Fixed part :

- Put three nodes into the network.
- Set all nodes to be stationary with locations of (0,0,0), (15, 0, 0) and (280, 0, 0) respectively.
- Set the Capture Threshold to $10dB$.
- Use the channel mode of "Free Space".
- Set "Rayleigh Fading" to "OFF".
- Set "AWGN" to be "OFF".
- Set MAC layer type to be "802_11_DCF".
- Set RTS_Threshold to 10000.

Unfixed part :

- Set "Capture Option" to "ON/OFF".

Actions :

- Transfer a file from node 1 and node 3 to node 2 simultaneously.

(b)Executable commands corresponding to the configuration

1. mov set node 1 switch off (e.g. node 1).
2. mov set node 1 location 0 0 0 (e.g. node 1).
3. ch set prop mode F.
4. ch set fad switch off.
5. ch set awgn switch off.
6. mac set node 1 mode D11.

7. mac set node 1 rtsthres 10000.

8. wcard set node 1 cap on/off (e.g. node 1).

B.2 Scenarios for physical layer and channel validation

B.2.1 The power gain for different propagation models

Free Space Model :

- Compile all files in the directory of "/util/propagation/freespace/" by: "*gcc -Wall -o fs *.c -lm -lpthread -g*".
- Run the executable file by the command of: "*./fs -p 0.2818 -f 900 -s 5 -n 100 -d*".

Two-Ray Ground Reflection Model :

- Compile all files in the directory of "/util/propagation/tworayground/" by: "*gcc -Wall -o tr *.c -lm -lpthread -g*".
- Run the executable file by the command of: "*./tr -t 0.5 -r 0.5 -p 0.2818 -f 900 -s 5 -n 100 -d*".

Shadowing Model :

- Compile all files in the directory of "/util/propagation/shadow/" by: "*gcc -Wall -o sh *.c -lm -lpthread -g*".
- Run the executable file by the command of: "*./sh -r 1 -p 0.2818 -f 900 -s 5 -n 100 -b 3 -v 4 -d*".

B.2.2 Total throughput of different propagation model

Configuration procedures

Fixed part :

- Put two nodes into the network.
- Set all nodes to be stationary with locations of (0,0,0) and (5, 0, 0)respectively.

- Set "Rayleigh Fading" to be OFF.
- Set "AWGN" to be OFF.
- Set MAC layer type to be "802_11_DCF".
- Set RTS_Threshold to be 100000.
- Set "Capture Option" to be OFF.

Unfixed part :

- Set the propagation mode to be "Free Space", "Two-Ray Ground Reflection" and "Shadowing".
- Change the T-R distance.

Actions :

- Transfer a file from node 1 to node 2.

Executable commands corresponding to the configuration

1. mov set node 1 switch off (e.g. node 1).
2. mov set node 1 location 0 0 0.
3. ch set fad switch off.
4. ch set awgn switch off.
5. mac set node 1 mode D11.
6. mac set node 1 rtsthres 10000.
7. wcard set node 1 cap off (e.g. node 1).
8. ch set prop mode F/T/S.
9. mov set node 2 location x 0 0 ($x = 10, 15, 20...$).

B.2.3 Frame error probability in AWGN channel

(a)Mathematical bound emulated by the Error module of GINI

Fixed frame length :

- Compile all files in the directory of "/util/awgn/" by:
  *"gcc -Wall -o awgn *.c -lm -lpthread -g"*.
- Run the executable file by the command of:
  "./awgn -b 2 -l 1534 -p 0 -s 0.5 -n 40 -m 1".

Fixed SNR per bit :

- Compile all files in the directory of "/util/awgn/" by:
  *"gcc -Wall -o awgn *.c -lm -lpthread -g"*.
- Run the executable file by the command of:
  "./awgn -b 2 -r 13 -p 100 -s 100 -n 40 -m 2".

(b)The frame demodulation errors rate in AWGN channel

Configuration procedures

Fixed part :

- Put two nodes into the same communication area of the network and set them to be stationary.
- Set "Capture Option" to be OFF.
- Set MAC layer type to be "802_11_DCF".
- Set RTS_Threshold to be 100000.
- Set "AWGN" to "ON" and set the AWGN to be "snr" mode.
- Set "Rayleigh Fading" to be OFF.
- Set the propagation mode to "Free Space".

