A COMPARISON OF WIRELESS NETWORK PROTOCOLS FROM A SIMULATION PERSPECTIVE

by

Robert Bathmann

A Thesis

Presented to the Faculty of

Bucknell University

In Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science with Honors in Computer Science

May 4, 2009

Approved:

Felipe Perrone
Thesis Advisor

Xiannong Meng
Chair, Department of Computer Science

Acknowledgments

Thank you to everyone who helped me complete my Honors Thesis. I could not have done it without all the support I received from many people. I want to especially thank:

- Felipe Perrone, for all the hard work and time he spent working with me on my thesis. He always provided me answers when I had questions and encouragement when I felt discouraged. Although he insisted that he would become my worst enemy, he was always a friend.
- Kristen Brown, for forcing (nagging?) me to work on my thesis when I otherwise would have not.
- My family and friends who supported me while writing my thesis and throughout my education

Contents

List of Figures

Abstract

In the course of recent years, a new and exciting wireless network protocol has emerged. The IEEE 802.15.4 standard provides the specification to build low-power, low-rate wireless network devices. This facilitates the creation of wireless sensor networks (WSNs). WSNs can be used in many different scenarios ranging from farming data collection to home automation and security. As applications that use IEEE 802.15.4 begin to become more prevalent, there is a tremendous research interest in these networks.

In order to study IEEE 802.15.4 wireless networks and the scenarios it makes possible, simulation is very helpful. Simulation eliminates the need to set up a large-scale experiment and provides repeatable results. At Bucknell, we currently have a scalable network simulator, but it is only capable of simulating IEEE 802.11b wireless networks. However, it is possible to develop a IEEE 802.15.4 model for the network simulator. Not much is known about IEEE 802.15.4 in comparison with the widely-used and understood IEEE 802.11 WiFi standard for Wireless Area Networks. This thesis provides a comparison between the two wireless networking standards so that one who knows WiFi well can understand easily how it differs from IEEE 802.15.4. The comparison will also serve to guide the creation of a simulation model for IEEE 802.15.4.

Chapter 1

Introduction

The goal of this thesis is to compare, with an emphasis on simulation, two wireless network protocols: WiFi, which is defined by the IEEE 802.11 specification and Zigbee, which is defined by IEEE 802.15.4. Chapter 1 introduces the concept of wireless networks and explains the different goals of IEEE 802.11 and IEEE 802.15.4 wireless networks. Chapter 2 explains the medium access mechanisms of both standards. Chapter 3 details the differences between IEEE 802.11 and IEEE 802.15.4 with regards to radio characteristics. Chapter 4 describes simulator architecture and the effort to port an existing IEEE 802.15.4 implementation to another simulator. Finally, Chapter 5 concludes the thesis with a road map to implementing Zigbee.

Note that for the purpose of this thesis, Zigbee and the IEEE 802.15.4 specification are synonymous. The Zigbee standard actually refers to routing protocols outside the scope of this thesis, but Zigbee devices rely on the IEEE 802.15.4 specification.

1.1 Introduction to Wireless Networks and Simulation

Wireless networking has increased the level of mobility, convenience, and productivity of computers and technology. It has allowed laptops to connect to the Internet from anywhere in the vicinity of an access point or other computer. Wireless headsets can now communicate with cell phones to provide a safer way to drive a vehicle.

Additionally, a network of low-powered devices can be used to collect data where wires would make it difficult or impossible. These networks had to be studied before they could become mainstream technology.

Directly studying wireless networks is a challenging task for two reasons. First, the conditions surrounding the experiment can never be fully controlled and are therefore not repeatable. Second, it is difficult to manage (and pay for) a large-scale experiment with many human-operated hardware devices. Instead, a model can be built in software to replicate the behavior of a real network. A high-performance computer simulation that uses these models allows for fully controllable experiments with repeatable results. Since networking hardware is not required, simulations can be conducted at low cost and still yield results that can predict or estimate the metrics of the real system (Liu et al. 2001).

Three metrics are commonly used to characterize the performance of wireless networks. First, the packet delivery ratio is used to indicate the success rate of transmitting data. It is the percentage of packets received relative to the number of packets sent. Since packets can be lost in a variety of ways during transmission, this is an important metric. Second, the time delay between sending and receiving a packet represents the network latency. It is desirable to minimize the network latency to improve the response time of network applications. For example, low latency is important for voice chat or online gaming. Last, the amount of overhead traffic needed for network maintenance and routing is used to indicate the efficiency of the network. A network that has too much control traffic will reduce the available bandwidth for data (Perrone and Nelson 2006).

Networks are too complex to be understood all at once. Therefore, a standard reference model was created to separate the different components of networking hardware and software. The International Standards Organization Open Systems Interconnection (ISO OSI) Reference Model specifies seven distinct layers of standard functionality, depicted in Figure 1.1. The scope of this thesis concerns itself with the two lowest layers on which all other layers rely. The physical layer is responsible for transforming information into signals that travel over the network medium. The data link layer is responsible for controlling access to the medium and detecting and correcting errors during transmission (Tanenbaum 2003). Normally, it is only these two bottom layers that distinguish one type of network from another. Upper layers typically remain the same regardless of the network type (Gutierrez, Callaway Jr., and Barrett Jr. 2007).

The main benefit from the ISO OSI layered modeled is abstraction. Abstraction

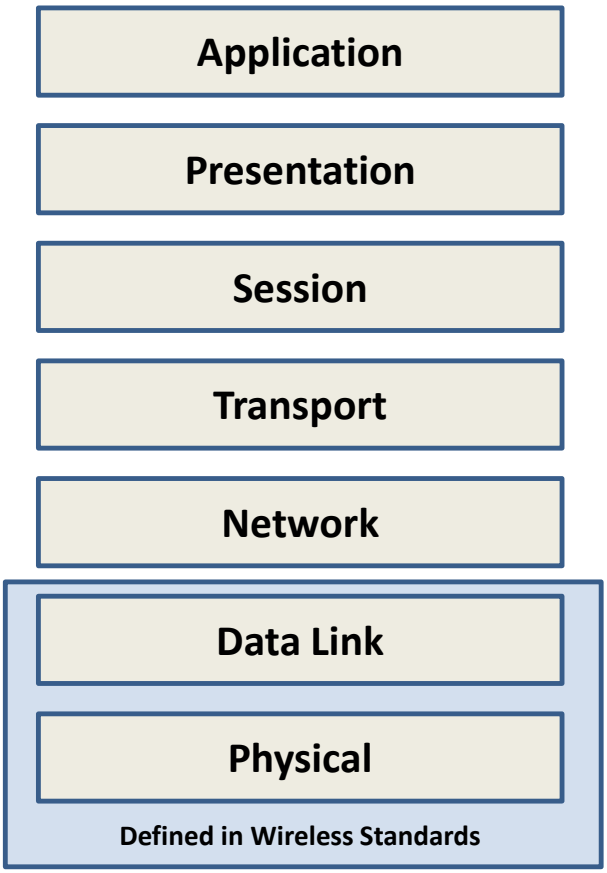

Figure 1.1: ISO/OSI Seven-Layer Networking Stack.

hides the specific details of each layer's behaviors. This results in interoperability and flexibility between layers. Since the interfaces between layers are fixed, a layer can be swapped out with a different implementation and everything will still work. For example, the Medium Access Control (MAC) and Physical (PHY) layers can be implemented as either a wired or wireless protocol. The next layer above (Network) will function with either the wired or wireless implementation.

Wireless networks can be partitioned into two main classifications: Wireless Local Area Networks (WLANs) and Wireless Personal Area Networks (WPANs). WLANs often provide fast access to the Internet and other resources. They offer high-rate data

transfer over an average distance of about 300 ft. On the other hand, WPANs provide networking on short distances of about 30 ft and at lower data rates. Since they are limited in bandwidth and range, their power consumption and cost is minimized (Cooklev 2004).

1.2 WiFi

The IEEE 802.11 standard defines the WiFi specification. WiFi was created to eliminate the main disadvantages of local area networks: wires. Although Ethernet was a success, it required special networking cables to be installed from computer to computer. A wireless implementation of Ethernet would solve this problem. As a result, the IEEE 802.11 WiFi standard was created (Cooklev 2004).

Different physical layers exist for IEEE 802.11 wireless networking. Each physical layer will have unique power requirements, signal range, and bandwidth specifications. For example, the IEEE 802.11b physical layer has a maximum throughput of 11 megabits per second (Mbps) and has a range between 150-1000ft depending on environmental conditions with a frequency of 2.4GHz. The bandwidth was increased to 54 Mbps with IEEE 802.11g, which uses a different physical layer and a modified MAC layer to ensure backwards compatibility with IEEE 802.11b (Cooklev 2004).

There are two main types of network architectures described in the IEEE 802.11 standard. The first is an Independent Basic Service Set (IBSS). Using this structure, network devices communicate ad-hoc, without infrastructure and independent of a central authority. Decision making (e.g., when a device should transmit) for the network is based on distributed algorithms that work across the IBSS. All the devices have the same responsibilities and the network is typically short-lived. Any device can connect and disconnect from an IBSS without major disruption since the network is self-adjusting. Figure 1.2 depicts the structure of an ad-hoc wireless network where the squares represent wireless devices (Cooklev 2004).

The second type of network architecture is a Basic Service Set (BSS). In this model, a single Access Point (AP) device is responsible for decision making. Multiple devices can connect and communicate with the AP, but not with each other. The AP is usually connected to an additional wired network to join two networks together. For instance, an AP connected to a broadband modem is a very common scenario. BSSs are more permanent than an IBSS since the AP must be established and always

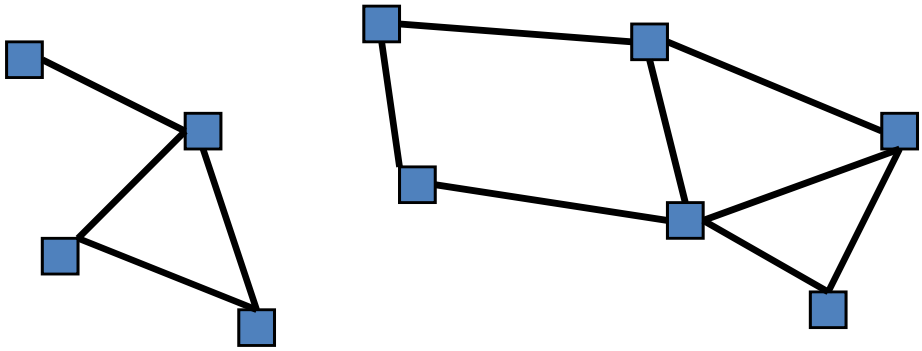

Figure 1.2: IEEE 802.11 Ad-hoc Network.

available, which improves network reliability. Figure 1.3 depicts the structure of several infrastructure wireless networks with a common wired backbone (Cooklev 2004).

Whereas the goal of WiFi was to make Ethernet wireless, WPAN wireless technologies seek to create small, low-power networks for devices to communicate. One of these WPAN standards, Zigbee, is introduced in the next section.

1.3 Zigbee

The IEEE 802.15.4 standard defines the Zigbee specification. This protocol is intended for use with extremely low-powered devices, short transmission distances up to about 30 ft, and low data rates up to about 250 kilobits per second. The primary application of Zigbee is to create Wireless Sensor Networks (WSNs), which are used to decrease the cost of installing sensors, avoid the problems caused with cable connections, and decrease the complexity of the network (Gutierrez, Callaway Jr., and Barrett Jr. 2007).

