

**Study of ZigBee Wireless Networking and
Simulation Using OMNeT++ Simulator**

By

Shabir Ahmad (102305)

Ibrahim Wonge Lisheshar (102311)

Supervised by

Muhammad Rezaul Hoque Khan

Assistant Proffesor

Dep. Of Electrical and Electronic Engineering (EEE)

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



Islamic University of Technology (IUT)

The Organization of Islamic Cooperation (OIC)

Gazipur-1704, Dhaka, Bangladesh

October 2013



Islamic University of Technology (IUT)
Organization of Islamic Cooperation (OIC)
Gazipur-1704, Dhaka, Bangladesh

Study of ZigBee Wireless Networking and Simulation Using OMNeT++ Simulator

This Thesis is submitted to the Department of Electrical and Electronic Engineering, in partial fulfillment of the requirements for the degree of **Higher Diploma in Electrical and Electronic Engineering (HD-EE)**

30th October 2013

Declaration

This is to certify the project titled “**Study of ZigBee Wireless Networking and Simulation Using OMNeT++ Simulator**” is supervised by Md. Razaul Hoque Khan. This project work has not been submitted anywhere for a degree or diploma.

Approved by

Md. Razaul Hoque, Assistant Professor,

Project supervisor

Dept. of Electrical and Electronic Engineering (EEE)

.....

Date.....

Signature of Head of the Department

.....

Prof. Dr. Md. Shahid Ullah

Head

Date

Ibrahim Wonge Lisheshar (102311).....

Shabir Ahmad (102305).....

Acknowledgement

Foremost we would like to express our deep gratitude to our supervisor, Muhammad Rezaul Hoque khan, for his encouragement to do this work and his kind support during our stay in Bangladesh. The valuable discussions with him gave us many important impulses that contributed considerably to both our scientific and personal development. We are also indebted to all the teachers and staff members of the department of Electrical and Electronic Engineering (EEE) Islamic University of Technology Dhaka, Bangladesh. Finally, many sincere thanks to all those who assisted us in connection with this work.

Abstract

Wireless sensor networks have overseen a very rapid growth over the past few years. As the usage increases with time, the demand for efficiency and lower cost increases as well. ZigBee standard tries to meet these increasing demands by defining a low data rate, low cost and low power (battery powered) wireless network and control, with better sensitivity and larger coverage.

This document gives an overview of the IEEE 802.15.4 standard and explains the various layers of the ZigBee network protocol. It also explains how a simulation based experimentation of the ZigBee network is conducted with the use of OMNeT++ simulator.

Contents

Declaration	iii
Acknowledgement.....	iv
Abstract	v
Contents	vi
Dedication.....	viii
Acronyms.....	ix
Chapter 1	- 1 -
Introduction to ZigBee Technology	- 1 -
1.1: Introduction	- 1 -
Chapter 2	- 2 -
The IEEE Standard	- 2 -
2.1 IEEE 802.15.4 Specification	- 2 -
2.2 Scope of 802.15.4	- 2 -
2.3 PHY Layer	- 2 -
2.4 MAC Layer	- 5 -
2.5 Properties of 802.15.4	- 11 -
2.6 Network Topologies	- 12 -
Chapter 3	- 14 -
ZigBee	- 14 -
3.1: ZigBee Overview	- 14 -
3.1 Interoperability	- 14 -
3.2 Device Types	- 14 -
3.3 Device Roles	- 14 -
3.4 ZigBee versus Bluetooth and IEEE 802.11	- 15 -
3.5 ZigBee Networking Topologies	- 16 -
3.6 ZigBee Network layers	- 17 -
Chapter 4	- 24 -
ZigBee Applications	- 24 -
4.1 Security Systems	- 24 -
4.2 Meter-Reading Systems	- 24 -
4.3 Irrigation Systems	- 24 -
4.4 Light Control Systems	- 24 -
4.5 Consumer Electronics: Remote Control	- 24 -
4.6 Industrial Automation	- 25 -

4.7 Healthcare	- 25 -
Chapter 5	- 26 -
Simulation with OMNeT++	- 26 -
5.1 Introductions	- 26 -
5.2 Installing OMNeT++	- 27 -
5.3 OMNeT++'s Simulation Frameworks	- 28 -
5.4 Simulation	- 33 -
5.5 Results and Interpretation	- 34 -
Chapter 6	- 41 -
Conclusion and future work	- 41 -
6.1 Conclusions	- 41 -
6.2 Future work	- 41 -
References	- 42 -

Dedication

We would like to dedicate this work to our beloved parents.

Acronyms

Abbreviation	Phrase
AF	Application Framework
APDU	Application Support Sub-layer Protocol Data Unit
APL	Application Layer
APS	Application Support Sub-layer
APSD	Application Support Sub-layer Data Entity
APSD-SAP	APSD-Service Access Point
APSM	Application Support Sub-layer Management Entity
APSM-SAP	APSM-Service Access Point
ASDU	APS Service Data Unit
BPSK	Binary Phase-Shift Keying
CAP	Contention Access Period
CCA	Clear Channel Assessment
CFP	Contention-Free Period
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
DSSS	Direct Sequence Spread Spectrum
ED	Energy Detection
FFD	Full-Function Device
GTS	Guaranteed Time Slot
IFS	Interframe Spacing
ISM	Industrial, Scientific, and Medical
LQI	Link Quality Indicator
LR-WPAN	Low-Rate Wireless Personal Area Network
MLME	MAC Layer Management Entity
MLME-SAP	MAC Layer Management Entity Service Access Point
MSDU	MAC Protocol Data Unit
MSD	MAC Service Data
NLDE	Network Layer Data Entity
NLDE-SAP	Network Layer Data Entity Service Access Point
NLME	Network Layer Management Entity
NLME-SAP	Network Layer Management Entity Service Access Point
NPDU	Network Layer Protocol Data Unit
NSDU	Network Service Data Unit
NWK	Network Layer
PAN	Personal Area Network
PD	PHY Data
PD-SAP	PHY Data Service Access Point
PER	Packet Error Rate
PHR	PHY Header
PHY	Physical Layer
PLME	Physical Layer Management Entity

PLME-SAP	Physical Layer Management Entity Service Access Point
POS	Personal Operating Space
PPDU	PHY Protocol Data Unit
PSD	Power Spectral Density
PSDU	PHY Service Data Unit
RF	Radio Frequency
RFD	Reduced Function Device
SAP	Service Access Point

Introduction to ZigBee Technology

1.1: Introduction

ZigBee standard is managed by the ZigBee Alliance, a global consortium of more than 50 companies (IC vendors & tech companies).

ZigBee is a home-area network designed specifically to replace the proliferation of individual remote controls. ZigBee was created to satisfy the market need for a cost-efficient, standards-based wireless network that supports low data rates, low power consumption, security, and reliability. To address this need, the ZigBee Alliance, an industry working group is developing standardized application software on top of the IEEE 802.15.4 wireless standard. The alliance is working closely with the IEEE to ensure an integrated, complete, and interoperable network for the market. For example, the working group will provide interoperability certification testing of 802.15.4 systems that include the ZigBee software layer. The ZigBee Alliance will also serve as the official test and certification group for ZigBee devices. ZigBee is the only standards-based technology that addresses the needs of most remote monitoring, control and sensory network applications. Following the standard Open Systems Interconnection (OSI) reference model, ZigBee protocol stack is structured in layers. The first two layers, physical (PHY) and media access (MAC), are defined by IEEE 802.15.4 standard. The layers above them are defined by ZigBee Alliance.

The IEEE Standard

2.1 IEEE 802.15.4 Specification

802.15.4, defines a standard for a low-rate WPAN (LR-WPAN). This standard uses radio communication in the personal area networks (PAN) to carry out the communication between the devices and also define protocols for establishing connection between them. It uses channel access mechanisms like Slotted CSMA/CA [1] which helps avoid the collisions. PAN coordinator can reserve time slots for the devices using the super frame structure at the media access layer. This standard deals with two PHYs, one is 868/915 MHz PHY and the other is 2450 MHz PHY, both uses the direct sequence spread spectrum (DSSS) modulation scheme. The data rates supported are 20 kb/s and 40 kb/s (for 868/915 MHz PHY) and 250 kb/s for 2450 MHz PHY. This low rate wireless personal area networks (IEEE 802.15.4/ LR-WPAN) standard will provide the solutions for different applications at low power and low cost. This standard describes the two lower layers Physical and MAC of the ZigBee Stack.

2.2 Scope of 802.15.4

802.15.4, is a packet-based radio protocol. It addresses the communication needs of wireless applications that have low data rates and low power consumption requirements. It is the foundation on which ZigBee is built.

2.3 PHY Layer

The PHY layer defines the physical and electrical characteristics of the network. The basic task of the PHY layer is data transmission and reception. At the physical/electrical level, this involves modulation and spreading techniques that map bits of information in such a way as to allow them to travel through the air. Specifications for receiver sensitivity and transmit output power are in the PHY layer.

The PHY layer is also responsible for the following tasks:

- enable/disable the radio transceiver
- Link quality indication (LQI) for received packets
- Energy detection (ED) within the current channel
- Clear channel assessment (CCA)

2.3.1 PHY Data Service

The data that needs to be transmitted is always provided as a MAC Protocol Data Unit (MPDU). The local MAC generates the request for transmission and provides the MPDU. The PHY attempts the transmission and

reports the result of the attempt (successful or unsuccessful) to the MAC. The reasons for an unsuccessful transmission attempt can be any of the following:

- Radio transceiver is disabled.
- Radio transceiver is in receive mode. Radio cannot transmit and receive simultaneously.
- Radio transceiver is busy (already engaged in transmitting).

When the data is received by the radio transceiver, the PHY layer notifies the MAC layer of the arrival of an MPDU. The PHY layer not only provides the MPDU to the MAC layer, it also delivers the LQI information.

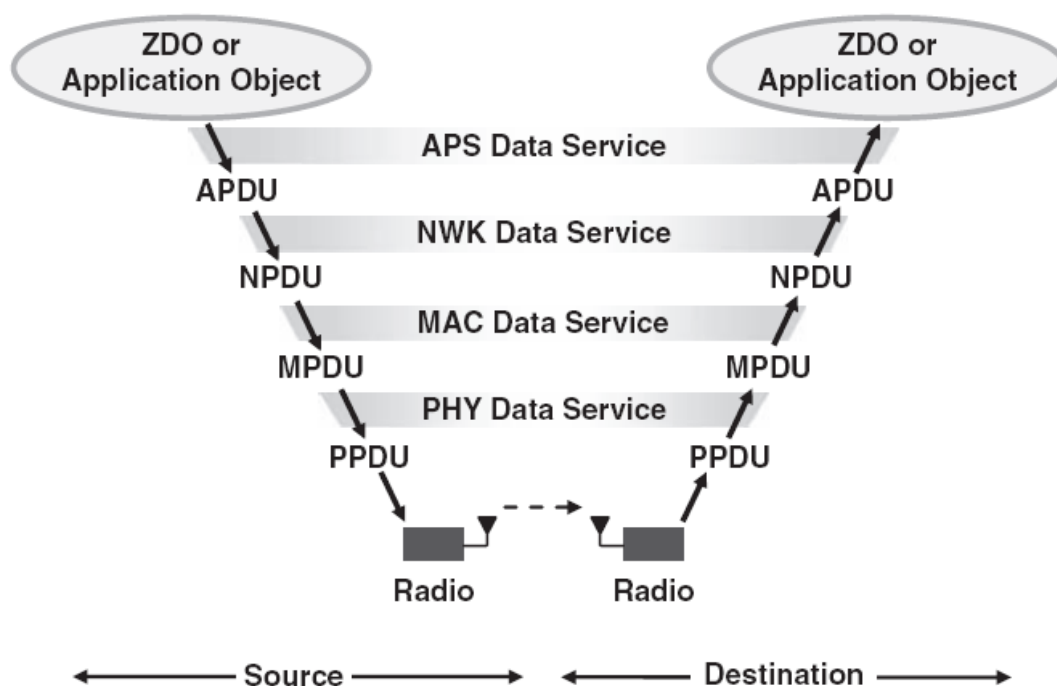


Figure 2.1: Data Transfer Service Between Two Devices

The data does not always come from the application layer. The data, for example, can be generated by the MAC layer without engaging the next higher layer. In the scenario shown in Figure 2.1, the data is provided by either the ZigBee Device Objects (ZDO) or an application object to the Application Support sub-layer (APS). In the transmitting device, each layer adds its own header and footer (if applicable) to the data unit (DU) and then passes the result to the next lower layer. The DU in each layer is known by the name of the layer. In the APS and NWK layers, the DU is called the APS Protocol Data Unit (APDU) and the NWK Protocol Data Unit (NPDU), respectively. The PHY data service receives a MAC Protocol Data Unit (MPDU) and creates the PHY Protocol Data Unit (PPDU) that will be transmitted by the radio. On the receiver side, the data is

passed upward from one layer to the next higher layer and the header and footers are removed until the DU reaches the intended layer at the destination.