Unfixed part :

- Change the SNR of the AWGN channel.

Actions :

- Transfer a file from node 1 to node 2.

Executable commands corresponding to the configuration

1. mov set node 1 switch off (e.g. node 1).
2. mov set node 1 location 0 0 0.
3. mov set node 2 location 50 0 0.
4. wcard set node 1 cap off (e.g. node 1).
5. mac set node 1 mode D11.
6. mac set node 1 rtsthres 10000.
7. ch set fad switch off.
8. ch set prop mode F.
9. ch set awgn switch on.
10. ch set awgn mode snr.
11. ch set awgn para ratio x ($x = 11, 12, 13...$).

B.2.4 Packet drop rate for different channel

Configuration procedures

Fixed part :

- Put two nodes into the network.
- Set the map size to be (60 60 40).
- Set node 1 to be stationary with the location of (0,0,0).
- Set MAC layer type to be "802_11_DCF".
- Set RTS_Threshold to be 10000.

- Set "Capture Option" to "OFF".
- Set the propagation mode to be "Free Space".
- Set "AWGN" to "ON" and set the AWGN to be "snr" mode.
- Set the mobility speed of node 2 to be constant of $10m/s$.

Unfixed part :

- Change the SNR of the AWGN channel.
- Set "Rayleigh Fading" to "ON/OFF".

Actions :

- Transfer a file from node 1 to node 2.

Executable commands corresponding to the configuration

1. sys set map size 60 60 40.
2. mov set node 1 switch off.
3. mov set node 1 location 0 0 0.
4. mov set node 2 spd max 10.
5. mov set node 2 spd min 10.
6. wcard set node 1 cap off (e.g. node 1).
7. mac set node 1 mode D11.
8. mac set node 1 rtsthres 10000.
9. ch set prop mode F.
10. ch set awgn switch on.
11. ch set awgn mode snr.
12. ch set awgn para ratio x ($x = 11, 12, 13...$).
13. ch set fad switch on/off.

Appendix C

Examples of Different Format Reports

C.1 Sample of packet statistical report

```
==========================================================
SENDING PACKETS INFORMATION OF NODE 2.
==========================================================
GENERAL SENDING PACKETS INFORMATION.
==========================================================
The number of the BROADCAST packets have been sent is: 0
The number of the UNICAST packets have been sent is: 72924
==========================================================
CLASSIFIED SENDING PACKETS INFORMATION.
==========================================================
The number of the RTS packets have been sent is: 75652
The number of the CTS packets have been sent is: 142576
The number of the DATA packets have been sent is: 72924
The number of the ACK packets have been sent is: 142576
==========================================================
The TOTAL number of the packets have been sent is: 433728
==========================================================
==========================================================
SENDING PACKETS RETRY INFORMATION OF NODE 2.
```

==========

The DATA PACKET RETRY TIMES number is: 0

The RTS PACKET RETRY TIMES number is: 2728

==========

==========

SENDING PACKETS INFORMATION OF NODE 2.

==========

The number of the packet DROPED FOR EXCEEDING THE MAX RTS RETRY VALUE is: 0

The number of the packet DROPED FOR EXCEEDING THE MAX DATA RETRY VALUE is: 0

==========

The TOTAL number of the packet DROPED FOR FAILURE OF RETRANSMISSION is: 0

==========

==========

DROPING INFORMATION OF THE PACKETS RECEIVED BY NODE 2.

==========

The number of the received packet DROPED FOR TRANSMITTING AT THE SAME TIME is: 2685

The number of the received packet DROPED FOR COLLISION is: 5410

The number of the received RTS packet DROPED FOR COLLISION is: 5410

The number of the received CTS packet DROPED FOR COLLISION is: 0

The number of the received ACK packet DROPED FOR COLLISION is: 0

The number of the received DATA packet DROPED FOR COLLISION is: 0

The number of the received packet DROPED FOR LOW RECV POWER is: 0

==========

The TOTAL number of the received packet DROPED by ERROR is: 8095

==========

==========

GENERAL INFORMATION OF THE PACKETS RECEIVED BY NODE 2.