Since the specific applications of Zigbee and WSNs are probably less familiar to consumers than WiFi, it is helpful to illustrate with examples. A sensor network for home automation could be deployed to monitor temperature readings in different

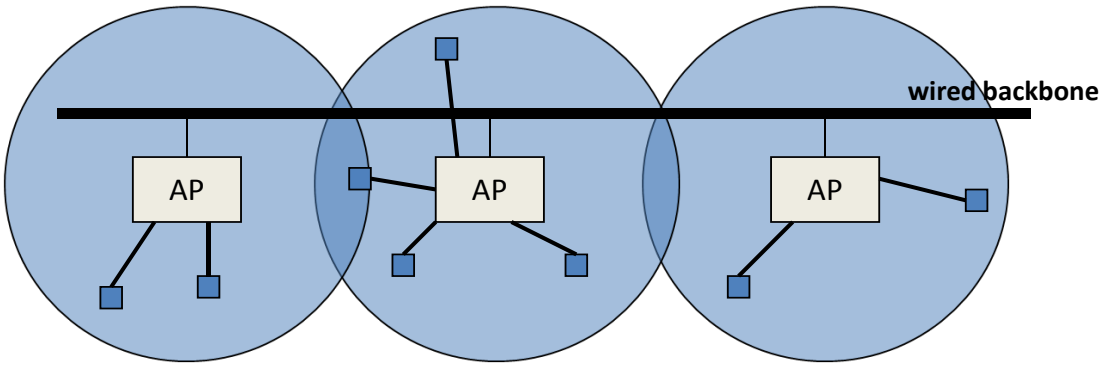

Figure 1.3: IEEE 802.11 Infrastructure Network.

rooms and adjust the thermostat accordingly. Wireless smoke detectors and burglar intrusion sensors could interact with a home security system. A WSN could also be used to increase the efficiency of farming by creating a self-organizing network to take soil readings and other statistics across large fields. Additionally, small wireless gadgets such as remote controls and toys can take advantage of the low cost and power demands of Zigbee (Gutierrez, Callaway Jr., and Barrett Jr. 2007).

The network architecture specified by IEEE 802.15.4 is very different from the IEEE 802.11 architecture. Devices on a Zigbee network can have one of three different roles. The first is the Personal Area Network (PAN) Coordinator. The PAN Coordinator is responsible for the general management of the network. It is in control of the type of network and responsible for allowing other devices to join the network. The second type of role is a coordinator. Coordinators can forward messages to other nodes and act as a proxy to the PAN Coordinator for nodes that are out of range. The last type of role is called a network device. A IEEE 802.15.4 network exists when there is exactly one PAN coordinator and at least one network device (Gutierrez, Callaway Jr., and Barrett Jr. 2007).

While there are three different types of roles a device may serve, there are two different physical device types. The first type of device is a Full Function Device (FFD). FFDs are fully capable of all MAC services and can serve any of the three roles. The second type is a Reduced Function Device (RFD). RFDs are only capable of becoming a network device. They were designed to allow for the creation of extremely low-powered devices that were not required to forward messages or perform any other complex or unnecessary power consuming tasks (Gutierrez, Callaway Jr., and Barrett

Jr. 2007).

Two network topologies are supported by Zigbee. The star network topology is characterized by a central PAN Coordinator that can directly communicate with every other node. Other network devices can only communicate with the PAN Coordinator. The peer-to-peer network topology consists of a single PAN Coordinator and other coordinators or network devices. As long as they are within range, devices may communicate with each other without the direct involvement of the PAN Coordinator. The peer-to-peer network is depicted in Figure 1.4 (IEEE 802.15.4 2006).

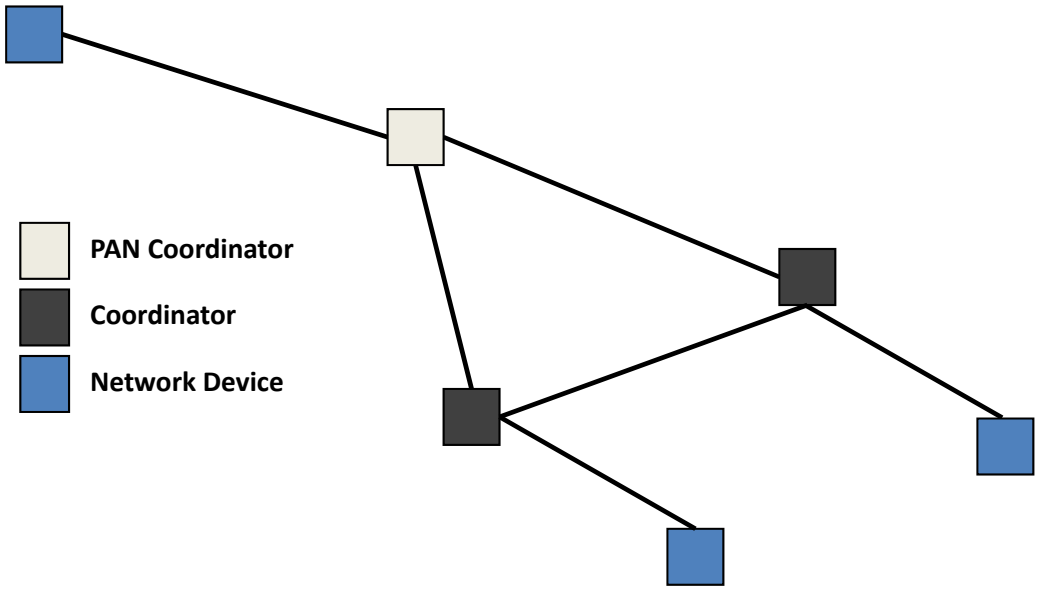

Figure 1.4: IEEE 802.15.4 Peer-to-Peer Network.

1.4 Chapter Summary

In this chapter, the importance of wireless networks and why it is useful to simulate them was discussed. Additionally, the basic network topologies of both WiFi and Zigbee were presented. WiFi has two modes of operation: as a BSS, and an IBSS. BSSs have a central device to manage the network whereas IBSSs form a distributed, ad-hoc network. Zigbee also has two similar network structures: star and peer-to-peer. However, both network formations need a central PAN Coordinator to manage the network.

In the next chapter, the MAC layers of WiFi and Zigbee are examined.

Chapter 2

The MAC Layer

Several issues arise using a broadcast medium such as wireless where all nodes share access to the network simultaneously. Consider a room full of people who always have something to say to everyone else. If everybody started talking at the same time, it would be impossible to listen to what individual people have to say. Therefore, the restriction must be imposed that only one person may talk at a time. But who decides which person should talk at a given time? How is that person chosen? And once the message is sent, did everybody hear the message correctly?

Essentially, this is the problem the Medium Access Control (MAC) layer solves (Tanenbaum 2003). By imposing strict regulations on when a wireless device can send data across the shared medium, the chance that another transmitting device will interfere is reduced.

Each wireless standard has its own MAC layer. While their goals are relatively the same (controlling access to the airwaves), they often achieve them using different means.

2.1 Contention Access

Contention access is a mechanism by which different nodes compete with each other in order to use the shared medium. In the case of the room of people, this will

involve a set of policies that dictate when a person can begin to talk. For instance, a person may talk if no other person is currently talking. Contention access is a form of distributed decision-making.

WiFi

The Distributed Coordination Function (DCF) is responsible for deciding when a wireless device can transmit on the medium when multiple stations are competing to transmit. The basic mechanism for arbitrating channel access in the IEEE 802.11 standard is Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA) and Exponential Backoff. Simply put, this method requires devices that have data to transmit to first listen for an open channel. Unlike wired devices, most wireless devices cannot send and receive at the same time, so collisions cannot be detected by transmitting stations. If no other devices are transmitting during some interval of time before some device wishes to transmit, there is a good chance no other device will start to transmit as well and distort the signal (O'Hara and Petrick 2005).

Timing is essential for CSMA/CA to work. Several different timing intervals are defined by the standard. The Interframe Space (IFS) is the time between frames. Short IFS (SIFS) is fixed by the physical layer and represents the time needed for the device to switch between send and receive mode. The Point IFS (PIFS) is SIFS plus the time to detect an open channel, switch modes, MAC processing, and air propagation. The sum of those is known as the slot time. The Distributed IFS (DIFS) is the SIFS plus two slots times. The relationship between DIFS and PIFS is illustrated in Figure 2.1. And the Extended IFS (EIFS) is the sum of the SIFS, DIFS, and the time needed to transmit an acknowledgment frame. If the wireless medium has been idle for a period of time equal to the DIFS, the device will pass the physical checking mechanism. If the previous frame received had an error, the interval must be equal to the EIFS. (Cooklev 2004).

In addition to physically listening for use of the medium before transmissions, a virtual carrier sense mechanism is also used. Each device has a special variable called a network allocation vector (NAV) to decide if the medium is busy. All frames that are sent wirelessly contain duration information on how long the frame will take to transmit. The NAV is updated with the duration information to specify the amount of time that must pass before the medium could be idle (O'Hara and Petrick 2005). The NAV value is continuously decremented through time. When the NAV reaches a value of zero, the virtual carrier sense mechanism reports the medium as free (IEEE

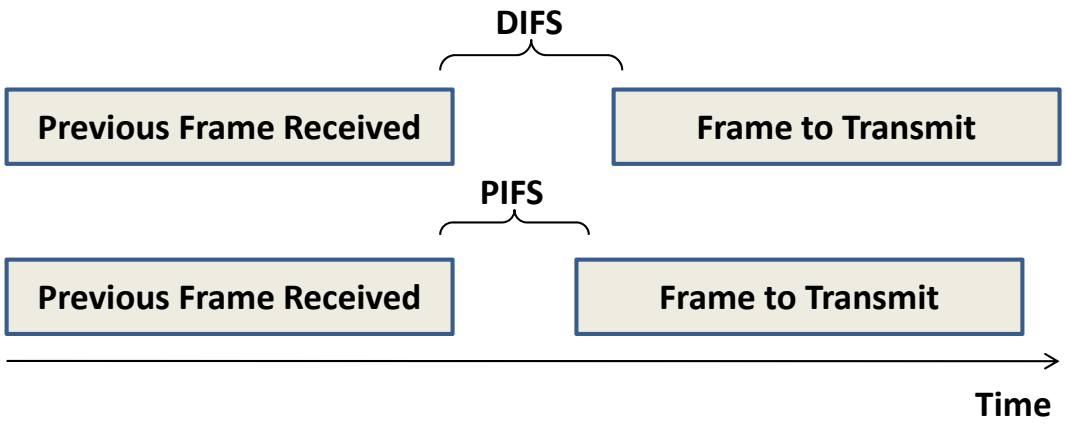

Figure 2.1: IFS for Distributed Coordination Function and Point Coordination Function.

802.11 2007).

When the MAC layer receives data to transmit from a higher layer in the networking stack, these two mechanisms are checked to attempt to avoid a collision on the wireless medium. If both indicate the medium should be free, a MAC frame is created and sent to physical layer for transmission. However, if one or both checks fail, the device does not transmit and the backoff algorithm is triggered. The algorithm is also triggered if the frame was sent, but an acknowledgment was not received by the destination device.

The backoff algorithm first increments a retry counter for the frame being transmitted. There are two different retry counters: one for long frames and one for short frames. After the correct retry counter is incremented, a random number is selected within a range called the Contention Window (CW). The CW represents the amount of time the medium must be idle before attempting transmission again. Each time the frame fails to transmit, the CW is doubled. The minimum and maximum values for the CW are specific to each physical layer (O'Hara and Petrick 2005).

Zigbee

Like IEEE 802.11, the IEEE 802.15.4 standard also uses CSMA-CA to regulate access to the wireless medium during a contention access period (Gutierrez, Callaway Jr., and Barrett Jr. 2007). However, two different types of CSMA-CA are used: slotted and unslotted. Slotted CSMA-CA is used in beacon-enabled networks and unslotted CSMA-CA is used in non-beacon-enabled networks. While the basic idea of the algorithms are the same, they differ in the timing details (IEEE 802.15.4 2006).

Slotted CSMA-CA is used when beacons are used by coordinators to synchronize communications in the network. Each period of time between beacons is subdivided into 16 time slots. As described previously, CSMA-CA generates random backoff intervals to aid in avoiding collisions in the air. Slotted CSMA-CA mandates that the backoff intervals directly align with the beginning of one of the 16 time slots. A contention window is also used for the slotted algorithm. The contention window represents the number of time slots that must pass with idle wireless activity to begin a transmission. The window is initialized at two and is decremented each time the channel is assessed to be free. When the window value reaches zero, the device may begin transmission. Each time the medium is determined to be in use, the window is reset to its initial value of two and continues until the maximum number of transmission attemps occurs (IEEE 802.15.4 2006).