2.3.2 Clear Channel Assessment (CCA)

The MLME requests that the PLME perform a CCA whenever the CSMA-CA requires an assessment of the channel. The result of a CCA can be any of the following:

- The transceiver is disabled and therefore no CCA is performed.
- The channel is available (idle) and can be used for transmission.
- The channel or transceiver is busy:
- The channel is busy (another device is using this frequency channel).
- The transceiver is busy transmitting and therefore no CCA is performed.

The PLME does not distinguish between a busy channel and a busy transceiver and delivers the same busy status to the MLME in both cases.

2.3.3 Energy Detection (ED)

The ED request is generated by the MLME and issued to the PLME. If the ED measurement is completed successfully, the energy level is reported back to the MLME. A disabled radio or a transceiver engaged in transmission will cause the ED request to fail.

2.3.4 Enabling and Disabling the Radio Transceiver

The MLME can request that the PLME put the transceiver in one of three main states: transceiver disabled, transmitter enabled, and receiver enabled.

2.3.5 Obtaining Information from the PHY-PIB

The PLME can read the value of any PHY attribute in the PHY-PIB and provide it to the MLME. The request to read a PHY attribute is always issued by the MLME.

2.3.6 Setting the Value of a PHY-PIB Attribute

The read-only PHY attributes can be changed only by the PHY. However, for all other attributes, the MLME can request that the PLME sets the PHY-PIB attribute to a given value.

2.3.7 PHY Packet Format

The PHY Protocol Data Unit (PPDU) format is shown in Figure 2.2. The PPDU consists of three components: the Synchronization header (SHR), the PHY header (PHR), and the PHY payload. The SHR enables the receiver to synchronize and lock into the bit stream. The PHR contains frame length information. The PHY payload is provided by upper layers and includes data or commands that need to be transmitted to another device. The SHR consists of a preamble and a start-of-frame delimiter (SFD). The preamble field is used by the

receiver to obtain chip and symbol synchronization. The bits in the preamble field in all PHYs, except for the ASK PHYs, are binary zeros [2].

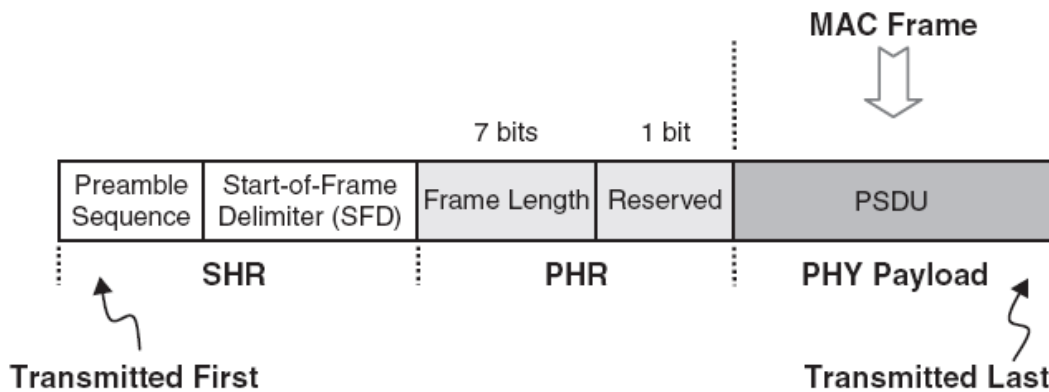


Figure 2.2: PPDU Format

2.4 MAC Layer

The MAC layer defines how multiple 802.15.4 radios operating in the same area will share the airwaves. This includes coordinating transceiver access to the shared radio link and the scheduling and routing of data frames. There are network association and disassociation functions embedded in the MAC layer. These functions support the self-configuration and peer-to-peer communication features of a ZigBee network. The MAC layer is responsible for the following tasks:

- beacon generation if device is a coordinator,
- implementing carrier sense multiple access with collision avoidance (CSMA-CA),
- handling guaranteed time slot (GTS) mechanisms,
- data transfer services for upper layers.

2.4.1 CSMA-CA

The channel access mechanism supported by the IEEE 802.15.4 MAC is called the Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA). In CSMA-CA, whenever a device wants to transmit, it performs a CCA to ensure that the channel is not in use by any other device. Then the device starts transmitting its own signal. This section provides details of the CSMA-CA.

In addition to transmitting beacons, there are two more occasions on which a device accesses the channel without using the CSMA-CA algorithm:

- The channel access during the contention-free period (CFP).

- Transmitting immediately after acknowledging a data request command. In other words, if a device requests data from a coordinator, the coordinator transmits the acknowledgment followed immediately by the data without performing CSMA-CA between these two transmissions, even during the contention access period (CAP).

There are two types of CSMA-CA: slotted and un-slotted. Slotted CSMA-CA is referred to as performing CSMA-CA while there is a superframe structure in place. A superframe divides the active period into 16 equal time slots. The back-off periods of the CSMA-CA algorithm need to be aligned to specific time slots discussed below. The un-slotted CSMA-CA algorithm is used when there is no superframe structure; consequently, no back-off slot alignment is necessary. A nonbeacon-enabled network always uses the un-slotted CSMA-CA algorithm for channel access.

If the CCA indicates a busy channel, the device will back off for a random period of time and then try again. This random back-off period in both slotted and un-slotted CSMA-CA is an integer multiple of the unit back-off period. The unit back-off period is equal to `aUnitBackoffPeriod` (a MAC constant) symbols.

Figure 2.3 is the flowchart of the CSMA-CA algorithm. In the first step of the algorithm, a decision is made to use either slotted or un-slotted CSMA-CA. Three variables are used in the CSMA-CA algorithm: the back-off exponent (BE), the number of back-offs (NB), and the contention window (CW) length.

Every time the algorithm faces a busy channel, it backs off for a random period of time and the BE determines the allowed range for this random period.

This random back-off is any integer number between 0 to $(2^{BE} - 1)$ multiplied by the unit back-off period:

Back-off = (A random integer number between 0 to $(2^{BE} - 1)$ * `aUnitBackoffPeriod`).

The initial value of BE is equal to `macMinBE` in an un-slotted CSMA-CA channel access. In a slotted CSMA-CA, the choice of the battery life extension (BLE) option in the superframe structure affects the value of BE. If the BLE option is active, the coordinator turns off its receiver after a period equal to `macBattLifeExtPeriods` following the transmission of a beacon frame, to conserve energy. In this case, the range for the back off period is limited to the lesser of 2 and the value of `macMinBE`:

$$BE = \min(2, \text{macMinBE})$$

If the BLE option is not selected, the coordinator is active during the entire CAP, and the value of BE is equal to macMinBE (similar to the un-slotted CSMA-CA). The value of BE is incremented every time a CCA is performed and the channel is busy. But the value of BE cannot exceed macMaxBE.

The NB is a counter that keeps track of the number of times the device backs off and retries the channel access mechanism. At the beginning of the algorithm, NB is equal to zero, and every time the device has to back off due to facing a busy channel, BE is incremented once. If NB reaches macMaxCSMABackoffs and still the channel is not accessed successfully, the CSMA-CA algorithm quits and reports channel access failure to the NWK layer.

The contention window (CW) variable determines the number of back-off periods that the channel must be available before starting to transmit. For example, if CW is equal to 2, the device starts transmitting only after two consecutive back-offs resulted in an available (idle) channel. The CW is used only in the slotted CSMA-CA algorithm.

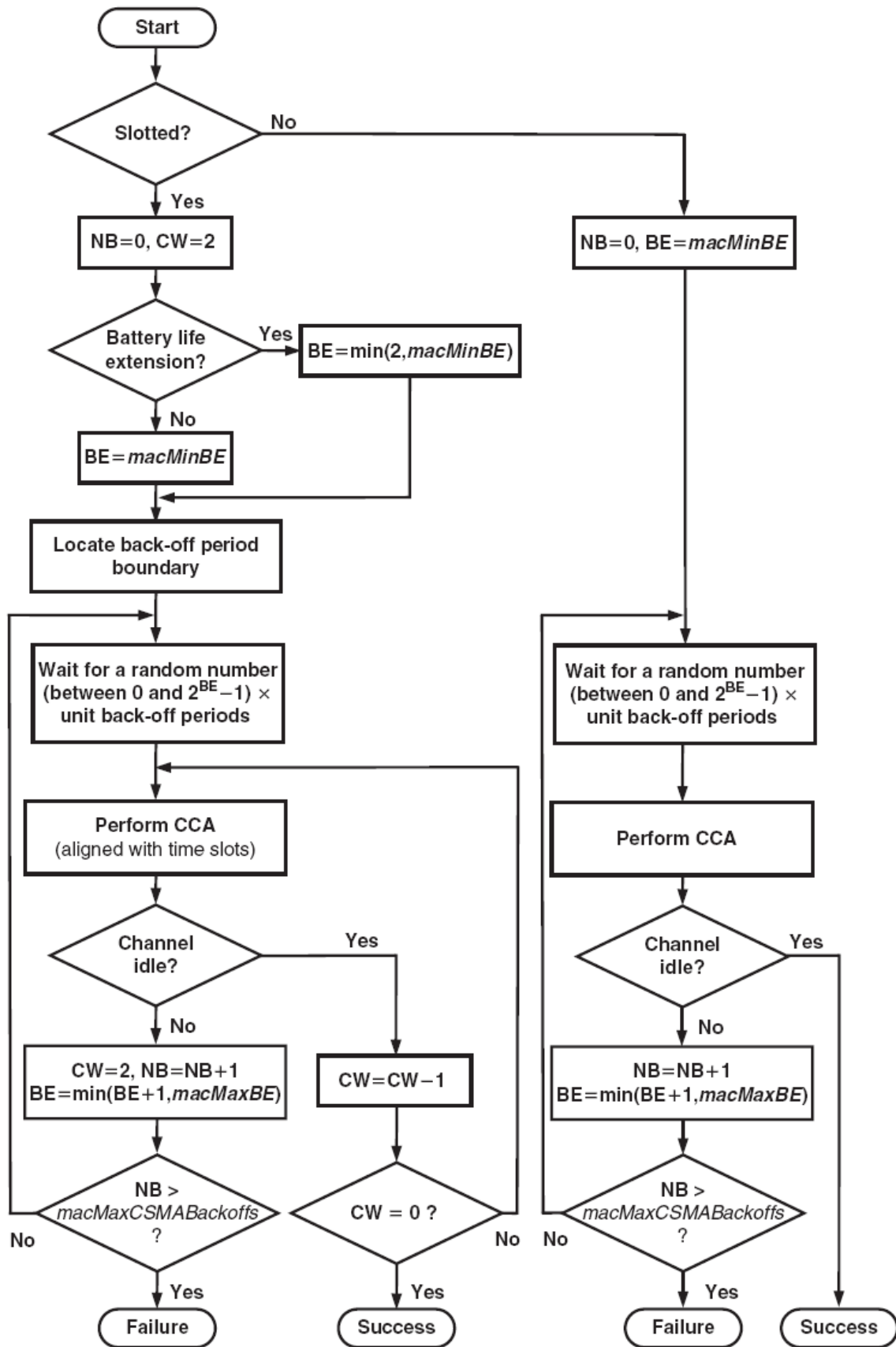


Figure 2.3: The CSMA-CA Algorithm

2.4.2 MAC Frame Structures

The IEEE 802.15.4 defines four MAC frame structures:

- Beacon frame
- Data frame
- Acknowledge frame
- MAC command frame

The beacon frame is used by a coordinator to transmit beacons. The beacons are used to synchronize the clock of all the devices within the same network. The data and acknowledgment frames are used to transmit data and accordingly acknowledge the successful reception of a frame. The MAC commands are transmitted using a MAC command frame.