==========

The TOTAL number of the packet received is: 439096

The number of the packet received CORRECTLY is: 431001

The number of the packet received NOT FOR ITSELF is: 0

==========

CLASSIFIED INFORMATION OF THE PACKETS RECEIVED BY NODE 2.

==========

The number of the RTS packet received is: 142576
The number of the CTS packet received is: 72924
The number of the DATA packet received is: 142577
The number of the ACK packet received is: 72924

==========

==========

INFORMATION OF CAPTURE OF NODE 2.

==========

The number of the packet CAPTURED is: 0

==========

==========

STATISTIC INFORMATION OF NODE 2.

==========

The RTS PACKET RETRANSMISSION RATE is: 0.036060
The DATA PACKET RETRANSMISSION RATE is: 0.000000
The TOTAL PACKET RETRANSMISSION RATE is: 0.018361
The RECV ERROR DROP RATE FOR RECV COLLISION is: 0.012321
The RECV ERROR DROP RATE FOR RECV WHILE TX is: 0.006115
The RECV ERROR DROP RATE FOR LOW POWER is: 0.000000
The RECV TOTAL ERROR DROP RATE is: 0.018436
The RECV PKT DROP RATE is: 0.018436
The TOTAL COLLISION RATE is: 0.018436

==========

C.2 Sample of node mobility report

==========

LOCATION COORDINATOR OF NODE 1.

==========

The coordinate is: 286.520578 407.181437 319.512658
The coordinate is: 288.705297 409.057496 323.978003
The coordinate is: 290.890015 410.933555 328.443348

The coordinate is: 293.074734 412.809614 332.908693
The coordinate is: 295.259452 414.685673 337.374038
The coordinate is: 297.444171 416.561732 341.839383
The coordinate is: 299.628889 418.437791 346.304728

C.3 Sample of wireless channel fading report

==========

Channel Fading Factor was updated ...

==========

Node 1: 0.000000 -1.551845 3.190267 -1.318436 -10.108854 4.469874 1.218008
Node 2: 3.651160 0.000000 -4.533440 2.225721 -12.342975 3.141941 2.551107
Node 3: -4.989391 -3.492962 0.000000 -2.346363 -6.275330 -5.298275 -4.884673

==========

Channel Fading Factor was updated ...

==========

Node 1: 0.000000 -7.480700 -4.214451 4.507660 2.916981 2.289612 -1.786955
Node 2: -1.937427 0.000000 3.000990 1.330337 3.454486 -1.172887 0.812446
Node 3: -0.225566 -12.392354 0.000000 0.293990 5.356779 -1.902211 -11.039767

==========

Channel Fading Factor was updated ...

==========

Node 1: 0.000000 -7.655197 -1.412751 -4.386268 0.755487 -3.430891 -6.284442
Node 2: -6.139914 0.000000 -0.029139 1.308025 -2.007705 4.333620 -1.205514
Node 3: 3.122818 3.672300 0.000000 -5.722578 -12.486868 -2.734982 -0.243340

Appendix D

Abbreviations and acronyms

GINI	software developed by ANRL lab in Mcgill university
API	application programming interfaces
UML	user-mode linux
UVS	UML virtual switches
WLAN	wireless local area network
CSMA-CA	carrier sense multiple access protocol with collision avoidance
CSMA-CD	carrier sense multiple access with collision detection
LLC	logic link control
MAC	medium access control
PHY	physical layer
AP	access point
FHSS	frequency-hopping spread spectrum
DSSS	direct sequence spread spectrum
BSS	basic service set
IBSS	independent basic service set
DS	distribution system
ESS	extended service set
LOS	line-of-sight
ISM	industrial, scientific, and medical
T-R	transmitter-receiver
DBPSK	differential binary phase shift keying
DQPSK	differential quadrature phase shift keying
DCF	distributed coordination function
PCF	point coordination function
RTS	request to send frame
CTS	clear to send frame
IFS	inter-frame space