Unslotted CSMA-CA is used in networks without beacons. Clearly, there is no requirement for backoff intervals to be aligned to slots since there are no time slots. As with slotted-CSMA the backoff interval increases each time a clear channel assessment fails (IEEE 802.15.4 2006).

The unslotted algorithm most closely compares with the CSMA-CA algorithm defined in IEEE 802.11. The terminology is slightly different, though. The contention window in IEEE 802.11 is similar to the backoff periods of IEEE 802.15.4. They both represent the amount of time required before a clear channel assessment is attempted again.

2.2 Contention-Free Access

Contention-free access is noncompetitive mechanism to use a shared medium. In the room of people, contention-free access could involve electing one person to act as a leader who tells each person when he can talk. Contention-free access can be useful if applications need a guaranteed quality of service or have low-latency requirements (Gutierrez, Callaway Jr., and Barrett Jr. 2007).

WiFi

The Point Coordination Function (PCF) is responsible for controlling contention-free access to the wireless medium. This is accomplished through the use of a single Access Point (AP) with which all devices can communicate. During a contention-free period, the AP has complete control over all wireless activity and no other device may interrupt. It follows a basic "speak only if spoken to rule" (O'Hara and Petrick 2005).

The series of events leading up to and during a contention-free period is as follows. First, devices must register with the AP in order to request contention-free access to the medium. This is necessary so the AP can maintain a list of devices to poll. Next, the AP must gain sole access to the medium. It accomplishes this through the contention access protocol described previously (the DCF) (O'Hara and Petrick 2005). As a result, the length of the contention-free period might be shortened (IEEE 802.11 2007). However, the AP has an advantage: the Point Interframe Space timing interval is shorter than the Distributed Interframe Space interval. Therefore, the AP (when beginning a contention-free period) has priority access to the medium (O'Hara and Petrick 2005).

When the AP has control, it sends a beacon to indicate to all other stations that a contention-free period has begun. This is the primary mechanism to ensure that no other devices will attempt to transmit during this time. The beacon contains the maximum expected length of the contention-free period, so all NAVs can be updated to reflect this. If a device attempts to transmit, it will fail the virtual carrier sensing mechanism. Additionally, since the PIFS is shorter than the DIFS, devices using CSMA-CA will not have the opportunity to transmit since they must wait longer. This is a backup method in case a device did not receive the beacon. To bring this about, the AP will always transmit within the PIFS time during contention-free

access. (O'Hara and Petrick 2005).

Devices must first register with the AP in order to be placed on a polling list. While it is not mandatory to use a contention-free period, all network stations must be able to recognize it is occurring and not conduct normal CSMA-CA operations. When a device is polled by AP to check if there is data to transmit, the device may transmit zero or one data frames in response. In order to increase efficiency, it is also possible for data delivered by the AP to a device to contain a poll request (O'Hara and Petrick 2005).

The AP will end a contention-free period by transmitting a MAC control frame indicating such. When received, all wireless devices can update their NAVs accordingly (O'Hara and Petrick 2005).

Zigbee

Much like an access point, the PAN coordinator is responsible for managing contention-free periods on a IEEE 802.15.4 beacon-enabled network. As previously described, 16 time slots are allocated between beacons. Some of these time slots can be directly allocated to a specific device. These slots are called Guaranteed Time Slots (GTS) and serve as mechanism to enable contention-free access (Gutierrez, Callaway Jr., and Barrett Jr. 2007).

Guaranteed Time Slots are only managed and used by the PAN coordinator of the network. Therefore, only devices that are in range and associated with the PAN coordinator may take advantage of GTSs. A Guaranteed Time Slot may extend over multiple slots of the 16 time slots in between beacons. A maximum of seven GTSs are permitted between beacons. Requests from devices for GTSs are handled with a first-come-first-served policy. The GTSs are required to be allocated in the slots after the contention-access period (IEEE 802.15.4 2006).

Devices can request GTSs using the mechanism pictured in Figure 2.2. A GTS request message is generated by a device and sent to the PAN coordinator. The PAN coordinator sends back an acknowledgment indicating the request was received. Some time after, a regular beacon frame is sent by the PAN coordinator. The beacon contains information about the Guaranteed Time Slots, so the device that requested the contention-free access can check if the request was granted (Gutierrez, Callaway Jr., and Barrett Jr. 2007).

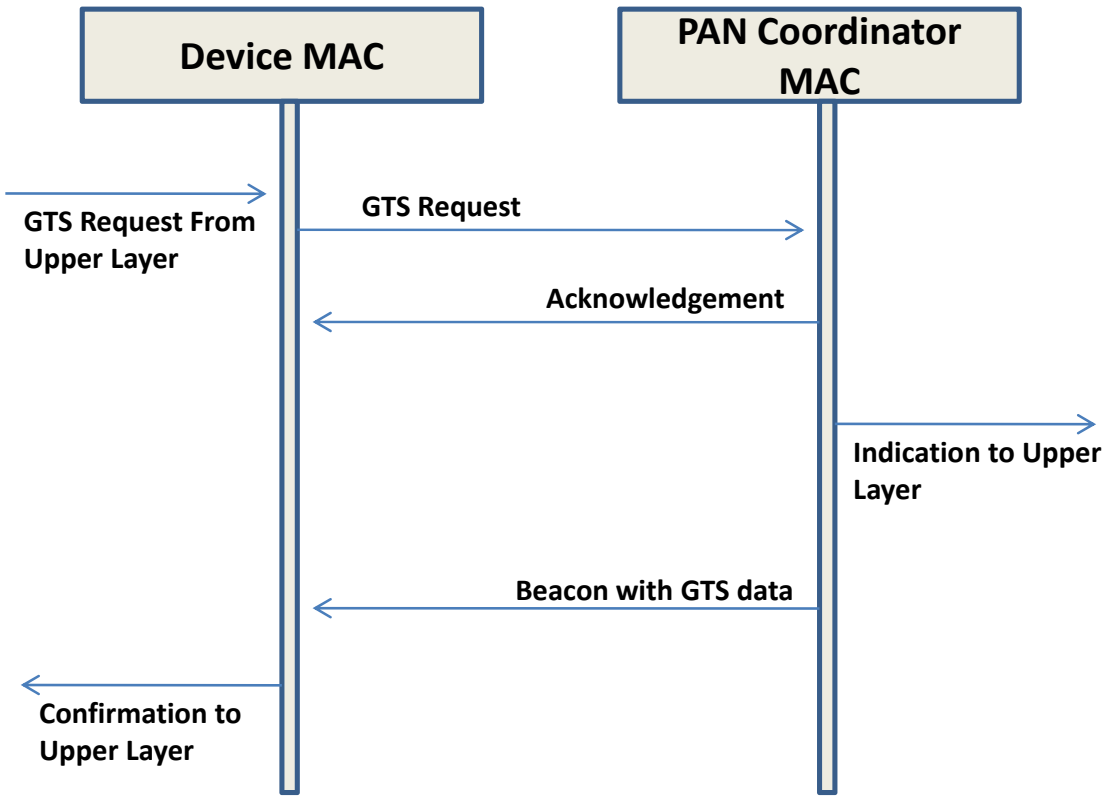

Figure 2.2: GTS Allocation Method.

The deallocation of GTSs can be initiated by either the device that requested the service or the PAN coordinator. In either case, after the GTS is released, the device still has the opportunity to communicate during the contention-access period. After several allocations and deallocations, it is possible for gaps to emerge between assigned GTSs. The PAN coordinator has the responsibility of removing the gaps (Gutierrez, Callaway Jr., and Barrett Jr. 2007).

2.3 The Hidden Node Problem

The CSMA-CA protocol works well when all devices are within range of one another. However, consider the scenario presented in Figure 2.3. It consists of three devices, two of which (A and C) attempt to send data to the third (B). Devices A and C are also not in range of each other.

Since Device A is out of range of Device C, it will never be able to detect a transmission from Device C (O'Hara and Petrick 2005). Therefore, the CSMA-CA protocol will not be effective if Device A attempts to transmit to Device B if Device C is also transmitting at the same time. The physical carrier sensing mechanism of Device A will report the wireless medium is free, when in fact it is in use near Device B. As a result, the two signals will collide and Device B will not be able to decipher the message (O'Hara and Petrick 2005).

Note that if contention-free access is used by Devices A and C to communicate with Device B, this problem would not occur. With WiFi, the AP would poll Devices A and B at different times and avoid a collision. With Zigbee, the devices would have their own Guaranteed Time Slots for transmission which would also avoid a collision.

The IEEE 802.11 standard for WiFi mitigates the hidden node problem by introducing small Request to Send (RTS) and Clear to Send (CTS) control messages. Before a device transmits data, it will first send a RTS message to the destination device. The destination will then send a CTS message back (O'Hara and Petrick 2005). However, the CTS is sent only if the NAV of the destination indicates the medium is free. Otherwise the CTS message will not be sent. The source will wait for a certain amount of time to pass to receive the CTS message. If the message is not received, the backoff algorithm is triggered (IEEE 802.11 2007).

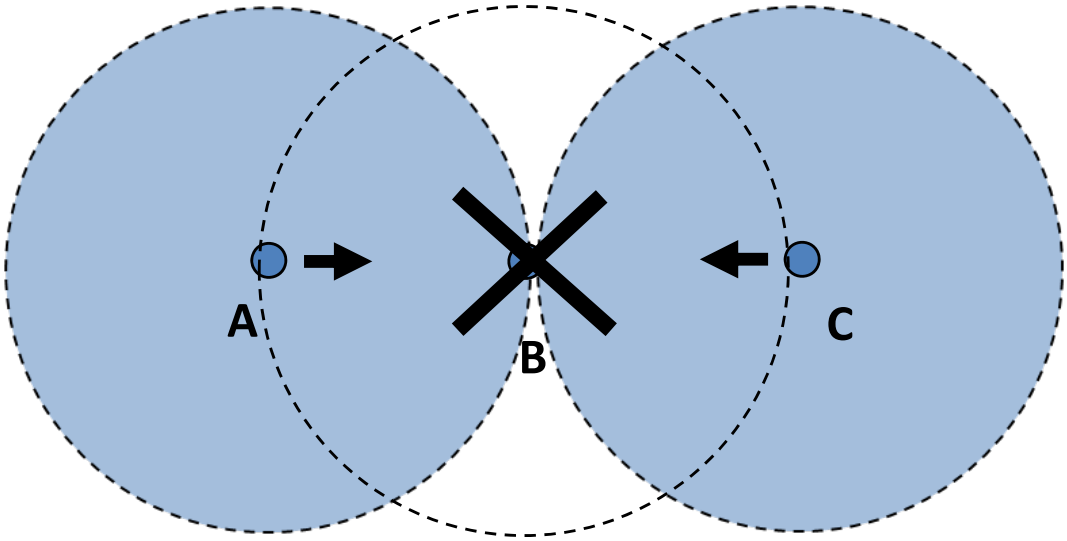

Figure 2.3: The Hidden Node Problem Scenario.

This exchange informs other devices in the vicinity of both the source and destination that the medium will be in use. When a device (that isn't the destination) receives a RTS frame, it will not use the medium until the requested frame is sent by the source. When a device receives a CTS frame, it will not use the medium until the acknowledgment is sent by the destination (O'Hara and Petrick 2005).

Clearly, this method to solve the hidden node problem introduces some overhead with the RTS and CTS messages. Therefore, the IEEE 802.11 standard allows for a minimum frame size threshold for the RTS/CTS procedure to be configured. Any frames of size less than the threshold will not be proceeded with a RTS frame. This enables network administrators to set an appropriate threshold based on the specific network geography, the prevalence of hidden nodes, and the data rate of the physical layer (O'Hara and Petrick 2005).

The IEEE 802.15.4 standard does not contain a solution to the hidden node problem; it ignores it. To keep things simple by reducing control overhead and saving battery life, the RTS/CTS method is not used (Hwang et al. 2005). As such, Zigbee is prone to many collisions and performance degradation due to hidden nodes. However, it is still possible to reduce the impact of hidden nodes by configuring network

settings. For instance, increasing the signal power of nodes will enable detection of transmissions (from a clear channel assessment) from a further distance. In beacon-enabled networks, increasing the time between beacons can also reduce collisions from hidden nodes (Harthikote-Matha, Banka, and Jayasumana 2007).