(i) The Beacon Frame Format

The beacon frame (see Figure2.4) is a special form of the MAC general frame format. The sequence number field contains the current value of macBSN (beacon sequence number). The superframe specification field is part of the MAC payload, and its subfields are shown in Figure2.4. The beacon order (BO) subfield determines the transmission intervals between the beacons.

The superframe is divided into 16 equal time slots, and the final CAP slot subfield determines the last time slot of the CAP. If there are any time slots left after the CAP, they are used as GTSs.

If this beacon frame is transmitted by the PAN coordinator, the PAN coordinator subfield is set to one. This helps distinguish between a frame received from the PAN coordinator and any other coordinator in the same network. The last bit in the superframe specification field is the association permit. If the association permit is set to zero, it means that the coordinator does not accept association requests at this moment.

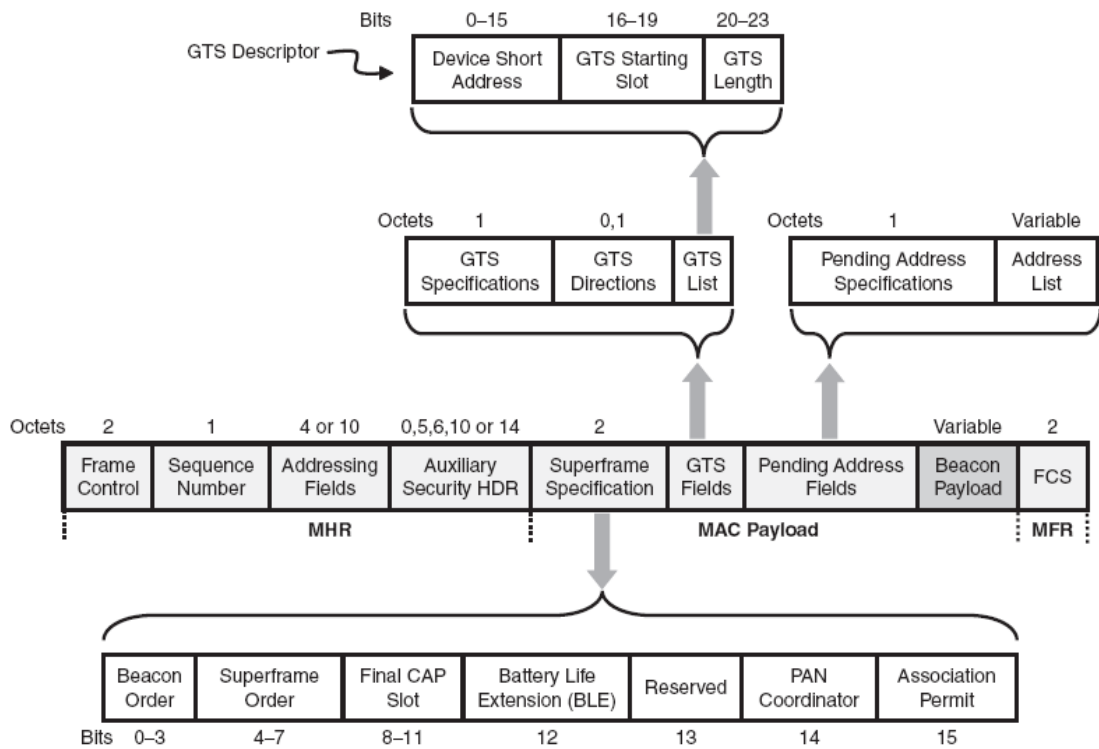


Figure 2.4: Beacon Frame Format and its Subfields

The beacon frames are used to establish GTSs. The GTS specification subfield clarifies whether the PAN coordinator currently accepts GTS requests. It also determines the number of GTSs listed in the GTS list field. The GTS direction subfield can be used to set the direction of the GTSs to receive-only or transmit-only modes. The GTS list is the list of all GTSs that are currently maintained (GTS descriptors). Each GTS descriptor contains the short address of the device that will use the GTS, the GTS starting slot, and the GTS length.

The pending address field contains the addresses of all the devices that have data pending at the coordinator. Each device checks for its address in this field and if there is a match, the device will contact the coordinator and request the data be transmitted to it.

(ii) The Data Frame

The data payload is provided by the NWK layer. The data in the MAC payload is referred to as the MAC Service Data Unit (MSDU). The fields in this frame are similar to the beacon frame except the super-frame, GTS, and pending address fields are not present in the MAC data frame. The MAC data frame is referred to as the MAC Protocol Data Unit (MPDU) and becomes the PHY payload.

(iii) **The Acknowledgment Frame**

The MAC acknowledgment frame is the simplest MAC frame format and does not carry any MAC payload. The acknowledgment frame is sent by one device to another to confirm successful reception of a packet.

(iv) **The Command Frame**

The MAC commands such as requesting association or disassociation with a network are transmitted using the MAC command frame. The command type field determines the type of the command (e.g. association request or data request). The command payload contains the command itself.

The entire MAC command frame is placed in the PHY payload as a PSDU.

2.4.3 The Interframe Spacing

In transmitting data from one device to another, the transmitting device must wait briefly between its successive transmitted frames to allow the recipient device process a received frame before the next frame arrives. This is known as interframe spacing (IFS). The length of IFS depends on the transmitted frame size. The MPDUs with sizes of less than or equal to `aMaxSIFSFramesSize` (a MAC constant with default value of 18 octets) are considered short frames. A long frame is an MPDU with a size that exceeds `aMaxSIFSFramesSize` octets. The waiting period after a short frame is referred to as short IFS (SIFS). The minimum value of SIFS is `macMinSIFSPeriod`. Similarly, a long frame is followed by a long IFS (LIFS) with minimum length of `macMinLIFSPeriod`. The values of `macMinSIFSPeriod` and `macMinLIFSPeriod` are 12 and 40 symbols, correspondingly.

2.5 Properties of 802.15.4

802.15.4, defines operation in three license-free industrial scientific medical (ISM) frequency bands. Below is a table that summarizes the properties of IEEE 802.15.4 in two of the ISM frequency bands: 915 MHz and 2.4 GHz.

Comparison of IEEE 802.15.4 Frequency Bands

Property Description	Prescribed Values	
	915 MHz	2.4 GHz
Raw data bit rate	40 kbps	250 kbps
Transmitter output power	1 mW = 0 dBm	
Receiver sensitivity (<1% packet error rate)	-92 dBm	-85 dBm
Transmission range	Indoors: up to 30 m; Outdoors: up to 100 m	
Latency	15 ms	
Channels	10 channels	16 channels
Channel numbering	1 to 10	11 to 26
Channel access	CSMA-CA and slotted CSMA-CA	
Modulation scheme	BPSK	O-QPSK

2.6 Network Topologies

According to the IEEE 802.15.4 specification, the LR-WPAN may operate in one of two network topologies: star or peer-to-peer. 802.15.4, is designed for networks with low data rates, which is why the acronym “LR” (for “low rate”) is prepended to “WPAN.”

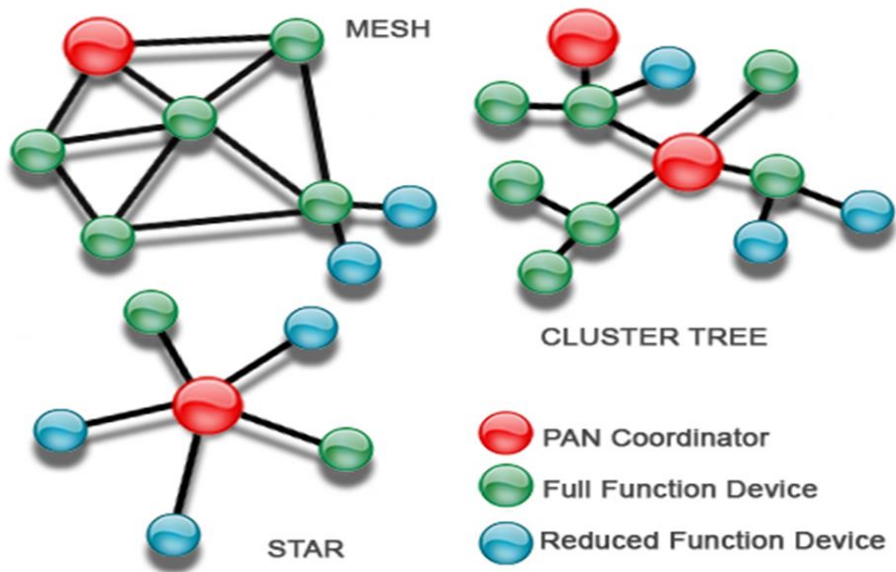


Figure 2.6: Network Topologies

ZigBee

3.1: ZigBee Overview

ZigBee is a standard that defines a set of communication protocols for low-data-rate, short range wireless networking. ZigBee-based wireless devices operate in 868 MHz, 915 MHz, and 2.4 GHz frequency bands. The maximum data rate is 250 K bits per second.

ZigBee is targeted mainly for battery-powered applications where low data rate, low cost, and long battery life are main requirements. In many ZigBee applications, the total time the wireless device is engaged in any type of activity is very limited; the device spends most of its time in a power-saving mode, also known as sleep mode. As a result, ZigBee enabled devices are capable of being operational for several years before their batteries need to be replaced. In many ZigBee applications, the devices have duty cycles of less than 1% to ensure longer years of battery life.

3.1 Interoperability

ZigBee has a wide range of applications; therefore, several manufacturers provide ZigBee-enabled solutions. It is important for these ZigBee-based devices be able to interact with each other regardless of the manufacturing origin. In other words, the devices should be interoperable. Interoperability is one of the key advantages of the ZigBee protocol stack. ZigBee-based devices are interoperable even when the messages are encrypted for security reasons.

3.2 Device Types

There are two types of devices in an IEEE 802.15.4 wireless network: full-function devices (FFDs) and reduced-function devices (RFDs). An FFD is capable of performing all the duties described in the IEEE 802.15.4 standard and can accept any role in the network. An RFD, on the other hand, has limited capabilities. For example, an FFD can communicate with any other device in a network, but an RFD can talk only with an FFD device. RFD devices are intended for very simple applications such as turning on or off a switch. The processing power and memory size of RFD devices are normally less than those of FFD devices.

3.3 Device Roles

In an IEEE 802.15.4 network, an FFD device can take three different roles: coordinator, PAN coordinator, and device. A coordinator is an FFD device that is capable of relaying messages. If the coordinator is also the principal controller of a personal area network (PAN), it is called a PAN coordinator. If a device is not acting as a coordinator, it is simply called a device. The ZigBee standard uses slightly different terminology. A ZigBee coordinator is an IEEE

802.15.4 PAN coordinator. A ZigBee router is a device that can act as an IEEE 802.15.4 coordinator. Finally, a ZigBee end device is a device that is neither a coordinator nor a router. A ZigBee end device has the least memory size and fewest processing capabilities and features. An end device is normally the least expensive device in the network.

3.4 ZigBee versus Bluetooth and IEEE 802.11

Comparing the ZigBee standard with Bluetooth and IEEE 802.11 WLAN helps us understand how ZigBee differentiates itself from existing established standards. Figure 3.1 summarizes the basic characteristics of these three standards.

IEEE 802.11 is a family of standards; IEEE 802.11b is selected here because it operates in 2.4 GHz band, which is common with Bluetooth and ZigBee. IEEE 802.11b has a high data rate (up to 11 Mbps), and providing a wireless Internet connection is one of its typical applications. The indoor range of IEEE 802.11b is typically between 30 and 100 meters. Bluetooth, on the other hand, has a lower data rate (less than 3 Mbps) and its indoor range is typically 2 to 10 meters. One popular application of Bluetooth is in wireless headsets, where Bluetooth provides the means for communication between a mobile phone and a hands-free headset. ZigBee has the lowest data rate and complexity among these three standards and provides significantly longer battery life.

ZigBee's very low data rate means that it is not the best choice for implementing a wireless Internet connection or a CD-quality wireless headset where more than 1Mbps is desired. However, if the goal of wireless communication is to transmit and receive simple commands and/or gather information from sensors such as temperature humidity sensors, ZigBee provides the most power and the most cost-efficient solution compared to Bluetooth and IEEE 802.11b.