CP	contention period
CFP	contention-free period
VPL	virtual physical layer interface
ARP	address resolution protocol
NAV	network allocation vector
AWGN	additive white gaussian noise
BER	bit error rate
MPDU	MAC protocol data unit
PLCP	physical layer convergence protocol
PMD	physical medium dependent sublayer
PPDU	PLCP protocol data unit
PSM	power saving mode
DIFS	DCF interframe space
EIFS	extended interframe space
LRC	long retry count
SRC	short retry count
MSDU	MAC service data unit
MMPDU	MAC management protocol data unit
CW	contention window size
SAP	service access point

References

[1] *ANSI/IEEE Std 802.11, 1999 Edition (R2003), Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, June 2003. IEEE-SA Standards Board.

[2] W. Xu, "The GINI router: A Routing Element for User-level Micro Internet," Master's thesis, School of Computer Science, Mcgill University, Montreal, Canada, Jun. 2004.

[3] K. Fall and K. Varadhan, *The ns Manual(formerly ns Notes and Documentation).* The VINT Project, Dec. 2003.

[4] W. peng Chen, Y. Ge, C. Hu, H. Zhang, and R. Zheng, "*J-Sim Wireless Extension Tutorial*," Feb. 2004.

[5] G. Mazzini, *Simple Ad hoc siMulator (SAM) User's Manual.* University of Ferrara.

[6] N. Burri, R. Wattenhofer, Y. Weber, and A. Zollinger, "SANS: A Simple Ad hoc Network Simulator," in *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA)*, (Montreal, Quebec, Canada), 27th Jun - 2nd Jul. 2005.

[7] V. Guffens, G. Bastin, and O. Bonaventure, "An emulation infrastructure for multi-hop wireless communication networks." Université Catholique de Louvain, March 31, 2005.

[8] S. H. Michael Engel, Matthew Smith and B. Freisleben, "Wireless Ad-hoc Network Emulation Using Microkernel-based Virtual Linux Systems." Dept. of Mathematics and Computer Science, University of Marburg.

[9] M. Hohmuth, "The fiasco kernel: Requirements definition," tech. rep., December. 1998.

[10] B. P. Crow, F. Indra Widjaja, J. G. Kim, and P. T. Sakai, "Ieee-802.11 wireless local area networks," *IEEE Communications Magazine*, vol. 35, pp. 116–126, September. 1997.

[11] J. Dike, "User-mode linux kernel home page," http://user-mode-linux.sourceforge.net/, 2003.

[12] T. S. Rappaport, *Wireless communications, principles and practice.* Prentice Hall, 1996.

[13] M. Pätzold and F. Laue, "Statistical properties of jakes' fading channel simulator," in *IEEE 48th Veh. Technol. Conf., VTC'98, Ottawa, Ontario, Canada*, May. 1998.

[14] W. C. Jakes, *Microwave mobile communications.* McGraw Hill, 1993. Piscataway, NJ : IEEE Press.

[15] L. Gavrilovska and V. Atanasovski, "Influence of packet length on ieee 802.11b throughput performance in noisy channels." Center for TeleInfrastruktur (CTIF), Aalborg University, Denmark & Faculty of Electrical Engineering, Ss Cyril and Methodius University C Skopje, Macedonia.

[16] J. Proakis, *Digital communication.* McGraw Hill, 4th ed., 2001.

[17] A. Aguiar, "Wireless channel models," tech. rep., TKN Technical Report TKN-03-007, Berlin, Apr. 2003.

[18] K. Jamieson, "Implementation of a power-saving protocol for ad hoc wireless networks," Master's thesis, Massachusetts Institute of Technology, USA, Feb. 2002.

[19] C. Brandauer, G. Iannaccone, C. Diot, T. Ziegler, S. Fdida, and M. May, "Comparison of tail drop and active queue management performance for bulk-data and web-like internet traffic," *Sixth IEEE Symposium on Computers and Communications.*

[20] O. Sharon and E. Altman, "An efficient polling mac for wireless lans," *IEEE/ACM Trans. Networking*, vol. 9, no. 4, Aug. 2001.