2.4 MAC Frame Formats

The different types of MAC Frame Formats for WiFi and Zigbee are discussed in this section.

WiFi

The IEEE 802.11 MAC accepts MAC Service Data Units (MSDUs) from upper layers in the networking stack. Headers and trailers are added by the MAC layer to form MAC Protocol Data Units (MPDUs) which are then passed to the physical layer for transmission. These MPDUs are called MAC frames. The general MAC frame is presented in Figure 2.4 (O'Hara and Petrick 2005).

All MAC frames contain the first three and the last field of the general frame. This includes the Frame Control, Duration/ID, Address 1, and the FCS fields. The Frame Control field specifies the protocol version, frame type, retry status, power management status, and other information. The purposes of the Duration/ID and Address 1 fields change based on the frame type, and the FCS is a cyclic redundancy check sequence to make sure all other fields in the frame were correctly transmitted (IEEE 802.11 2007).

The optional fields include three additional address fields, the Sequence Control field, and the Frame Body. Each address field can contain one of five address types: the transmitter address, receiver address, source address, destination address, or the BSSID. The Sequence Control field can be used to eliminate duplicate frames sent to a wireless device. The Frame Body is the first field that is not part of the MAC header and represents the actual payload. It has a variable length up to 2304 bytes without WEP encryption or 2312 bytes with WEP encryption (O'Hara and Petrick 2005).

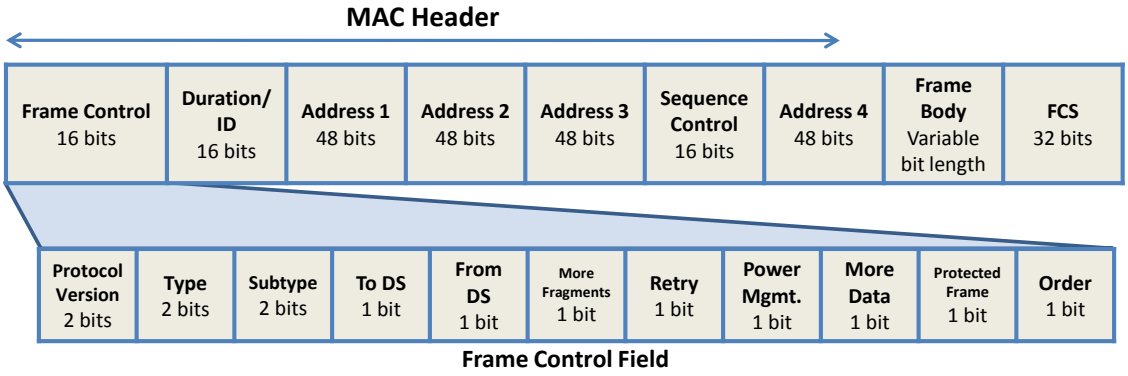

Figure 2.4: IEEE 802.11 General MAC Frame.

There are three main types of MAC frames: control, data, and management. Control frame subtypes include the Request to Send frame, Clear to Send frame, Acknowledgment frame, and others. Data frame subtypes include the simple data frame (that carries data only) as well as other data frames that also contain an acknowledgment or other management message. Some data frame subtypes do not include any data payload at all. Management frame subtypes include the beacon frame, probing frames, authentication and deauthentication frames, and others (O'Hara and Petrick 2005).

Zigbee

The general frame format of a IEEE 802.15.4 MAC MPDU is presented in Figure 2.5. All frames contain the Frame Control, Sequence Number, and FCS fields. Much like IEEE 802.11, the Frame Control field contains information on how the frame is interpreted. It includes subfields to specify the frame type, if an acknowledgment is requested, how the addresses are to be understood, if other data is pending, and more. The Sequence Number field is used to identify unique MAC frames and the FCS is a cyclic redundancy check for all fields before it (IEEE 802.15.4 2006).

The optional fields include the four addressing fields, the Auxiliary Security Header field, and the Frame Payload. The first addressing field is the Destination PAN Identifier which specifies the PAN identifier of the destination. The Destination Address field contains the 16-bit short address of the 64-bit full address of the intended re-

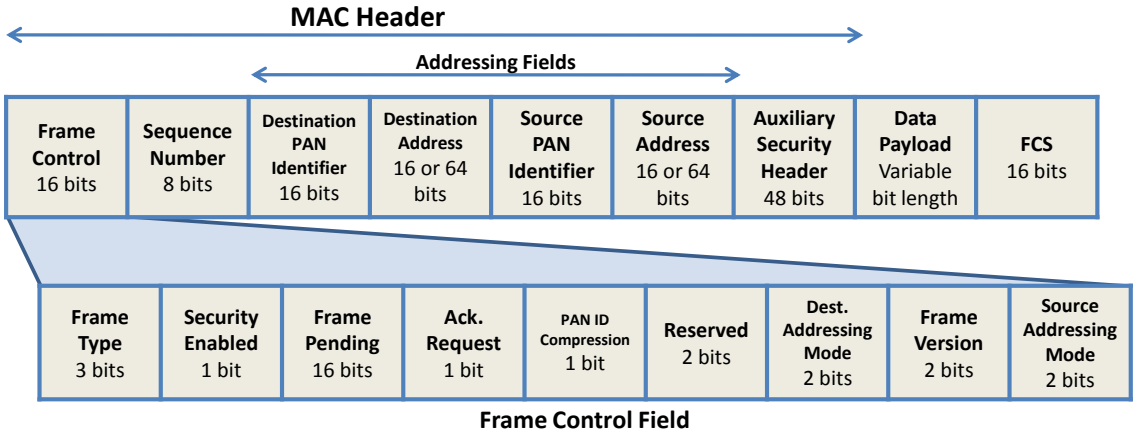

Figure 2.5: IEEE 802.15.4 General MAC Frame.

cipient. The Source PAN Identifier field specifies the PAN identifier of the sender. The Source Address field contains the 16-bit or 64-bit address of the sender. The next optional field after the addressing fields is the Auxiliary Security Header. It contains the information necessary to process the frame if security is enabled. The last optional field is the Frame Payload which is specific to the type of frame (IEEE 802.15.4 2006).

There are four types of frames defined in the IEEE 802.15.4 standard: beacon frames, data frames, acknowledgment frames, and MAC Command frames. Beacon frames are responsible for specifying the superframe structure and listing devices that have data pending at the coordinator. Data frames are used only for sending pure data. Acknowledgment frames are used to send back a notification that a frame was received properly. The acknowledgment frame contains only the three required fields. The last frame type, the MAC command frame, is used to issue association and disassociation requests, data requests, GTS requests, and other types of notifications (Gutierrez, Callaway Jr., and Barrett Jr. 2007).

A clear difference can be seen between the different types of MAC frames between WiFi and Zigbee. Whereas the IEEE 802.11 standard included three main frame types with many different subtypes, IEEE 802.15.4 has a total of just four types of frames. This speaks to the simplicity that Zigbee attempts to achieve. Additionally, Zigbee has a lower number of required fields which helps increase flexibility and efficiency. For example, IEEE 802.15.4 does not require an address acknowledgments, whereas

IEEE 802.11 requires the address of the recipient. This is eliminated in Zigbee since the Sequence Number field is adequate enough to process an acknowledgment.

2.5 Chapter Summary

In this chapter, the different mechanisms by which WiFi and Zigbee arbitrate channel access were examined. The contention access mechanisms of both protocols use CSMA/CA to detect when the medium is busy. However, Zigbee offers slotted and unslotted CSMA/CA depending on beaconed or beaconless operation. Additionally, both standards support contention-free access. In WiFi networks, APs act as a central arbitrator whereas PAN coordinators fill that role in Zigbee networks. The hidden node problem was also examined with regards to IEEE 802.11 and IEEE 802.15.4 networks. WiFi directly addresses the problem of hidden nodes by introducing RTS and CTS messages, while Zigbee ignores the problem completely. Finally, the frame formats of the two protocols were examined.

In the next chapter, the physical layer of WiFi and Zigbee is examined. The physical layer is directly below the MAC layer and completes the set of differences between the two protocols.

Chapter 3

The Physical Layer

The physical layer is at the bottom of the ISO/OSI seven-layer networking stack. It is responsible for converting the binary data received from the MAC layer into electromagnetic or optical signals for transmission over the medium. For wireless media, the physical layer is constrained to operate in portions of the electromagnetic spectrum defined by regulatory bodies and to operate in specific ways. In the United States, for example, the Federal Communications Commission oversees the allocation of the spectrum.

The original IEEE 802.11 specification defined three distinct physical layers. One of them is for infrared communication and is beyond the scope of this thesis. The other two, which are a Direct Sequence Spread Spectrum (DSSS) PHY and a Frequency Hopping Spread Spectrum (FHSS) PHY, represent different ways to handle wireless communication. The now common IEEE 802.11a, IEEE 802.11b, and IEEE 802.11g extensions to the standard were adopted after publication of the IEEE 802.11 standard (O'Hara and Petrick 2005). More attention will be given to the IEEE 802.11b standard, but discussion of the original DSSS and FHSS PHYs and IEEE 802.11g will also be included.

The IEEE 802.15.4 standard defines four physical layers. Two additional physical layers are defined in IEEE 802.15.4a.

In the remainder of this chapter, the two standards are compared based on how they deal with modulation, data rates, use of the electromagnetic spectrum, range of signal propagation, transmission power, and the structure of the PHY layer's protocol

data unit.

3.1 Modulation and Data Rates

The process of modulation involves converting source data to a form that can be transmitted over a physical medium. For digital communications, modulation can also be viewed as transforming binary data to a signal that can be sent by one wireless device and received by another (Rappaport 1996).

WiFi

Spread spectrum is a type of modulation to increase the signal-to-noise ratio by spreading the signal across many different frequencies. By allowing the signal to span multiple frequencies, the impact of possible of a narrow frequency band of interference. Spread spectrum techniques are used in the original DSSS and FHSS physical layers. The DSSS method translates individual bits (zeros or ones) into a sequence of bits called *chipping codes*. For instance, a binary 1 is converted to 0010011100 for transmission. Through this translation, the radio is able to operate with more outside interference since the chipping code can be heard more clearly (Geier 1999).

Consider the following example to show how chipping codes can increase the signal-to-noise ratio. The words "yes" and "no" are short and can be easily misheard. If the chipping code of "yes, you are right" is assigned to "yes" and "no, you are wrong" is assigned to "no", it would be easier to distinguish between the two words "yes" and "no" in a loud room. There is simply more data to decide which word was heard. The same is applied to binary ones and zeros.

The DSSS PHY of IEEE 802.11 is capable at transmitting at data rates of 1Mbps or 2Mbps. In order to achieve the different data rates, different modulation techniques are used, both a form of differential phase-shift keying, which is beyond the scope of this thesis (Cooklev 2004).

The IEEE 802.11 standard also defines a Frequency Hopping Spread Spectrum (FHSS) physical layer. Using this method, the signal is spread by periodically changing the broadcast radio frequency. This increases the signal-to-noise ratio by using a

different frequencies. If there is interference on one frequency, only a small fraction of time will be used to transmit on that frequency before it is changed again. A hopping code specifies which frequencies will be used and their sequence. This allows different stations using FHSS to share the same central frequency, but use a different hopping code to avoid interference with each other. A device wishing to receive the signal must use the same hopping code as the sender (Geier 1999).

Figure 3.1 depicts how frequency hopping works for IEEE 802.11. However, the figure has been greatly simplified since there are 79 distinct channels of 1Mhz width in the 2.4Ghz band and it does not represent any defined IEEE 802.11 hopping code. Like the DSSS PHY, the PHSS PHY can transmit at data rates of 1Mbps or 2Mbps (O'Hara and Petrick 2005).

In addition to the three original PHYs defined in IEEE 802.11, several amendments to the standard have been published. The IEEE 802.11b, IEEE 802.11a, and IEEE 802.11g all define new physical layers that can be used (IEEE 802.11 2007).