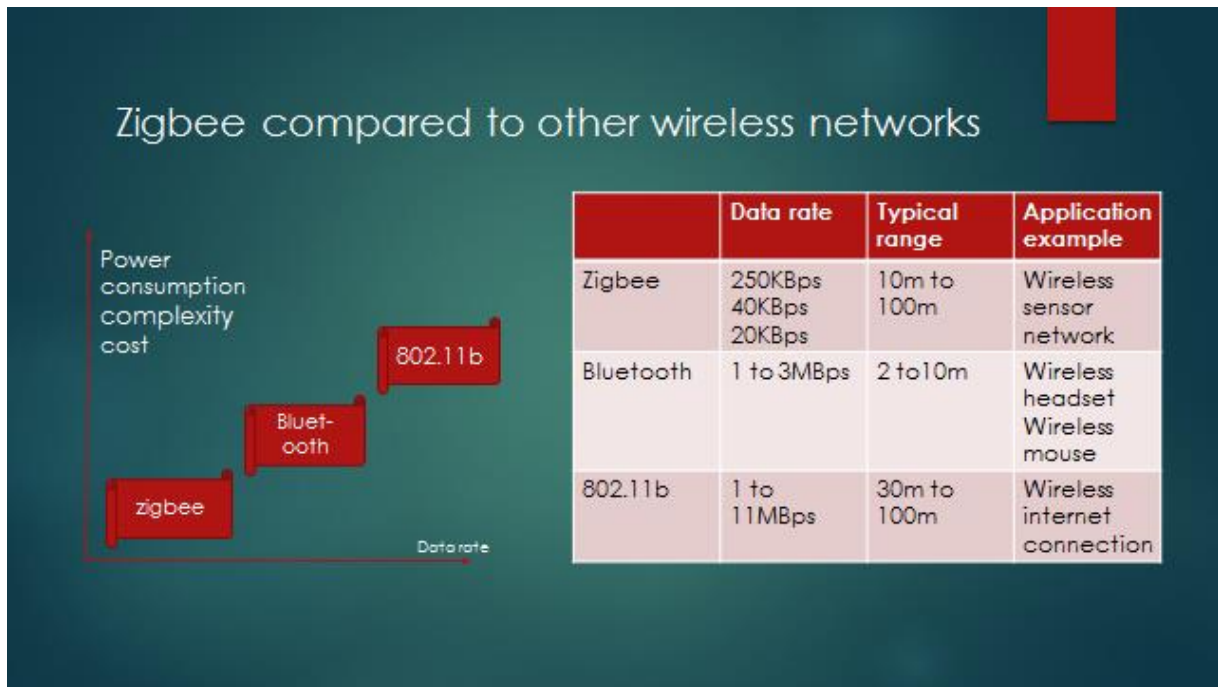


Figure 3.1: ZigBee compared to other wireless networks

3.5 ZigBee Networking Topologies

The network formation is managed by the ZigBee networking layer. The network must be in one of two networking topologies specified in IEEE 802.15.4: star and peer-to-peer.

In the star topology, shown in Figure 3.2, every device in the network can communicate only with the PAN coordinator. A typical scenario in a star network formation is that an FFD, programmed to be a PAN coordinator, is activated and starts establishing its network. The first thing this PAN coordinator does is select a unique PAN identifier that is not used by any other network in its radio sphere of influence, the region around the device in which its radio can successfully communicate with other radios. In other words, it ensures that the PAN identifier is not used by any other nearby network.

In a peer-to-peer topology, each device can communicate directly with any other device if the devices are placed close enough together to establish a successful communication link. Any FFD in a peer-to-peer network can play the role of the PAN coordinator. One way to decide which device will be the PAN coordinator is to pick the first FFD device that starts communicating as the PAN coordinator. In a peer-to-peer network, all the devices that participate in relaying the messages are FFDs because RFDs are not capable of relaying the messages. However, an RFD can be part of the network and communicate only with one particular device (a coordinator or a router) in the network.

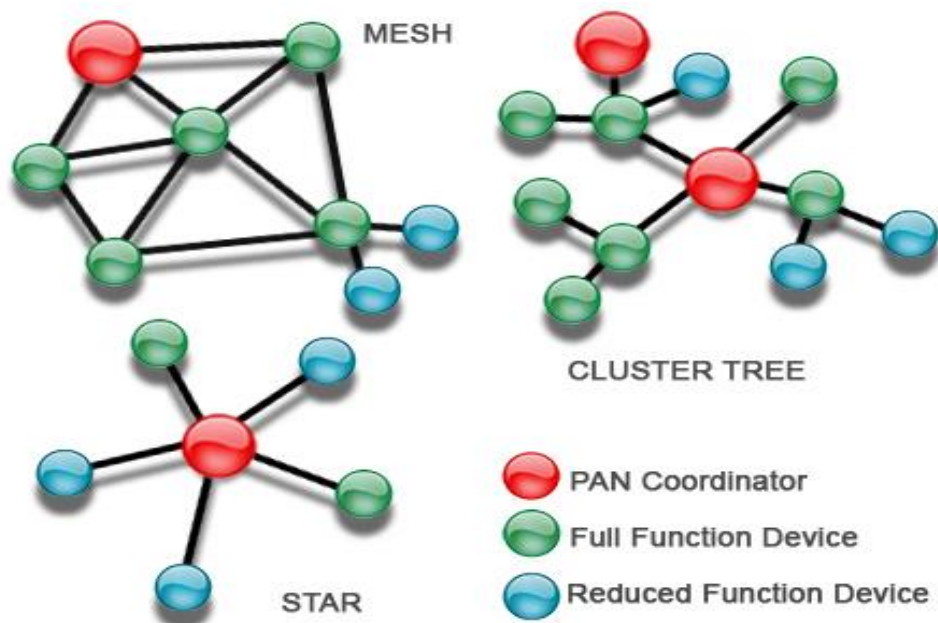


Figure 3.2 ZigBee Topologies

3.6 ZigBee Network layers

The ZigBee standard specifies a number of different layers. The ZigBee layers are discussed in detail in this section and include the Phy layer, Mac layer, Network layer and the Application layer.

3.6.1 Physical layer

PHY layer provides two services:

- PHY data service, which enables the transmission and reception of PHY protocol data units (PPDUs) across the radio channel.
- PHY management service

The PHY is responsible for the following tasks:

- Activation and deactivation of the radio transceiver,
- Energy detection (ED) within the current channel,
- Link quality indicator (LQI) for received packets, that provides information for application about quality of wireless link,
- Clear channel assessment (CCA) for carrier sense multiple access with collision avoidance (CSMA-CA),
- Channel frequency selection,
- Data transmission and reception.

It's worthy of note that energy detection (ED) is used not only to select the best channel during network initialization, but also it provides possibility to adapt to

a changing RF environment by selecting another channel, if a link quality in the current one is causing transmission.

3.6.2 MAC layer

The MAC layer is responsible for

- data reception and transmission scheduling,
- data validity/integrity checking,
- acknowledgment of frame delivery,
- node addressing,
- time synchronization,
- association and disassociation of nodes,
- CSMA/CA multiple access,
- handling of so called guaranteed time slots (GTMs).

The IEEE 802.15.4 MAC layer provides interface for higher layer by defining two types of services: MAC data service and MAC management service. It also provides hooks that can be used by security mechanisms. MAC data service enables the transmission and reception of MAC Protocol Data Units (MPDUs) across the PHY data service. The name of the interface to higher layer is MLDE-SAP.

MAC management service is responsible for invoking the layer management function and it maintains a database of managed object associated with the MACsub-layer.

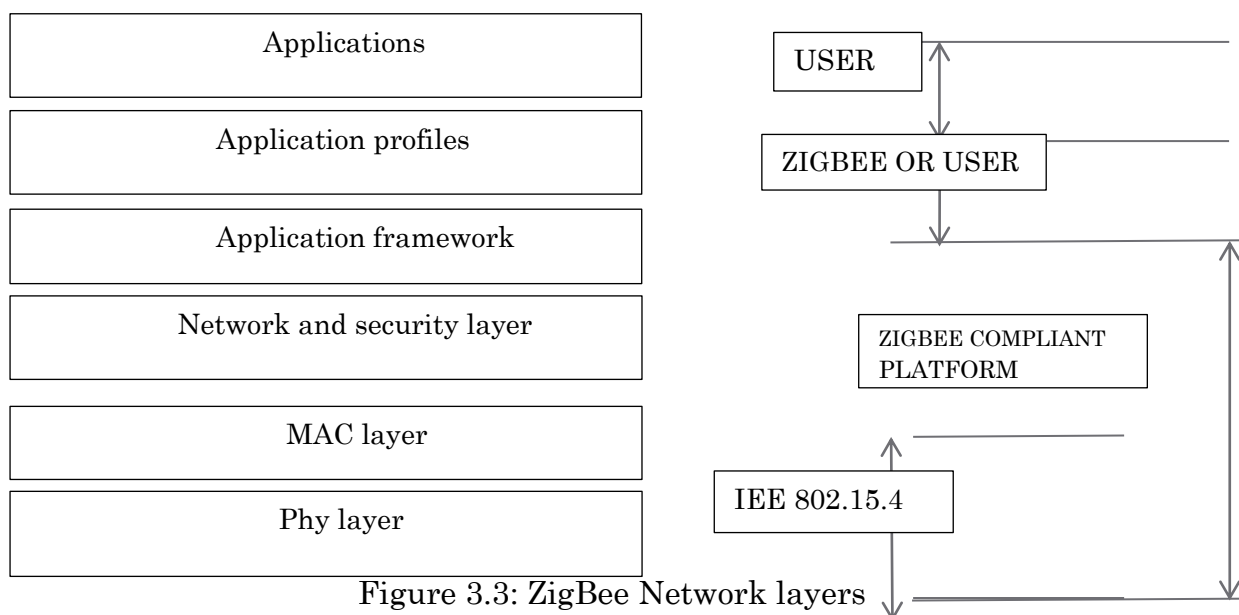


Figure 3.3: ZigBee Network layers

3.6.3 Network (NWK) layer of the ZigBee

Network layer controls IEEE 802.15.4 MAC sub-layer and provides an interface to the application layer. To accomplish this, two service entities are used. The first one, the NWK layer data entity (NLDE) provides the data transmission service via the NLDE-SAP. The NWK layer management entity (NLME) is used to manage services via the NLME-SAP.

Network layer is responsible for the following functions:

- joining and leaving the network by ZigBee devices,
- providing cryptographic security for transmitted frames,
- routing frames to their destination,
- discovering and maintain route tables,
- discovering one-hop neighbors,
- storing of important information about neighbor devices.

Three types of devices are used in the network: a coordinator, router and end-device. A coordinator is responsible for starting a network and assigning addresses to newly associated devices. Router routes frames and is likely to be constantly powered on, in contrast to end-devices.

Coordinator and router are Full Function Devices (FFD) and end-devices are Reduced Function Devices (RFD) as described in IEEE 802.15.4 standard.

When a coordinator attempts to establish a ZigBee network, it does an energy scan to find the best RF channel for its new network. When a channel has been chosen, the coordinator assigns the logical network identifier, also known as the PAN ID, which will be applied to all devices that join the network. A node can join the network either directly or through association. To join directly, the system designer must somehow add a node's extended address into the neighbor table of a device. The direct joining device will issue an orphan scan, and the node with the matching extended address (in its neighbor table) will respond, allowing this device to join.

3.6.4 Application layer of ZigBee

Application layer consists of Application Framework, ZigBee Device Object (ZDO) and Application Support Sub-layer (APS).

Application Framework (AF) provides the functions, data types and data-frame formats for transporting data to facilitate the profile building process. It carries the environment in which application objects are built into ZigBee devices. Application Object is responsible for initiation of standard network functions as well as controlling and managing of the protocol layers. AF communicates with

APS via APSDE-SAP, which includes primitives for data transfer such as: request, response, indication and confirmation. Transfer is carried out between peer app object entities. Up to 240 distinct Application Objects could be defined. There are also two special endpoints; to interface data to the ZDO and to interface data function to broadcast data to all app objects

ZigBee Device Object (ZDO) is responsible for:

- establishing a secure high-level connection between ZigBee devices,
- discovering devices and their services in the network,
- determining whether the device is coordinator, end device or router,
- initiating a binding requests or replying to them,
- providing the interface to the lower portions of the ZigBee protocol stack via End Point 0 (EP0).

Finally the APS sub-layer provides:

- messages forwarding between bound devices,
- removal and filtering of messages,
- reliable data transport,
- mapping 64 bit IEEE addresses to 16 bit NWK addresses,
- maintaining tables for binding.