The IEEE 802.11b physical layer uses DSSS with a more advanced modulation technique called complementary code keying (CCK) that allows it to achieve 5.5Mbps and 11Mbps data rates in addition to the original DSSS PHY rates of 1Mbps and 2Mbps (Cooklev 2004). A metric called the *bit error rate* (BER) reflects the number of bits incorrectly received per unit of time. It is affected by how resistant to propagation effects the modulation technique is at different rates. The BER is also affected by the strength of the received signal. Figure 3.2 depicts the BER at different radio signal strengths for high-rate IEEE 802.11b. Since modulation does not occur at a packet-level simulation, this information is used in simulation predict the chance that an error occurs in transmission for different signal strengths.

IEEE 802.11a does not use spread spectrum technology. Instead, it uses a modulation technique called orthogonal frequency division multiplexing (OFDM). OFDM works by transmitting data in narrow, overlapping channels. This allows it to achieve much higher data rates and use the wireless spectrum more efficiently. The IEEE 802.11a PHY can achieve a maximum data rate of 54Mbps (Cooklev 2004).

IEEE 802.11g uses the same OFDM technology to transmit wireless data as IEEE 802.11a and achieve the same maximum data rate. The other differences between IEEE 802.11a and IEEE 802.11g will be discussed in subsequent sections (Cooklev 2004).

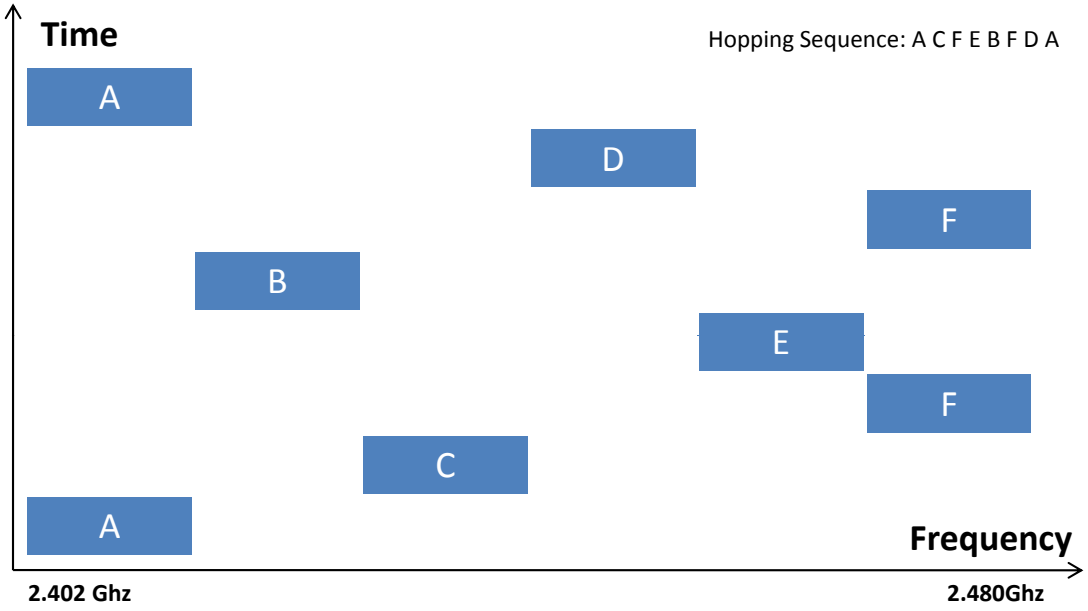

Figure 3.1: Simplified IEEE 802.11 FHSS Hopping Sequence.

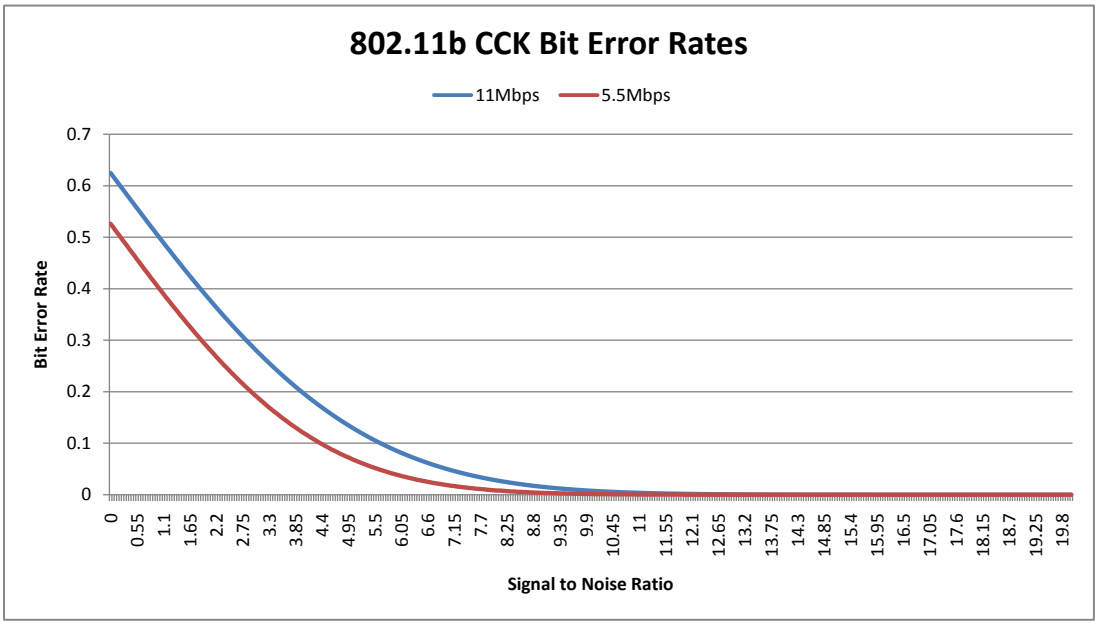

Figure 3.2: IEEE 802.11b CCK Bit Error Rates.

Zigbee

There are four different PHY layers defined by the IEEE 802.15.4 standard and another two defined by the IEEE 802.15.4a amendment. The first physical layer uses DSSS with BPSK modulation. It achieves a data rate of 20Kb/s when operating in the 868Mhz band and 40kb/s operating on the 915Mhz band. The second PHY reaches a data rate of 100Kb/s in the 868Mhz band and 250Kb/s in the 915Mhz band by using quadrature phase-shift keying (O-QPSK) modulation. The third achieves a data rate of 250Kb/s in the 2.4Ghz band also using O-QPSK (IEEE 802.15.4 2006).

The last physical layer defined by the IEEE 802.15.4 standard uses a different kind of spread spectrum technology called parallel sequence spread spectrum (PSSS). In order to achieve the data rate of 250Kb/s in the 868Mhz and 915Mhz bands, amplitude shift keying (ASK) modulation is used (IEEE 802.15.4 2006).

IEEE 802.15.4a introduces two new physical layers: a chirp spread spectrum PHY that can achieve a data rate of 1Mbps and a ultra-wide band PHY that can operate at a variety of frequencies (IEEE 802.15.4 2006). They will be omitted from further

discussion.

The data rates for Zigbee are much lower than WiFi. The slowest WiFi physical layer is still four times faster than the best original Zigbee PHY. Additionally, the IEEE 802.11a and IEEE 802.11g standards abandoned spread spectrum technology to achieve higher data rates. The priorities of WiFi and Zigbee are clearly illustrated to be different.

3.2 Frequencies of Operation and Channels

Both WiFi and Zigbee operate in the Industrial, Scientific, and Medical (ISM) bands in the United States. Since the ISM band is unlicensed, it is easy to deploy new wireless networks without consideration to frequency planning, regulations, and licenses. The ISM band is shown in Figure 3.3 (Geier 1999).

The frequency of radio communication and the number of channels is important to a packet-level simulation. The effects from different operating frequencies can be quantified in a simulation. For instance, some frequencies can be less prone to interference, but have a greater multi-path propagation effects and shorter range (Geier 1999). Additionally, the number and spacing between channels is important to determine the amount of overlap interference between channels.

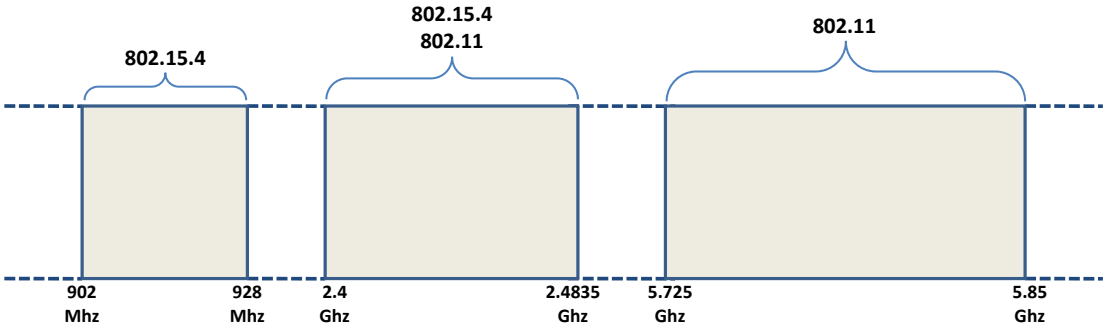

Figure 3.3: ISM Frequency Bands.

WiFi

The original DSSS PHY of IEEE 802.11 operates in the 2.4Ghz band with 14 channels available for use. Each channel requires 22Mhz of bandwidth and channels are spaced 5Mhz apart. This allows for three non-overlapping channels to be used simultaneously, which are spaced 25Mhz apart. This is shown in Figure 3.4. The IEEE 802.11b extension uses the same frequency and channel mappings as the original IEEE 802.11 DSSS PHY (O'Hara and Petrick 2005).

The FSSS layer operates in the 2.4Ghz band. Since the spreading method is frequency hopping, there are many more channels available compared to DSSS. In North America and much of Europe, 79 channels are defined each with a bandwidth of 1Mhz. Three sets of non-overlapping channel-hopping sequences are defined in each worldwide region to specify which channels are used (O'Hara and Petrick 2005).

While IEEE 802.11b devices operate in the 2.4Ghz band, IEEE 802.11a operates in the 5Ghz band. The IEEE 802.11a specification uses a total of 12 non-overlapping channels of 20Mhz bandwidth each. IEEE 802.11g operates in the 2.4Ghz band. As with IEEE 802.11a (which also uses OFDM), the total amount of bandwidth required per channel is 20Mhz. IEEE 802.11g uses the same channel assignments as IEEE 802.11b, and provides for three non-overlapping channels spaced 25Mhz apart (O'Hara and Petrick 2005).

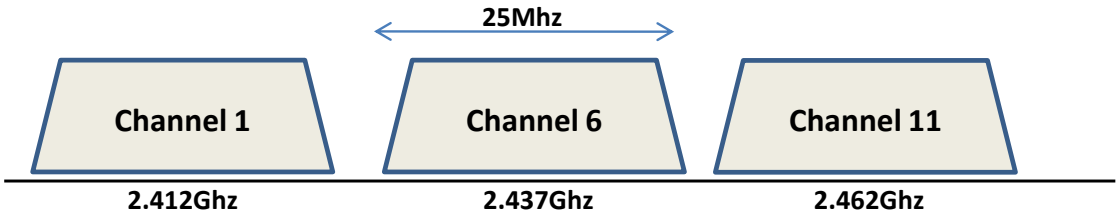

Figure 3.4: IEEE 802.11 DSSS PHY Channel Frequency Spacing

Zigbee

There are three frequency bands defined in the IEEE 802.15.4 standard: the 868Mhz band, 915Mhz band, and the 2.4Ghz band. The 868Mhz band is available for use in Europe, the 915Mhz is available in North America, and the 2.4Ghz band is available worldwide (Gutierrez, Callaway Jr., and Barrett Jr. 2007).

The IEEE 802.15.4 standard uses *pages*, sets of channel numbers, to define channel assignments. A maximum number of 32 pages may be defined, but only three are currently specified in the standard. The remaining pages are reserved for future use. Each page contains a maximum of 27 non-overlapping channel assignments, numbered 0 to 26 (IEEE 802.15.4 2006).

Page 0 specifies channel assignments for the 2.4Ghz and 868/915Mhz BPSK PHYs. It allocates one channel for the 868Mhz band, 10 for the 915Mhz band, and 16 channels for the 2.4Ghz band. Channels are spread evenly through the available spectrum and spaced 2Mhz apart for the 868/915Mhz bands and 5Mhz apart for the 2.4Ghz band (IEEE 802.15.4 2006). The channel spacing for the 2.4Ghz PHY is shown in Figure3.5.

Page 1 specifies channel assignments for the 868/915Mhz PSSS PHY. It allocates one channel in the 868Mhz band and 10 in the 915Mhz band. Channel assignments for Page 2 are identical to Page 1, except it describes the 868/915Mhz O-QPSK PHY (IEEE 802.15.4 2006).

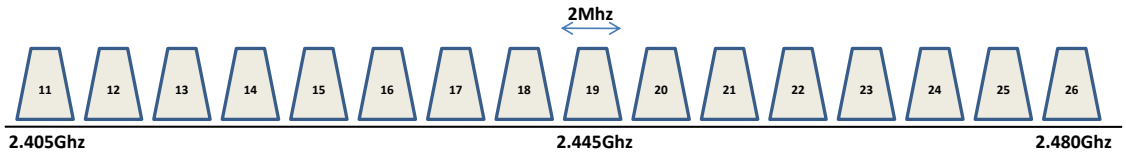

Figure 3.5: IEEE 802.15.4 2.4Ghz PHY Channel Frequency Spacing

3.3 Range and Power

The range of signals and the power needed to operate the device are important factors in distinguishing between different wireless standards. Given the different purposes and applications of WiFi and Zigbee, it is expected that the power requirements and operating range will be different.

The range of a radio signal depends on the amount of power used for transmission. The relationship between the two is positive: the more power used, the greater the range. The amount of power used for transmission is important because every wireless device has a specific minimum receive power level. Data that is received below a set signal strength cannot be interpreted. It is possible to calculate the maximum distance a signal can be transmitted using power loss equations. In free space, the power loss (in dBm) is given by

$$P_r = P_0 - 10n \log d$$

where P_0 is the received power at a 1m distance, n is a factor dependent on the environment (typically 1.6 to 3.3 indoors), and d is the is the distance between devices (Cooklev 2004). To convert from watts to dB, multiply the log base 10 of Watts by 10.

The power requirements for a typical wireless network device can be modeled by a finite state machine given like the one in Figure 3.6, which is a representation of the following model. In general, power consumed when the radio is off is less than the power needed to receive, which is less than the power needed to send. In fact, the power consumption for each state of the machine can be modeled with equations. The general cost equation is given by

$$Cost = m \times size + b,$$

where m is an incremental cost based on the size of the packet and b is a fixed cost for channel acquisition. The cost to send is given by

$$Cost = b_{sendctl} + b_{recvctl} + m_send \times size + b_send + b_{recvctl},$$

where m_{send} is the incremental cost to send the data, $size$ is the size of the data packet, b_send is the cost for channel acquisition for the data message, and all other variables are fixed cost control messages (acknowledgments and RTS/CTS traffic). Likewise, the cost to receive a packet is given by

$$Cost = b_{recvcts} + b_{sendctl} + m_{recv} * size + b_{recv} + b_{sendctl}.$$

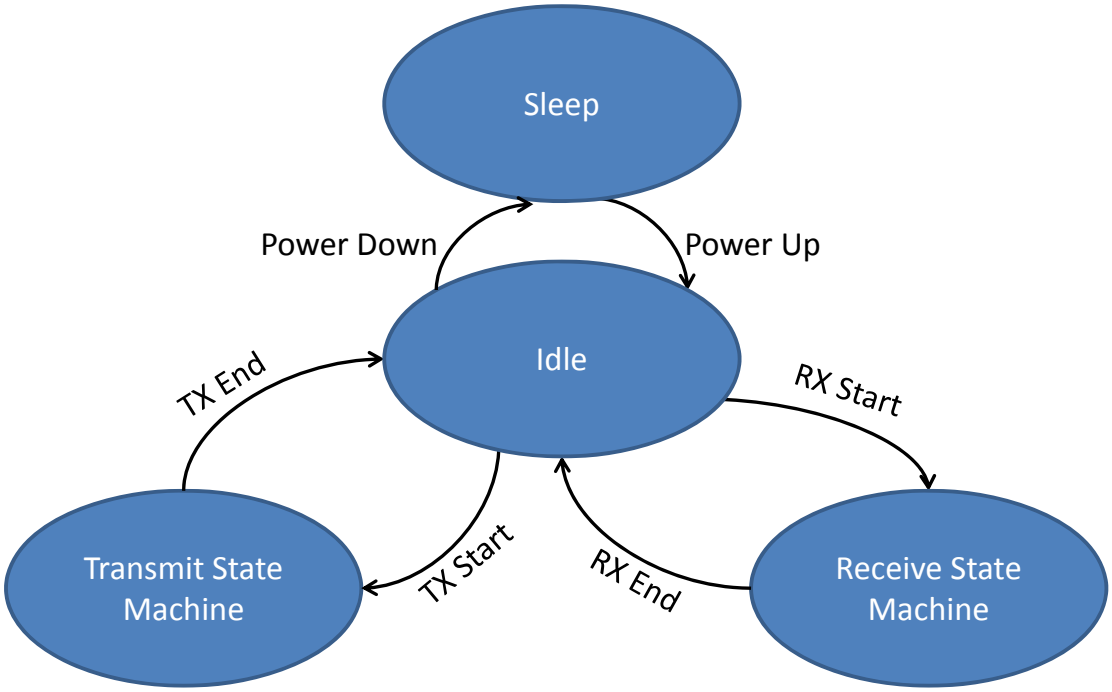

Figure 3.6: Physical Layer State Machine.

By conducting a power analysis of network cards, it is possible to determine the values for each of the variables and use them in a simulation to model energy consumption (Feeney 2001).

WiFi

In Feeney 2001, such a power analysis is conducted for WiFi. Through measuring power consumption by a IEEE 802.11 network card, the constants for the above variables are determined. This allows a simulation model to be constructed using the combination of the equations and the experimentally determined constant values. In addition to generalizing the power model, it is useful to dig deeper to discover the specifications of the IEEE 802.11 standard that relate to power consumption.

The maximum power output allowable by US regulations is 1 watt in the 2.4Ghz operating frequency. However, many vendors have set the default output power to 100mW in order to conserve power and abide by European regulations. In open areas, IEEE 802.11b can achieve an operating range of about 1000 feet, but this is significantly lowered to about 150-300 feet indoors. (Cooklev 2004).

In order to conserve battery life, IEEE 802.11 devices can enter a low power mode. The mechanism by which this occurs is different for ad-hoc and infrastructure networks. In an ad-hoc network, a device can send an indication to any other wireless device to signal that it will enter a low-power state. A device in a lower-power state must wake to receive beacon frames and stay awake for a specific amount of time after a beacon frame is received (O'Hara and Petrick 2005).

Nodes in an infrastructure network can achieve even better power management since all communication is routed through the AP. The AP can buffer all data that must be sent to a device in a low-power mode. A device that wishes to enter a low-power mode must inform the AP of the amount of time the device will be asleep. It is not necessary for a device to wake every beacon transmission as in the ad-hoc network (O'Hara and Petrick 2005).

Zigbee

Similar to (Feeney 2001), a IEEE 802.15.4 power model is presented in (Mura, Paolieri, and Fabbri 2007). However, the data presented is not compatible with the cost equations above. Instead, the equivalents of m_{send} and m_{recv} are given in terms of power, not energy per byte. In order to use the same cost equations, all instances of $m_{recv/send} \times size$ must be replaced with $p_{recv/send} \times time$ where $p_{send/recv}$ is the transmit or receive power and $time$ is the amount of time needed to send. As with WiFi, it is necessary to explain what drives power consumption according to the IEEE 802.15.4 standard.

The minimum output required by IEEE 802.15.4 for a wireless device is -3dBm. Since the standard was especially written with low power consumption in mind, most devices will transmit between 0dBm and 10dBm although a maximum of 1W (30dBm) is allowed by the FCC. A lower power output is encouraged in order to reduce interference between devices and save energy (IEEE 802.15.4 2006). However, lowering the transmit power will also decrease the visibility of nodes and contribute to the hidden node problem, discussed in Chapter 2.

Like IEEE 802.11, IEEE 802.15.4 specifies a mechanism by which devices can operate under a low-power mode. When operating with beacons, a coordinator can split the time between beacons into an active and an inactive portion. No data is exchanged during the inactive period. This allows coordinators and other devices to enter a low-power mode and awake at the beginning of the next active period (Gutierrez, Callaway Jr., and Barrett Jr. 2007).

3.4 PHY Packet Structure

WiFi

There are two sublayers within the IEEE 802.11 physical layers. The physical layer convergence procedure (PLCP) sublayer is responsible for communicating with the MAC layer. The physical medium dependent (PMD) sublayer is responsible for transmitting the binary data received by the PLCP sublayer (O'Hara and Petrick 2005).

In IEEE 802.11, physical layer frames are called PLCP protocol data units (PP-

DUs). They consist of a PLCP preamble, PLCP header, and a MAC protocol data unit (MPDU) or PLCP service data unit (PSDU). The PLCP preamble is used for signal acquisition and demodulation purposes and the PLCP header specifies information about the MPDU. Different physical layers may have different PPDU formats, but backwards compatibility is sometimes a goal. For example, the IEEE 802.11 DSSS PHY PPDU is compatible with the IEEE 802.11b PHY long format PPDU. This frame format is presented in Figure 3.7.

There are seven fields that make up a PPDU, each with a specific purpose. The sync field is used to acquire the signal. It is encoded as a sequence of all scrambled binary ones. The SFD field is the start of frame delimiter which is used to mark the beginning of the frame header. The signal field is used to specify the modulation and data rate of the PSDU. The service field contains more modulation details in addition to timing details. The length field specifies the number of microseconds it will take to transmit the PSDU. The CRC field is a cyclic redundancy check over all other fields in the PLCP header. Finally, the PSDU is the data payload associated with the frame (O'Hara and Petrick 2005).

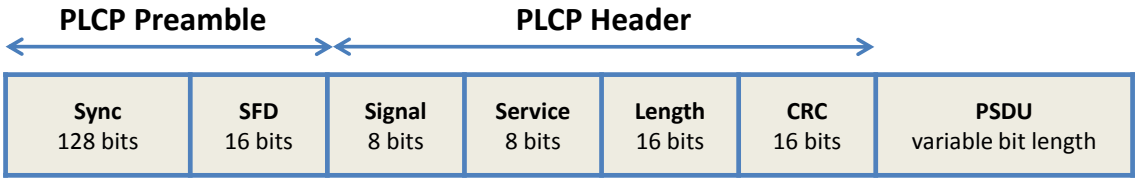

Figure 3.7: IEEE 802.11 DSSS/IEEE 802.11b Long PPDU Frame Format

Zigbee

Three main fields divide the IEEE 802.15.4 PPDU: the synchronization header, the PHY header, and the PSDU. The PPDU frame format for the 2.4Ghz O-QPSK PHY is presented in Figure 3.8. Frame formats for the other PHYs are identical in layout but differ only in the bit lengths of the synchronization header fields (IEEE 802.15.4 2006).

There are five fields that make up a IEEE 802.15.4 PHY frame. The preamble field is comparable to the sync field of IEEE 802.11. It is used to synchronize the received. The SFD field is the start of frame delimiter to indicate the division between the preamble and frame length fields. The frame length field indicates the number of octets in the PSDU. Finally, the PSDU is the data payload for the frame (IEEE 802.15.4 2006).

The IEEE 802.15.4 standard does not include many fields seen in the IEEE 802.11 PPDU. This speaks to the simplicity of implementation and operation Zigbee strives to achieve.