The application framework in ZigBee is the environment in which application objects are hosted to control and manage the protocol layers in a ZigBee device. Application objects are developed by manufacturers, and that is where a device is customized for various applications. There can be up to 240 application objects in a single device. The application objects use APSDE-SAP to send and receive data between peer application objects. Each application object has a unique endpoint address (endpoint 1 to endpoint 240). The endpoint address of zero is used for the ZDO. To broadcast a message to all application objects, the endpoint address is set to 255. Endpoint addressing allows multiple devices to share the same radio. In the light control, multiple lights were connected to a single radio. Each light has a unique endpoint address and can be turned on and off independently.

The ZigBee Device Object (ZDO) provide an interface between the APS sub-layer and the application framework. The ZDO contains the functionalities that are common in all applications operating on a ZigBee protocol stack. For example, it is the responsibility of the ZDO to configure the device in one of three possible logical types of ZigBee coordinator, ZigBee router, or ZigBee end device. The ZDO uses primitives to perform its duties and accesses the APS sub-layer

Management Entity via APSME-SAP. The application framework interacts with the ZDO through the ZDO public interface.

3.6.4.1 The Application Framework

The ZigBee standard offers the option to use application profiles in developing an application. The use of an application profile allows further interoperability between the products developed by different vendors for a specific application. For instance, in a light control scenario, if two vendors use the same application profile to develop their products, the switches from one vendor will be able to turn on and turn off the lights manufactured by the other vendor. The application profiles are also referred to as ZigBee profiles.

Each application profile is identified by a 16-bit value known as a profile identifier. Only the ZigBee alliance can issue profile identifiers. A vendor that has developed a profile can request a profile identifier from the ZigBee alliance. The ZigBee alliance evaluates the proposed application profile and if it meets the alliance guidelines, a profile identifier will be issued. The application profiles are named after their corresponding application use.

For example, the home automation application profile provides a common platform for vendors developing ZigBee-based products for home automation use.

The general structure of an application profile is shown in Figure 3.4. The application profile consists of two main components: clusters and device descriptions. A cluster is a set of attributes grouped together. Each cluster is identified by a unique 16-bit number called a cluster identifier. Each attribute in a cluster is also identified by a unique 16-bit number known as an attribute identifier. These attributes are used to store data or state values. For example, in a temperature control application, a device that acts as the temperature sensor can store the value of the current temperature in an attribute. Then another device that acts as the furnace controller can receive the value of this attribute and turn on or turn off the furnace accordingly. The application profile does not contain the cluster itself. Instead, the application profile has a list of the cluster identifiers. Each cluster identifier uniquely points to the cluster itself.

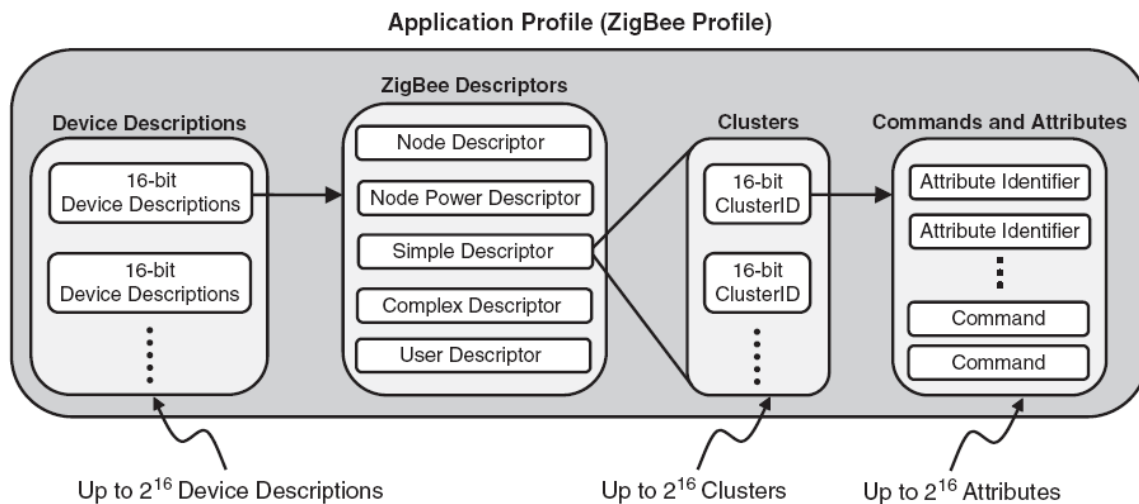


Figure 3.4: The Application Profile

The other part of an application profile is the device descriptions (Figure 3.4). The descriptions provide information regarding the device itself. For example, the supported frequency bands of operation, the logical type of the device (coordinator, router, or end device), and the remaining energy of the battery are provided by the device descriptions. Each device description is identified by a 16-bit value. The ZigBee application profile uses the concept of descriptor data structure. In this method, instead of including the data in the application profile, a 16-bit value is kept and acts as a pointer to the location of the data. This pointer is referred to as the data descriptor. When a device discovers the presence of another device in the network, the device descriptions are transferred to provide the essential information regarding the new device.

3.6.4.2 The ZigBee Device Objects

Figure 3.5 shows the ZigBee Device Objects (ZDO) as an interface between the APS sub-layer and the application framework. The ZDO are responsible for initializing the APS, NWK, and Security Service Provider (SSP). Similar to the application profiles defined in the application framework, there is a profile defined for the ZDO, which is known as the ZigBee Device Profile (ZDP), or simply the device profile. The device profile contains device descriptions and clusters, but the device profile clusters do not employ attributes.

The ZDO itself has configuration attributes, but these attributes are not included in the device profile. Another difference between the device profile and any application profile is that the application profile is created for a specific application, whereas the device profile defines capabilities supported by all ZigBee devices. The device profile has only one device description. The clusters are divided into two groups of mandatory and optional clusters. The mandatory clusters must be implemented in any ZigBee device.

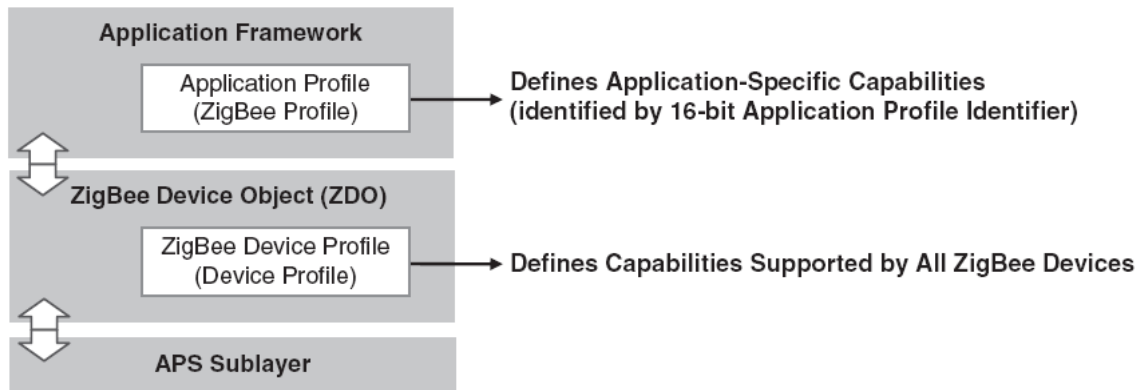


Figure 3.5: The ZDO Acts as an Interface Between the Application Framework and the APS Sub-layer

3.6.4.3 The APS Sub-layer

The APS sub-layer provides data service to both application objects and ZigBee Device Objects through the APS sub-layer Data Entity (APSDE). The APSDE receives the data that needs to be transmitted in the form of a Protocol Data Unit (PDU) from either ZDO or an application object. The APSDE adds proper headers to the PDU to create an APS data frame, which will be passed down the NWK layer. The APS sub-layer Management Entity (APSME) contains primitives to perform three tasks; bind management, APS Information Base (AIB) management, and group management. The binding primitives (APSME-BIND request and APSME-UNBIND request) allow the next higher layer to request to bind two devices by creating an entry in the local binding table or to unbind two devices by removing the corresponding entry from the local binding table. The APSME-GET request and APSME-SET request primitives allow the next higher layer to read and write an attribute in the APS Information Base. The group management primitives are used to add (or remove) certain endpoints of the node in a group table.

ZigBee Applications

4.1 Security Systems

A security system can consist of several sensors, including motion detectors, glass-break sensors, and security cameras. These devices need to communicate with the central security panel through either wire or a wireless network. ZigBee-based security systems simplify installing and upgrading security systems. Despite ZigBee's low data rate, it is still possible to transfer images wirelessly with acceptable quality. For example, ZigBee has been used in a wireless camera system that records videos of visitors at a home's front door and transmits them to a dedicated monitor inside the house.

4.2 Meter-Reading Systems

Utility meters need to be read on a regular basis to generate utility bills. One way to do so is to read the meters manually at homeowners' premises and enter the values into a database. A ZigBee-based automatic meter-reading (AMR) system can create self-forming wireless mesh networks across residential complexes that link meters with utilities' corporate offices. AMR provides the opportunity to remotely monitor a residence's electric, gas, and water usage and eliminate the need for a human visiting each residential unit on a monthly basis.

4.3 Irrigation Systems

A sensor-based irrigation system can result in efficient water management. Sensors across the landscaping field can communicate to the irrigation panel measuring the soil moisture level at different depths. The controller determines the watering time based on moisture level, plant type, time of day, and the season. A distributed wireless sensor network eliminates the difficulty of wiring sensor stations across the field and reduces the maintenance cost.

4.4 Light Control Systems

Light control is one of the classic examples of using ZigBee in a house or commercial building. In traditional light installation, to turn on or off the light it is necessary to bring the wire from the light to a switch. Installation of a new recess light, for example, requires new wiring to a switch. If the recess light the switch are equipped with ZigBee devices, no wired connection between the light and the switch is necessary. In this way, any switch in the house can be assigned to turn on and off a specific light.

4.5 Consumer Electronics: Remote Control

In consumer electronics, ZigBee can be used in wireless remote controls, game controllers, a wireless mouse for a personal computer, and many other applications.

4.6 Industrial Automation

At the industrial level, ZigBee mesh networking can help in areas such as energy management, light control, process control, and asset management.

4.7 Healthcare

One of the applications of IEEE 802.15.4 in the healthcare industry is monitoring a patient's vital information remotely. Consider a patient who is staying at his home but for whom it is important that his physician monitor his heart rate and blood pressure continuously. In this system, an IEEE 802.15.4 network can be used to collect data from various sensors connected to the patient. The 802.15.4 standard uses 128-bit Advanced Encryption Standard (AES) technology to securely transfer data between ZigBee devices and other networks.

Simulation with OMNeT++

5.1 Introductions.

OMNeT++ is an object-oriented modular discrete event network simulation framework [3]. It has a generic architecture, so it can be (and has been) used in various problem domains:

- modeling of wired and wireless communication networks
- protocol modeling
- modeling of queuing networks
- modeling of multiprocessors and other distributed hardware systems
- validating of hardware architectures
- evaluating performance aspects of complex software systems •

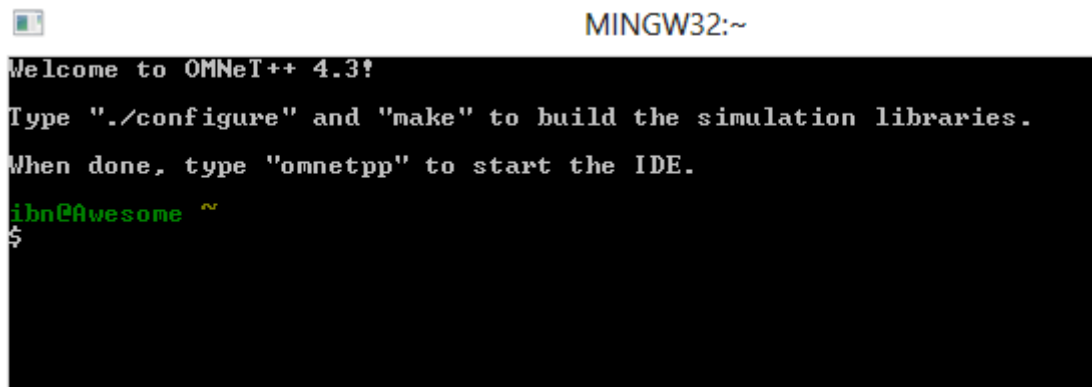
In general, modeling and simulation of any system where the discrete event approach is suitable, and can be conveniently mapped into entities communicating by exchanging messages. OMNeT++ itself is not a simulator of anything concrete, but rather provides infrastructure and tools for writing simulations. One of the fundamental ingredients of this infrastructure is a component architecture for simulation models. Models are assembled from reusable components termed modules. Well-written modules are truly reusable, and can be combined in various ways like LEGO blocks. Modules can be connected with each other via gates (other systems would call them ports), and combined to form compound modules. The depth of module nesting is not limited. Modules communicate through message passing, where messages may carry arbitrary data structures. Modules can pass messages along predefined paths via gates and connections, or directly to their destination; the latter is useful for wireless simulations, for example. Modules may have parameters that can be used to customize module behavior and/or to parameterize the model's topology. Modules at the lowest level of the module hierarchy are called simple modules, and they encapsulate model behavior. Simple modules are programmed in C++, and make use of the simulation library. OMNeT++ simulations can be run under various user interfaces. Graphical, animating user interfaces are highly useful for demonstration and debugging purposes, and command-line user interfaces are best for batch execution [4].