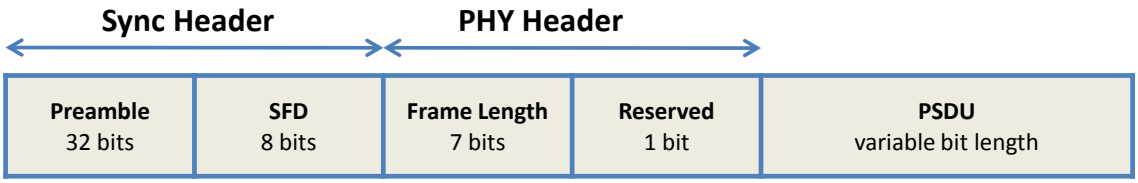

Figure 3.8: IEEE 802.15.4 2.4Ghz O-QPSK PPDU Frame Format

3.5 Chapter Summary

In this chapter, the different physical layers of WiFi and Zigbee were examined. Each standard defines multiple PHYs that can use different techniques to modulate, and send the signal. The differences in modulation determine how the signal is encoded for transmission, the resulting data rates, and the probability of producing errors. In general, IEEE 802.11 uses more complex ways to modulate the signal to achieve higher data rates than Zigbee. The operating frequencies of each wireless protocol also differ. Wifi and Zigbee both operate in the 2.4Ghz band, but IEEE 802.11 also allows operation in the 5Ghz band while Zigbee offers operation in the 868Mhz and 915Mhz bands. The operating range and power requirements of both protocols differ based on the uses for which they are intended. Finally, the physical layer packet structure of both protocols have many commonalities such as preambles and start of frame delimiters.

In the next chapter, the details of implementing the IEEE 802.15.4 standard in an existing network simulator are discussed.

Chapter 4

Simulating Zigbee

4.1 Introduction to SWAN

The Simulator for Wireless Ad-hoc Networks (SWAN) is an ongoing research project at Bucknell led by Professor Felipe Perrone. Its purpose is to simulate wireless ad-hoc networks in order to further understand their properties. SWAN is built in C++ and uses the Dartmouth Scalable Simulation Framework (DaSSF) for event scheduling, statistics collection, and other low-level functions (Perrone 2009).

SWAN simulations are easily configurable by end users. In order to run experiments, parameters must first be supplied to SWAN. This is accomplished through the use of a special text file written using the Domain Modeling Language (DML). A DML file consists of a sequence of key-value pairs that specify the parameter to configure and the value it will hold. These parameters include the number of wireless devices in the simulation, their positions in 3D space, and their mobility patterns. A DML file for the simulation of a IEEE 802.11 wireless network is presented in Figure 4.1 (Perrone 2009). Simulations are easily configurable by end users. In order to run experiments, parameters must first be supplied to SWAN. This is accomplished through the use of a special text file written using Domain Modeling Language (DML). A DML file consists of a sequence of key-value pairs that specify the parameter to configure and the value it will hold. These parameters include the number of wireless devices in the simulation, their relative distances, and their mobility patterns (Perrone 2009).

```
graph [
   session [
     name "app" use "tstapp.sess-app-session"

     traffic_gen [
     start_traffic 10.0
     packet_size 512
     session_len 60
     traffic_model constant
     bit_rate 3072.0
   ]]

   session [
     name "aodv" use "routing.aodv_sim.swan-aodv-session"
     use_rrep_ack false          # using MAC802.11 link level ACK
     use_hello_msg false         # using MAC802.11 link level ACK
     active_route_timeout 10     # 10 second lifetime
     rreq_retries 2              # number of retries of rreqs.
   ]

   session [ name "icmp" use "net.icmp-session" ]

   session [ name "net" use "net.ip-session" ]

   session [ name "arp" use "net.arp-session" ]

   interface [
     id 0 netid 1
     session [ name "mac" use "mac.mac-802-11-session" ]

     session [
       name "phy" use "phy.phy-802-11-session"
       bandwidth 11e6
     ]]]
]
```

Figure 4.1: Sample DML Configuration File.

The SWAN architecture follows directly from the ISO OSI seven-layer networking stack, which is discussed in the introduction and presented in Figure 1.1. Each layer in the simulation model of the networking software corresponds to an instance of a class that inherits from the `ProtocolSession` base class. For each layer in the networking stack, there can exist multiple implementations that can conform to different standards. As long as they inherit from `ProtocolSession`, they can be used as a layer in the stack. The classes that inherit from `ProtocolSession` are encapsulated by the `ProtocolGraph` class, which represents the entire networking stack (Perrone 2006).

Any child class of `ProtocolSession` requires four methods to be implemented: `pop`, `push`, `control`, and `config`. The `push` and `pop` methods are used to send and receive information from adjacent layers as shown in Figure 4.2. A layer that must send a message down the stack calls push on the layer below it. This occurs when data is sent to another device. To transmit data up the stack, `pop` is called on the layer above. This occurs when data is received from another device. The `control` method is used to pass simulation-related data outside of the network framework. The config function is used to parse the DML file and extract the parameters needed to configure the simulation (Perrone 2006).

At this point, SWAN only has PHY and MAC models for IEEE 802.11b wireless networks. However, this is not a limitation of any component of the simulator. In fact, SWAN makes it easy to incorporate other wireless standards into its architecture. Other standards simply have not been coded to conform to the specifications of SWAN. The simulator would be more useful if it were able to simulate other standards. Since a lot of research focuses on the family of IEEE 802.11 standards, IEEE 802.15.4 was selected as the second possible standard to simulate in SWAN. This also comes from the widespread interest that the community has on the development of wireless sensor networks, which might be using Zigbee.

4.2 From WiFi to Zigbee

In this section, two examples are given to demonstrate the type of work required to transform the existing IEEE 802.11b simulation model to an IEEE 802.15.4 SWAN model.

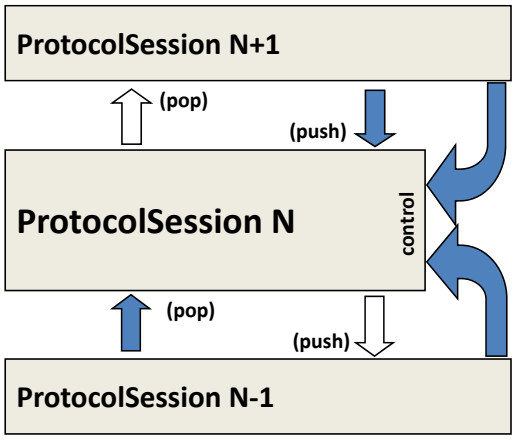

Figure 4.2: The SWAN ProtocolSession API.

4.2.1 Power Model

As discussed in Section 3.3, Feeney (2001) provides a power model for IEEE 802.11 wireless networks. These values are used in SWAN to determine the amount of energy that must be deducted from the battery following a radio operation. The code is presented in Figure 4.3. The code is fairly straightforward; since Feeney (2001) gives the cost values ($m_{send/recv}$) in energy per byte, the duration of the radio operation must be converted to number bytes sent or received during that time interval. This is easily accomplished multiplying the bandwidth by the time duration. Then, based on the operation, the power is deducted according to the specific radio operation and its associated m cost.

Mura, Paolieri, and Fabbri (2007) provides a power consumption model for IEEE 802.15.4 which documents the power requirements for sending and receiving in terms of power units mW. Therefore, in order to adapt the SWAN code to represent the IEEE 802.15.4 power model, the duration multiplied by the power required for the radio operation (send/receive) is the amount of energy to subtract from the battery. Unfortunately, the bandwidth at which the power constants were obtained was not given, but it can be assumed to be 250kbps.

```
void Host::battery_decrement(int mode, double duration, int iface) {
   if (battery_alive) {

     Phy80211Session* myPHY = (Phy80211Session*)getLowestSession(iface);
     assert(myPHY);

     double bwidth = myPHY->getBandwidth();
     double bytes = duration * bwidth / 8;

     switch(mode) {
     case POWER_PTP_SEND:
       battery_power -= (PowerConstant::mSend(bwidth) * bytes) +
         PowerConstant::bSend(bwidth);
       break;
     case POWER_BROAD_SEND:
       battery_power -= (PowerConstant::mBroadSend(bwidth) * bytes) +
         PowerConstant::bBroadSend(bwidth);
       break;
     .
     .
     .
     case POWER_IDLE: { // "Idle" consumption (no communication)
       battery_power -= (PowerConstant::idle(bwidth) *
         (VirtualTime(getNow()).second() - last_status_change.second()));
       last_status_change = getNow();
     } break;
     default:
       error_quit("Host::battery_decrement invalid mode");
     }

     if (battery_power <= 0){
       reboot(INFINITY);
       battery_alive = false;
     }
   }
```

Figure 4.3: SWAN Code Showing Battery Consumption.

4.2.2 Bit Error Rate Model

Discussed in Section 3.1, the bit error rate (BER) measures the number of bytes received incorrectly per unit of time. If bits have been corrupted, the received packet of data must be discarded. The BER depends on the type of modulation used and the signal to noise ratio. For higher signal to noise ratios, the BER is reduced.

The SWAN code that uses the BER to determine packet loss is presented in Figure 4.4. No bit errors actually happen during a simulation since it is running on a computer. However, it is possible to calculate the probability that they would occur on a real network. This probability is based on the BER and the size of the packet. The BER is determined through a lookup into the graph shown in Figure 3.2 for IEEE 802.11b CCK.

```
bool FixedRangeRadioNetwork::acceptFrame(Modulation m, double
signalPower_dBm, double noisePower_dBm, int size)
{
   double BER = compute_BER(m, signalPower_dBm, noisePower_dBm);
   double p, DBPSK_BER;

   // simplified computation for frame error rate
   p = (double) 1.0 - pow((1.0 - BER), (double) size);

   if (rng->bernoulli(p)) {
     return false;
   }
   else {
     return true;
   }
}
```

Figure 4.4: Simplified SWAN Code Showing Packet Acceptance.

Different modulation techniques are used by Zigbee. For instance, the 2.4Ghz PHY uses O-QPSK modulation instead of CCK. Therefore, the data in Figure 3.2 will need to be replaced by a different function specific to O-QPSK. The BER equation for Zigbee PHYs can be found in IEEE 802.15.4 2006.

4.3 The *ns-2* Implementation

The *ns-2* network simulator is a versatile discrete-event network simulator. Its main purpose was to facilitate the research of many different network protocols and was designed to be capable of scaling the simulation to handle a large number of network nodes. In addition to the large number of protocols it could simulate, it also had wide developer support (Breslau et al. 2000).

The *ns-2* simulator uses a unique split-language programming model. To set up and configure a simulation, *ns-2* uses an interpreted language called OTcl to describe the experiments. Whereas the C++ core of *ns-2* is mainly used for packet processing and coding of the major network functions, OTcl is used to configure the simulation and directly manipulate the C++ classes. It is very different from a SWAN DML file which defines the structure of the model and also assigns values to its parameters (Fall and Varadhan 2009).

The *ns-2* IEEE 802.15.4 implementation is written in C++ and is capable of running on *ns-2* 2.26 and above. It implements most of the features of Zigbee including pure and slotted CSMA-CA, star and peer-to-peer network configurations, and both beacon and beaconless operation (Zheng 2004). The Guaranteed Time Slot feature of Zigbee to produce contention-free periods is not supported (Zheng and Lee 2004).

There are several reasons why using *ns-2* is undesirable to simulate IEEE 802.15.4 networks. First, the architectural purity of the simulator has suffered under the large amount of development for the platform. The simulator core was not general enough in order to simulate all the different models that were developed. For instance, *ns-2* was not able to simulate wireless networks until a second node model was introduced (Perrone et al. 2009).

Second, it is difficult to configure a scenario to simulate. Due to the complexity of the split-level programming model, it is not always clear where configuration parameters are to be specified. Sometimes they must be given in the OTcl, other times they must be explicitly set in C++ header files. Poor and out of date documentation exacerbates this problem (Perrone et al. 2009).

Finally, *ns-2* lacks tools to analyze simulation results. The simulator produces packet traces in the form of text files that can be difficult to interpret. These trace files can be very large even for relatively small simulations. The lack of proper analysis tools often requires individual researchers to create specialized programs to interpret

the packet traces (Cicconetti et al. 2006).