The simulator as well as user interfaces and tools are highly portable. They are tested on the most common operating systems (Linux, MacOS/X, Windows), and they can be compiled out of the box or after trivial modifications on most Unix-

like operating systems. OMNeT++ also supports parallel distributed simulation. OMNeT++ can use several mechanisms for communication between partitions of a parallel distributed simulation, for example MPI or named pipes. The parallel simulation algorithm can easily be extended, or new ones can be plugged in. Models do not need any special instrumentation to be run in parallel it is just a matter of configuration. OMNeT++ can even be used for classroom presentation of parallel simulation algorithms, because simulations can be run in parallel even under the GUI that provides de tailed feedback on what is going on. OMNEST is the commercially supported version of OMNeT++. OMNeT++ is free only for academic and non-profit use; for commercial purposes, one needs to obtain OMNEST licenses from SimulcraftInc.

5.2 Installing OMNeT++

First and foremost, you need to download the latest and free version of OMNeT++ from the OMNeT++ page (<http://www.omnetpp.org/>) and it will be in a compressed (i.e. archive) file. Extract this file into a directory whose directory path has no name with white spaces (for example c:\myomnet). You will find a folder named doc wherein you can find a pdf file with instructions on how to install OMNeT++. To install OMNeT++ on windows, after extracting the file you look for a file named “mingwenv “. On double clicking of this file a command window will open as shown in Figure 5.1. Your command window will have your user name and computer name printed in green.



```
MINGW32:~
Welcome to OMNeT++ 4.3!
Type "./configure" and "make" to build the simulation libraries.
When done, type "omnetpp" to start the IDE.
ibn@Awesome ~
$
```

Figure 5.1: The MINGW32 command window.

On this command widow, type in the command “./configure” and press Enter to execute it. This will configure OMNeT++ and when it is done configuring, you will be prompted to execute the “make” command and it will take about 10minutes to complete the installation after which you can run the ide by typing the command “omnetpp”. You can navigate to the OMNeT++’s workbench by clicking on the icon name workbench of the home window. Also you can create a desktop shortcut of the ide from the folder named ide in which you find omnetpp.

5.3 OMNeT++'s Simulation Frameworks.

OMNeT++ simulation frameworks are basically projects built on the basis of different network simulations like wireless and wired. Some of these frameworks are: INET, INETMANET, and veins, ReaSE, MiXiM and Castalia as shown in Figure 5.2. We focus our attention to the MiXiM simulation framework.

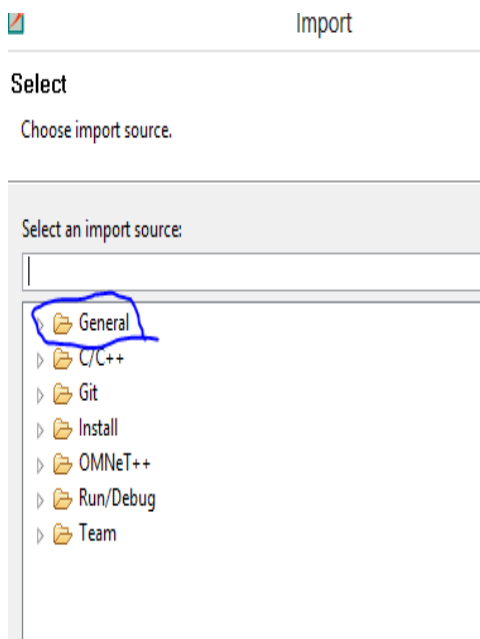


Figure 5.2: OMNeT++ frameworks

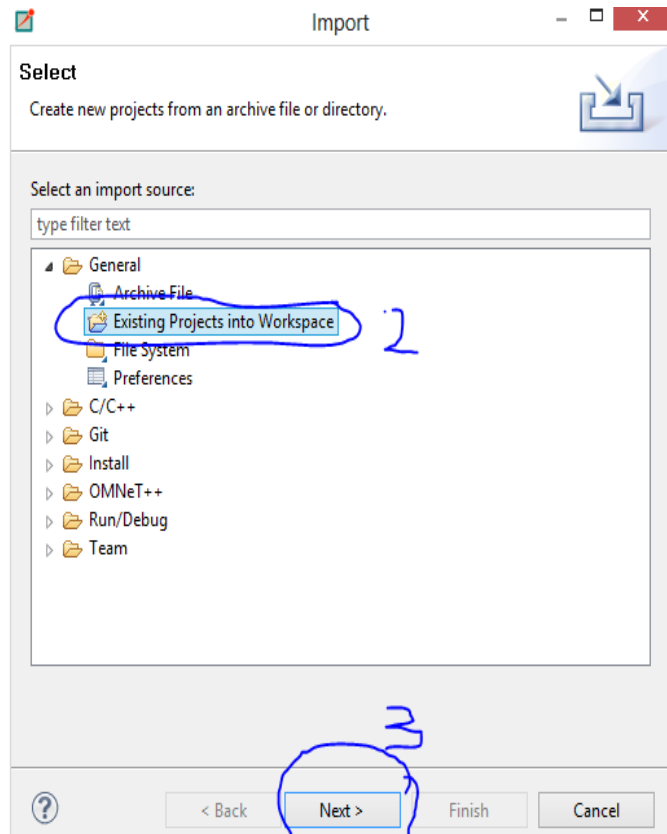
MiXiM is an OMNeT++ modeling framework intended for mobile and fixed wireless networks like wireless sensor networks, ad-hoc networks, vehicular networks and so forth. It offers detailed models of radio wave propagation, interference estimation, radio transceiver power consumption and wireless MAC protocols (e.g. ZigBee). It merges several OMNeT++ frameworks written to support mobile and wireless simulation among which include ChiSim by Universitaet Paderborn, Mac Simulator by Technische Universiteit Delft, Mobile framework by Technische Universiteit; Telecommunication Networks Group and Postif Framework by Technische Universiteit Delft.

5.3.1 How to Install the MiXiM Framework in OMNeT++

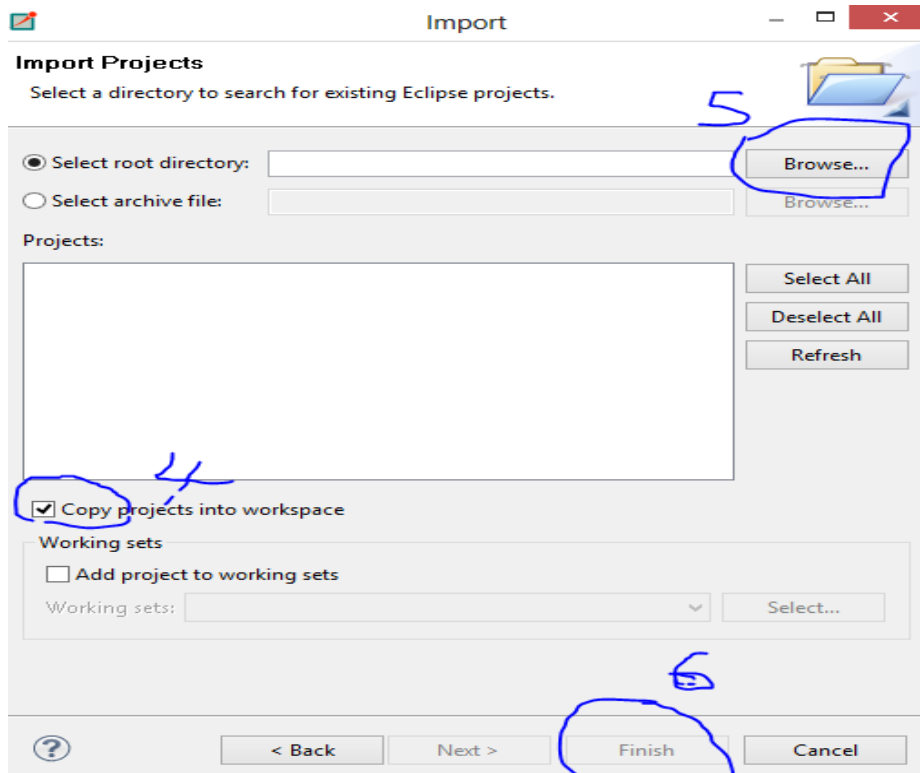
Obtain a copy of the latest version of MiXiM from the MiXiM website (<http://MiXiM.sourceforge.net>) [5]. With this done and with the OMNeT++ IDE opened, migrate to the work bench and go to files, there you find import which on clicking, opens a window as shown in Figure 5.3 (a). Expand General and chose “Existing Project into Workspace” then click “Next” as shown in Figure 5.3 (b). A new window opens as in Figure 5.3 (c) where you can browse to the folder location where MiXiM is found in your computer. If you are importing the uncompressed folder then you would want to make sure you check “copy to workbench”. After importing, open the MiXiM project by right clicking on the project and left clicking “open project”. You need to build your project, (in this case MiXiM) by navigating to project and clicking build all. Now you will find the binaries file after you open your project; meaning it’s been built and you can run the examples that come along with MiXiM.



(a)



(b)



(c)

Figure 5.3: Importing MiXiM into OMNeT++ Workbench.

5.3.2 Creating your own project using MiXiM

MiXiM has a built in template that enables you to build your own OMNeT++ project, without having to write your own files from scratch and program them in C++. To create your own project using MiXiM, go to file and select New -> OMNeT++ Project. Give the name of your project on the dialog box that pops on and click next (Figure 5.4(a)).

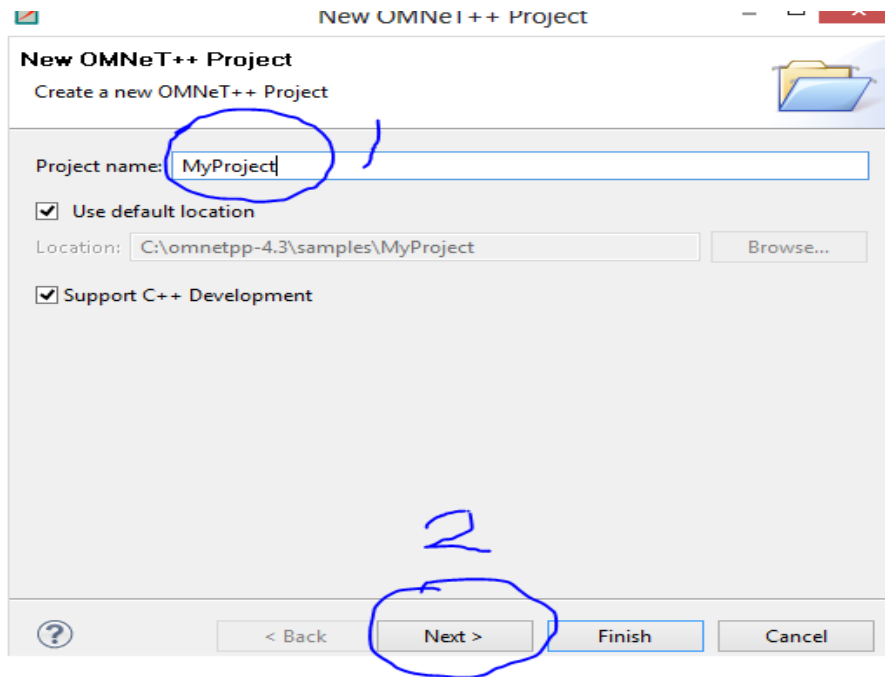


Figure 5.4(a)