For these reasons, it would be best to develop an IEEE 802.15.4 implementation in SWAN.

4.4 Challenges in a SWAN Implementation

The implementation of the IEEE 802.15.4 standard in SWAN was much more difficult than originally anticipated. Working from an existing implementation of Zigbee on the *ns-2* network simulator, the most desirable final product is a port of the code for that simulator into SWAN.

This thesis explains the basic architecture of the SWAN code and discusses the implementation details of the *ns-2* code. The implementation of the physical layer for IEEE 802.15.4 in SWAN is drafted here and its further development will need to address the following three main challenges.

4.4.1 Conflicting Data Types

SWAN and *ns-2* have different type specifications for similar data structures. For example, in the implementation of IEEE 802.15.4 in *ns-2*, the physical layer sends and receives `Packet` objects that hold header bits and data for the application level. In the SWAN implementation, similar data is stored in `RadioFrame` objects. The `RadioFrame` object stores information about the source and destination MAC addresses, transmission duration, and other information in explicitly named variables, unlike a *ns-2* `Packet` object. The `RadioFrame` and `Packet` class definitions are presented in an abbreviated form in Figure 4.5. In order to port the *ns-2* implementation to SWAN, all references to *ns-2* objects in its IEEE 802.15.4 will need to be changed to their SWAN counterparts. This requires considerable effort in the analysis of when a data type must be translated.

```
class Packet : public Event {
private:
 friend class PacketQueue;
 u_char* bits_;
 u_char* data_;
 u_int datalen_;
protected:
 static Packet* free_;
public:
 Packet* next_;
 static int hdrlen_;

 Packet() : bits_(0),
  datalen_(0), next_(0) {}
};

class RadioFrame : public SSF_Event {
public:
 int netid;
 MACADDR to_addr;
 MACADDR from_addr;
 ltime_t duration;
 void* frame;
 int framelen;
 float xpos;
 float ypos;
 float zpos;
 double tx_power_dbm;
 double tx_antenna_gain_db;
 RadioFrame(int nid, MACADDR from, MACADDR to,
  ltime_t duration, void* frame, int framelen,
  boolean need_copying = false);
};
```

Figure 4.5: Comparison of SWAN and *ns-2* Data Types

4.4.2 Poor *ns-2* IEEE 802.15.4 Documentation

In addition to mismatched data types, the poor state of the IEEE 802.15.4 implementation documentation makes it much more difficult to port to SWAN. It is easy to identify the purpose of most of the methods and data types provided in the code. However, the method bodies are difficult to understand since they lack proper code comments. On a line-by-line basis, it is nearly impossible to discern what needs to be ported to SWAN.

The situation is more complex when considering the architecture of the core *ns-2* code. As stated in Section 4.3, the poor adherence split-level programming model has complicated the implementation and simulation of different networking models. With configuration parameters spread in two locations, it becomes much more difficult to extract the code that is necessary for the simulation to work properly. Additionally, the mix between the interpreted OTcl simulation code and the core C++ code makes it difficult to know if one is looking at network configuration parameters for a simulation or the code that drives the simulation.

The relationship between the IEEE 802.15.4 model and the core *ns-2* code is much like a ball of tangled wires. In order to get the one needed wire out of the mess, it takes a careful process of examining how that wire interacts with all the other wires. The wire then must be picked apart and finally extracted.

4.4.3 Verification and Validation

The IEEE 802.15.4 SWAN implementation must be verified and validated to ensure its correctness. In order to verify the simulation, the code must be properly debugged. This can be accomplished through standard programming procedures such as developing and testing in smaller units, code review, and testing input parameters (Law 2007).

Verification in the form of debugging is not the only requirement. Although the simulation may act in the way the programmer intended, it does not ensure it models the real-world implementation correctly. This goal is accomplished through the process of validation. Validation involves comparing the results to another model, the real-world system, or comparing with an expert opinion (Law 2007).

First, the *ns-2* code should be validated by conducting experiments with actual IEEE 802.15.4 hardware. Two or more wireless devices should form a network and exchange data. Then, the same setup can be configured through the *ns-2* OTcl code. If the results of the real-world experiment and the simulation match closely, the model is valid. If there are significant differences between the experiment and the simulation, the simulation must be modified (Law 2007).

Once the *ns-2* code is validated, the model can be ported to SWAN. In order to validate the IEEE 802.15.4 code, it should first be compared to a known systems (Law 2007). Naturally, the first system it should be compared with is the *ns-2* implementation. By setting the same parameters for the two network simulators, the results of experiments should match. If there are large differences between the two simulation results, it indicates that the port from *ns-2* to SWAN introduced problems with the simulation. Experiments with IEEE 802.15.4 can then be conducted to match the simulation and hardware results.

4.5 Chapter Summary

SWAN is a powerful ad-hoc wireless network simulator, but it can only simulate IEEE 802.11b networks. In order to make SWAN more useful, IEEE 802.15.4 was chosen as another wireless protocol it should simulate given the research interest in wireless sensor networks. Since the *ns-2* network simulator has an existing implementation of IEEE 802.15.4, we decided that it would be best to port that implementation to SWAN. However, this creates a new set of difficulties given the differences in data types between the two simulators, the poor documentation of *ns-2*, and challenge of verification and validation.

In the next chapter, a road map to implement IEEE 802.15.4 in SWAN is presented and final thoughts are given.

Chapter 5

Conclusion and Future Work

This thesis explained and compared the IEEE 802.11 and IEEE 802.15.4 wireless networking standards. It was shown that they resembled each other closely from a simulation perspective by examining their network topologies, MAC layers, and PHY layers. However, there are many small but important differences between the two standards that must be accounted for in the construction of simulation models.

First, WiFi and Zigbee are both technologies that can be used to form unstructured, ad-hoc networks. However, the goals of the networks are different. WiFi networks are optimized for high-bandwidth communication whereas Zigbee networks use low-bandwidth and consume much less power. This makes WiFi suitable for Internet access and file sharing and Zigbee suitable for sensor networks.

Second, their MAC layers are very similar since they both provide contention and contention-free access. Contention access for both standards use CSMA/CA to manage the shared medium and avoid collisions. Contention-free access is made possible by the Point Coordination Function in a WiFi network to grant a device the privilege to transmit. Zigbee uses Guaranteed Time Slots and the PAN Coordinator to schedule times when a single device can transmit. While the MAC layers are similar in many ways, WiFi and Zigbee handle the hidden node problem differently: Zigbee ignores it completely and WiFi uses special control messages to check if the destination node is currently receiving data from another device. All of these properties are very important when building a simulation model.

Third, the PHY layers of WiFi and Zigbee are similar in general principles, but differ in specifications and parameters. Both use Direct Sequence Spread Spectrum modulation to spread the wireless signal across many frequencies. However, the data rates for even the original IEEE 802.11 DSSS PHY (2Mbps) is much higher than for IEEE 802.15.4 (250Kbps). Additionally, WiFi and Zigbee both operate in the unlicensed ISM band of frequencies but use different transmission powers due to the different goals in range of coverage. These physical layer specifications are less important to a simulation model than the MAC layer specification because the propagation of electromagnetic signals are not being simulated. For instance, the details about how a specific type of modulation works are not important because that is not what is being simulated. Instead, the general performance of a modulation technique (for instance, how it affects the bit error rate) is important to a packet-level simulation.

To follow through on the purpose of this thesis, it would be beneficial to obtain working Zigbee devices in order to run experiments and learn from it directly. A device such as the Freescale MC1202 would be suitable for this purpose. Instead of thinking of Zigbee as a foreign and unfamiliar protocol, actual hardware would make it seem within reach. It would also be essential in the validation of the simulation model once it is completed.

In order to create a working simulation model of Zigbee within SWAN, much work is still needed. The best course of action would be to use the existing *ns-2* 802.15.4 simulation model and port it to SWAN. Unfortunately, this presents a new set of challenges to change *ns-2*-based code to a SWAN equivalent.

This thesis can be used as a guide to implement the IEEE 802.15.4 simulation model in SWAN. The comparison of WiFi to Zigbee that this thesis provides will make it easier to implement a Zigbee model with the known reference point of WiFi.

Bibliography

Breslau, L., D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu (2000). Advances in network simulation. *Computer 33*(5), 59–67.

Cicconetti, C., E. Mingozzi, and G. Stea (2006). An integrated framework for enabling effective data collection and statistical analysis with ns-2. In *Proceedings of the 2006 Workshop on ns-2: The IP Network Simulator (SWNS2 '06)*, Pisa, Italy.

Cooklev, T. (2004). *Wireless Communication Standards*. IEEE Press.

Fall, K. and K. Varadhan (2009). *The ns Manual*.

Feeney, L. M. (2001). An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks. *Mob. Netw. Appl. 6*(3), 239–249.

Geier, J. (1999). *Wireless LANs: Implementing Interoperable Networks*. Macmillan Technical Publishing.

Gutierrez, J. A., E. H. Callaway Jr., and R. L. Barrett Jr. (2007). *Low-Rate Wireless Personal Area Networks*. IEEE Press.

Harthikote-Matha, M., T. Banka, and A. P. Jayasumana (2007). Performance degradation of IEEE 802.15.4 slotted CSMA/CA due to hidden nodes. In *Proceedings of the 32nd IEEE Conference on Local Computer Networks (LCN '07)*, Washington, DC, USA, pp. 264–266. IEEE Computer Society.

Hwang, L.-J., S.-T. Sheu, Y.-Y. Shih, and Y.-C. Cheng (2005). Grouping strategy for solving hidden node problem in IEEE 802.15.4 LR-WPAN. In *WICON '05: Proceedings of the First International Conference on Wireless Internet*, Washington, DC, USA, pp. 26–32. IEEE Computer Society.

IEEE 802.11 (2007). IEEE 802.11 Standard.

IEEE 802.15.4 (2006). IEEE 802.15.4 Standard.

Law, A. M. (2007). *Simulation and Modeling Analysis* (4rd ed.). McGraw-Hill.

Liu, J., F. L. Perrone, D. M. Nicol, C. Elliott, and D. Pearson (2001). Simulation modeling of large-scale ad-hoc sensor networks. In *Proceedings of the European Simulation Interoperability Workshop 2001 (EURO-SIW 2001)*, London, England.

Mura, M., M. Paolieri, and F. Fabbri (2007). Power modeling and power analysis for ieee 802.15.4: a concurrent state machine approach. In *Consumer Communications and Networking Conference (CCNC 2007)*, Las Vegas, NV, U.S.A., pp. 660–664.

O'Hara, B. and A. Petrick (2005). *IEEE 802.11 Handbook*. IEEE Press.

Perrone, F. (2006). SWAN: A simulator for wireless ad hoc networks. Invited Talk at Università di Bologna, Italy; http://www.eg.bucknell.edu/~perrone/research/UniBo-2.ppt [Accessed May 1, 2009].

Perrone, F. (2009). *SWAN User's Manual*.

Perrone, F. L. and S. C. Nelson (2006). A study of on-off attack models for wireless ad hoc networks. In *First IEEE International Workshop on Operator-Assisted (Wireless Mesh) Community Networks (OpComm 2006)*, Berlin, Germany.

Perrone, L. F., C. Cicconetti, G. Stea, and B. C. Ward (2009). On the automation of computer network simulators. In *Proceedings of the* 2^{nd} *International Conference on Simulation Tools and Techniques (SIMUTools 2009)*, Rome, Italy.

Rappaport, T. (1996). *Wireless Communications: Principles and Practice*. Englewood Cliffs, NJ: Prentice Hall.

Tanenbaum, A. S. (2003). *Computer Networks* (4^{rd} ed.). Prentice Hall.

Zheng, J. (2004). 802.15.4 and Zigbee routing simulation at Samsung/CUNY. http://www-ee.ccny.cuny.edu/zheng/pub/file/WPAN_ZBR_pub.pdf [Accessed May 1st, 2009].

Zheng, J. and M. J. Lee (2004). WPAN *ns-2* Simulation Model FAQs. http://www-ee.ccny.cuny.edu/zheng/pub/Pages/wpan-ns2-faq.htm [Accessed May 1st, 2009].