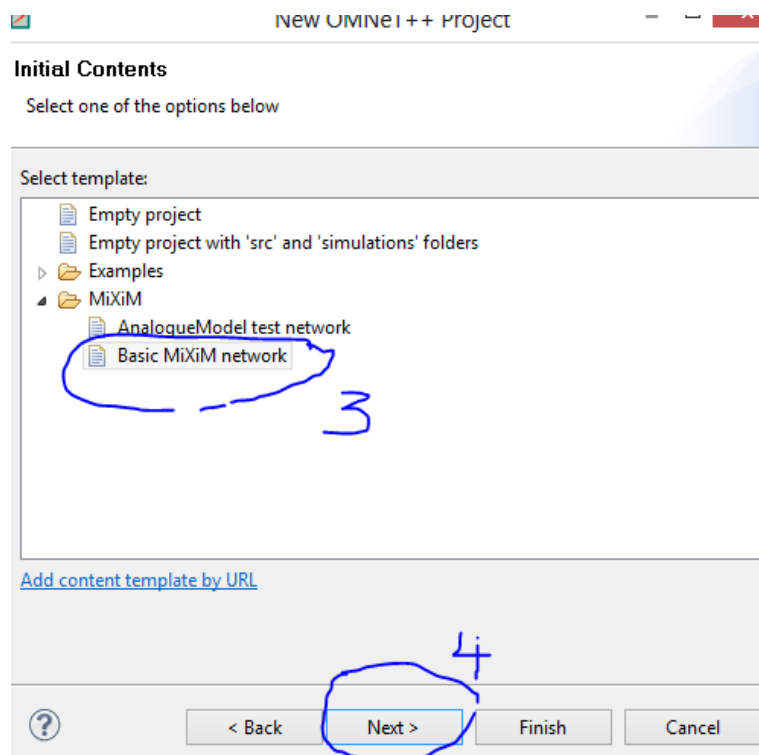


Figure 5.4(b)

Select a MiXiM template from among the two provided (for our case Basic MiXiM network) and click “Next” (Figure 5.4(b)). Now chose your desired network settings from the dialog box that pops up (for IEEE 802.15.4 we chose Application layer: sensor application layer, Network layer: wise route, Nic protocol: CSMA 802.15.4, Mobility module: Static mobility and Playground: 2-dimensional) and click finish as is illustrated in Figure 5.4(c). Now your project is created and you can find it on the project panel of the workbench which will open with files as shown in (Figure 5.4(d)). Notice that when your project is opened, the MiXiM project will automatically open as well. This is because MiXiM serves as a reference project and most the files in your project are borrowed from MiXiM. You can verify this by right clicking on your created project, selecting properties and then references. You will find that MiXiM is selected you can add other references if you want to borrow from other projects in the workbench.

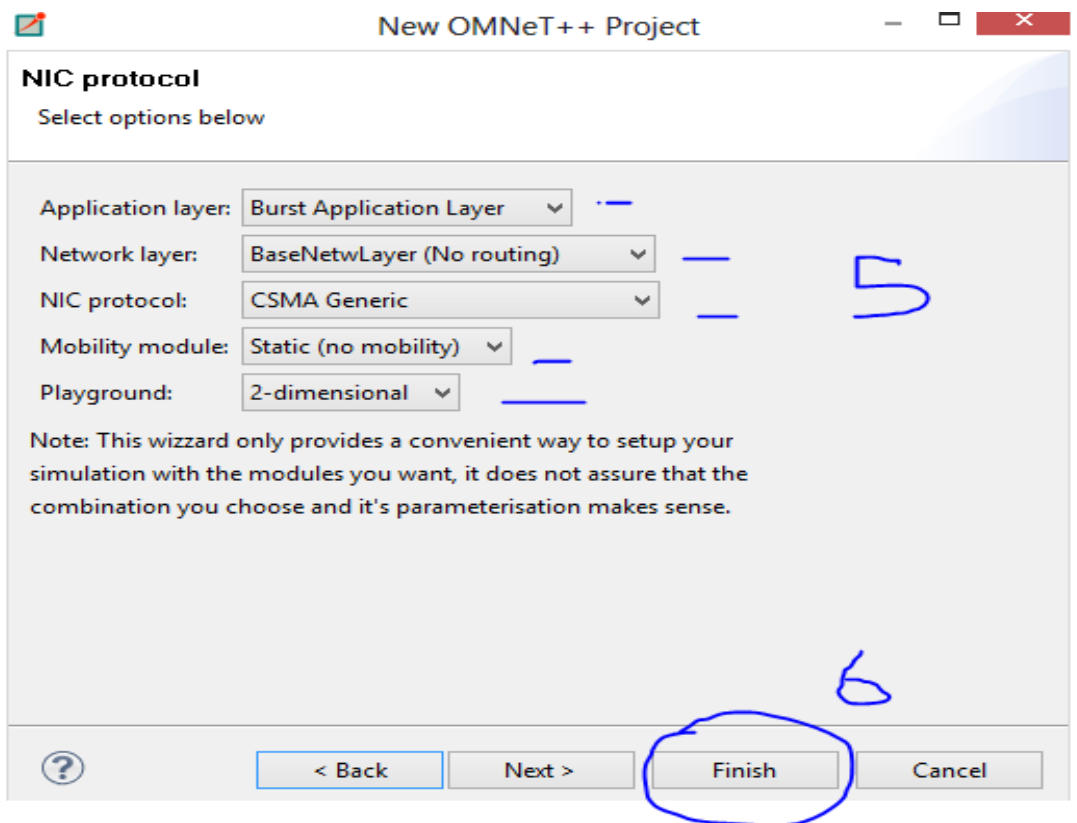


Figure 5.4(c)

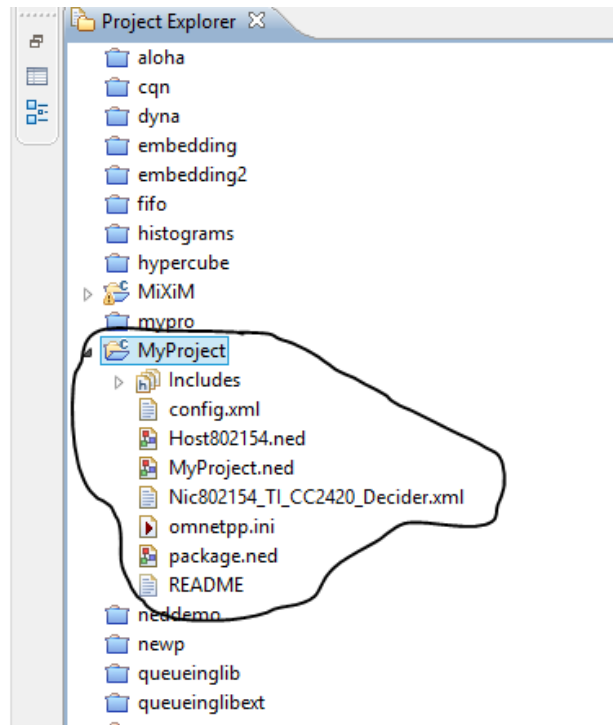


Figure 5.3(d)

Now your project is ready but you need to make one correction before it will run correctly. Open the Omnet.ini file and chose the source editor and do the corrections shown in Figure 5.4(e) (i.e. add time unit to trafficParam and change mobility type from “baseMobility” to “stationaryMobility”). You can now save, build and run your project. You can further customize your project to your desire, by defining experiments or tests in the ini file.

```

*omnetpp.ini
**.battery.voltage = 3.3V
**.battery.resolution = 10s
**.battery.publishDelta = 0.1
**.battery.publishTime = 0
**.battery.numDevices = 1
##### Application layer parameters #####
**.node[*].applicationType = "SensorApplLayer"
i **.appl.trafficType = "periodic"
**.appl.trafficParam = 1s in seconds
**.appl.broadcastPackets = true
**.appl.nbPackets = 3
**.appl.initializationTime = 10s
##### NETW layer parameters #####
**.node[*].networkType = "WiseRoute"
**.node[*].netw1.trace = true
**.node[*].netw1.stats = true
i **.node[*].netw1.useSimTracer = false
**.node[*].netw1.headerLength = 24 bit
i **.node[*].netw1.sinkAddress = 0
# RSSI threshold for route selection
i **.node[*].netw1.rssiThreshold = -50 dBm
# If set to zero, this node does not initiates route tree building.
# If set to a value larger than zero, this nodes periodically initiates route tree building.
**.node[*].netw1.routeFloodsInterval = 100 s
##### Mobility parameters #####
i **.node[*].mobilityType = "StationaryMobility"
i **.node[*].mobility.debug = false
**.node[*].mobility.updateInterval = 0.1s
**.node[0].mobility.initialX = 150m

```

Figure 5.4(e)

Now to create experiments in your project, you have to configure parameters that are important for your experiment and then you chose a configuration name which should be in brackets appended by config for example [config Test1-A]. You can create a number of different experiments in your project, by choosing different parameter configuration per experiment. In the configuration of Figure 5.4(f) parameters such as playgroundSize, packetsPerPacketTime, speed and node's initial position, are kept constant while the number of host (numHost) varies from 1 to 40 with an increment of 2 per each run of simulation. In order to obtain results that can be plotted in a graph, you must give statistical values to that parameter, whose effect on the network you are testing; for example varying parameters such as the number of hosts or distance between Hosts etc while keeping the other parameters constant. A sampled test configuration in the “ini file” is shown in Figure 5.4(f).

```

#####
[Config Test1-A]
description = "increasing number of hosts, unbursted, with queue"
*.playgroundSizeX = 10m
*.playgroundSizeY = 10m
*.numHosts = ${2..40 step 2}
*.node[*].mobility.initialX = uniform(0m, 10m)
*.node[*].mobility.initialY = uniform(0m, 10m)
*.node[*].mobility.initialZ = uniform(0m, 10m)
*.node[*].mobility.speed = 0
**.netwl.packetsPerPacketTime = 0.1

```

Figure 5.4(f)

Figure 5.4: Creating a Simulation using MiXiM

5.4 Simulation

After building the project you have to run the simulation and it will run according to the simulation time limit (i.e. *sim-time-limit* = value in time unit), set in the “ini” file. To run your simulation, right click on the *omnetpp.ini* file and select *run as-> Omnet++ simulation*. Assuming that you have defined your experiment(s), when you run your simulation, a dialog box pops up and you can chose the experiment you want to run and then the run number of the experiment or test (see Figure 5.5). The statistic parameter is incremented per each run of simulation and thus each simulation run has a different value.

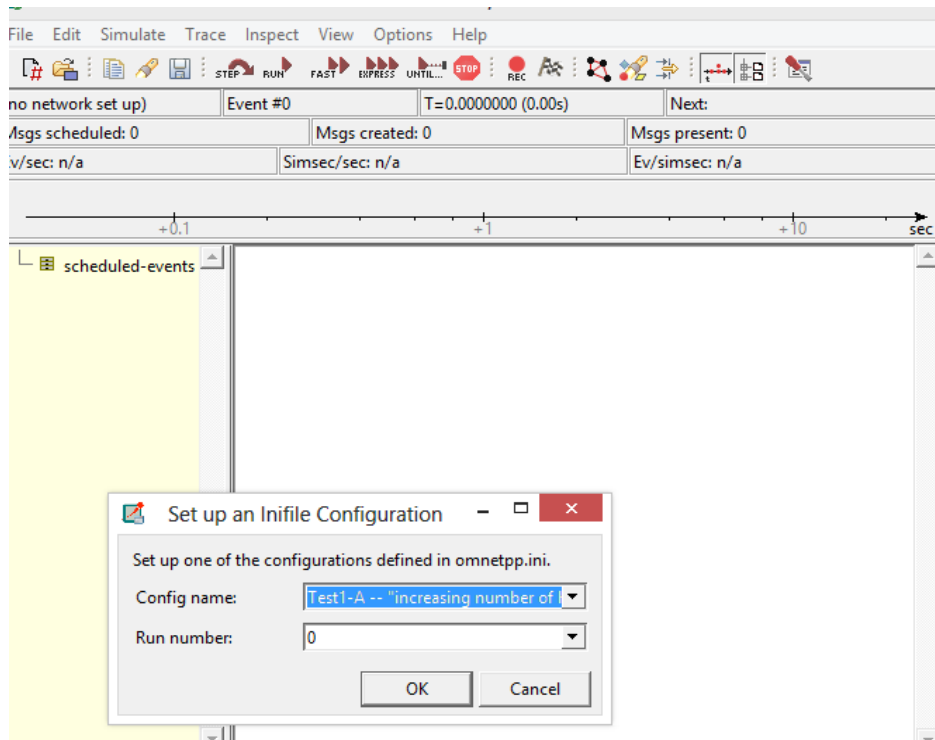


Figure 5.5: Omnet++/Tkenv editor.

Simulation using OMNeT++ is visualized in the Omnet++/Tkenv editor which is seen as a live animation. Here you are able to see how messages are communicated live between different nodes.

In our case three Tests are defined. In the first test, traffic (data bits) is generated and the traffic received in a wireless medium is studied and the results are plotted in a scatter chart (graph). The Test2 configurations are meant to show the effect of MAC-ACKS on the usage of the channel with increasing distance between the hosts. Test3 is meant to show that CSMA uses exponential back-offs by increasing the number of hosts in the network off course exponentially.

5.5 Results and Interpretation

The simulation results are stored in a results folder after termination of simulation and these results can be viewed and plotted in the “anf” file. Most of the results recorded may not make any sense so you have to sort the result of interest and create a dataset with these results. With your dataset, you can create a scatter chart, which will then plot your results in a graph.

N.B. For scalar parameters you have to run a particular test with many different run numbers (mostly 0 to 9). This gives you many varying scalar values that can then be sorted and plotted in a scatter chart.

5.5.1 Test1:

Test1 is made up of five different tests with different configurations. These tests produce the usage statistics of the channel for different configurations. Here usage statistics means how much of the channels possible maximum capacity is used at which amount of generated traffic. The results of these tests are visualized when the “Test1.anf” (Figure 5.6(a)) file is opened and shows traffic generated vs. traffic received where both values are normalized against bitrate and the simulation time. This is done in order to render the values independent of either simulation time or traffic. On double clicking the Test1.anf file, the anf editor opens and you can plot your results on a graph by navigating to dataset and double clicking on scatter chart found under dataset (see Figure 5.6(b)). The results obtained are visualized in Figure 5.6(c).

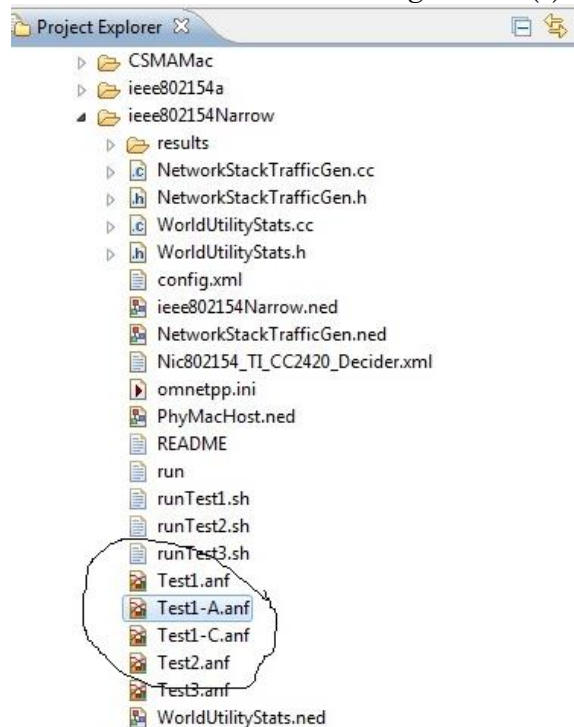


Figure 5.6(a): The anf file under project.

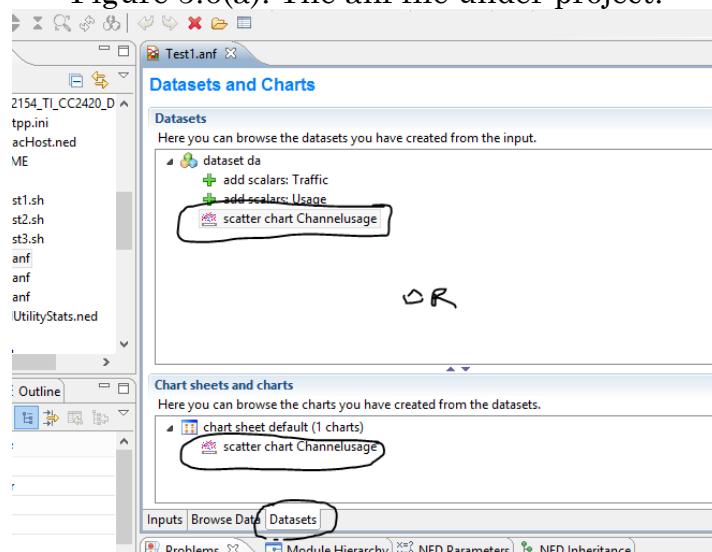


Figure 5.6(b): The anf editor

Creating a plot from the recorded data is done in the anf file. You can create this file manually by right clicking on your project and selecting *new-> Analysis File (anf)* and giving a name to your anf file or it can be done automatically by double clicking on .sca file found in the results file after you had run the simulation. The anf file created manually will not have any data thus you have to add either by drag and drop or by clicking on wildcards and giving the path name of your data file (e.g. /MyProject/General*.anf).

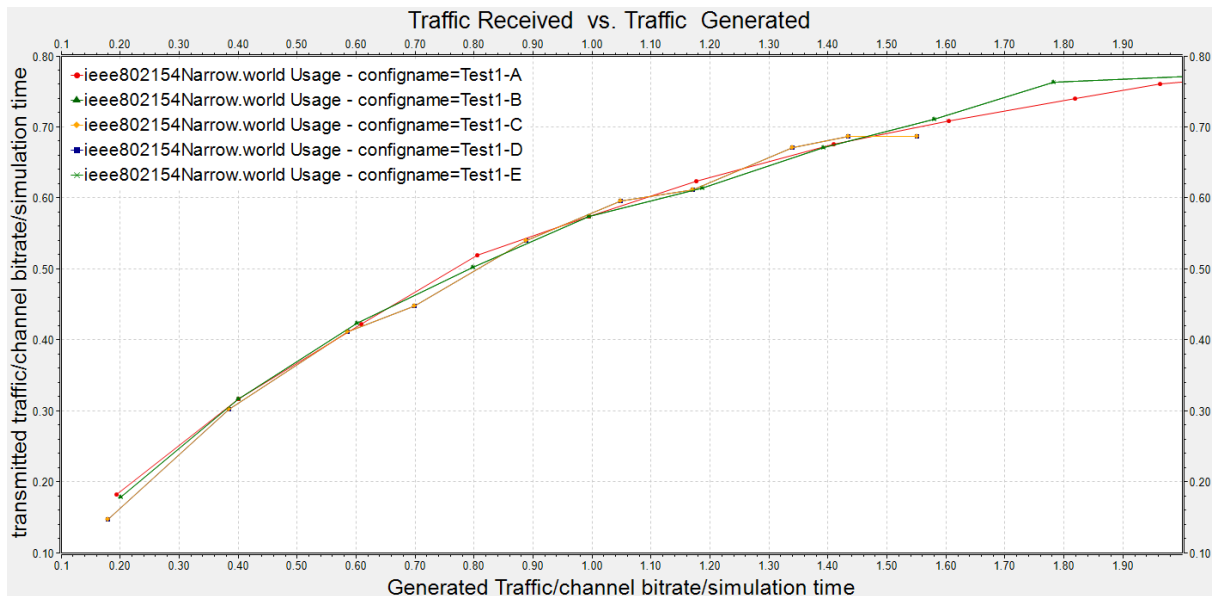


Figure 5.6(c): Traffic Transmitted vs. Traffic Generated

Creating the anf file and getting your data results into it is not yet enough to plot a graph. To create a dataset, navigate to dataset on the anf editor and there, click on dataset on the right panel of this window and name your dataset on the dataset pop up window. Now with your dataset selected, click on Add to add data to your dataset. Put a filter expression for the data you want to plot. The best way to filter is by filtering each set of data individually. The filter pattern for the plot in Figure 5.6(c) is; “Traffic”, and “Usage”. With dataset now selected clicking on scatter chart on the right panel will open a window where you can select the properties of your graph such as axes label, Font size of headings and labels, which parameter is plotted on the abscissa etc.

The graph shows five different curves for the five experiments. In Test1-A, we increase the number of hosts, unburst, with queue and monitor the amount of traffic received. We observe that as the number of hosts increases, the amount of traffic increases by default. The received traffic also increase with increase in the generated traffic. At one point (saturation) when the generated traffic increases the received traffic remains constant. This is due to several reasons like for example increase in interference due to so many hosts communicating. In Test1-B we increase send rate per host, unburst, with queue. The graph also has the same pattern as that of Test1-A, but goes to saturation faster. In Test1-C send

rate is also increase but bursted, with queue. In test1-D we increase send rate per host, bursted, without queue and in Test1-E send rate per host is increased, unbursted without queue. Some of the graphs appear to double on others like 1-c and 1-E.

5.5.2 Test2

The experiments of Test2 are configured with the aim to show the effect of mac-Acks on increasing distance between the hosts. Furthermore Test2-C is intended to show the effect of the hidden station problem of the CSMA (carrier sense multiple access). The results of this Test are plotted as shown Figure 5.7 below.

NB: To get a complete plot for Test2 experiments you have run at least 25 simulations with different run numbers (1...25), for each of the experiments. The same procedure for plotting Test1 can be followed except that the sort pattern here is “run(*) AND name(Usage)” and “run(*) AND name(dist)”.

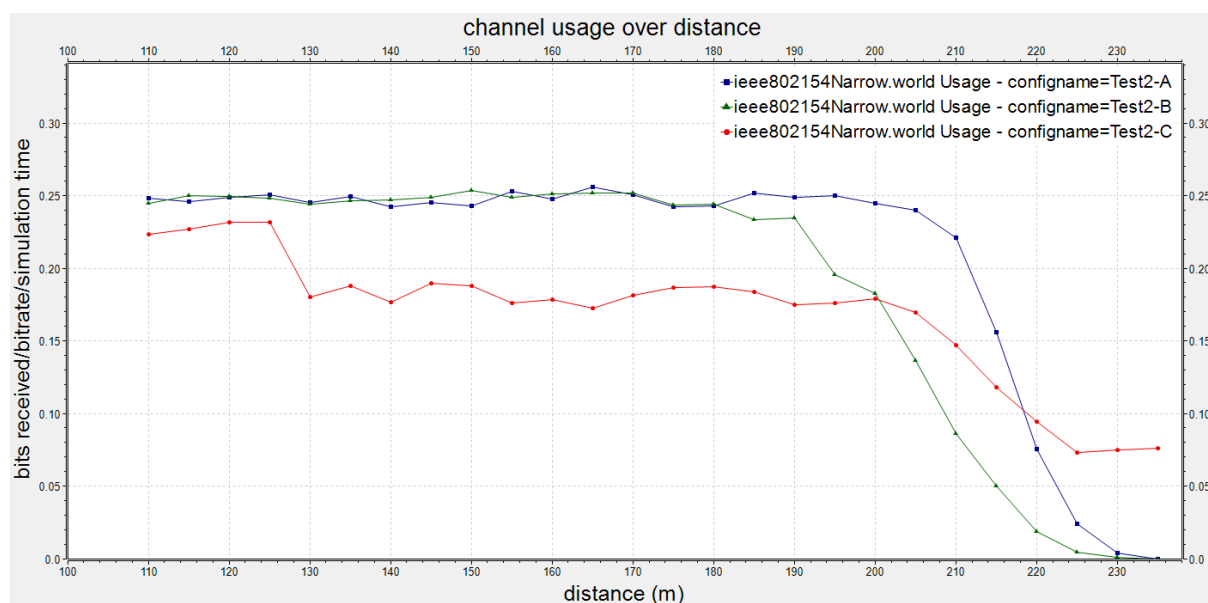


Figure 5.7: Bits receive vs. Distance between hosts.

The graphs of Figure 5.7 shows three different test results where Test2-A is done by moving one sending host towards the sink with mac acks. It is seen that the number of bits received for this test keep still longer than the other two tests. In Test2-B, one sending host is moving towards the sink but without mac acks and its graph start falling earlier as compared to the former test. Test2-C on the other hand has one host stationary in the range of the sink which off course is the hidden station problem. The hidden station problem arises when three nodes (hosts) A, B and C are communicating as demonstrated in Figure 5.8 below.

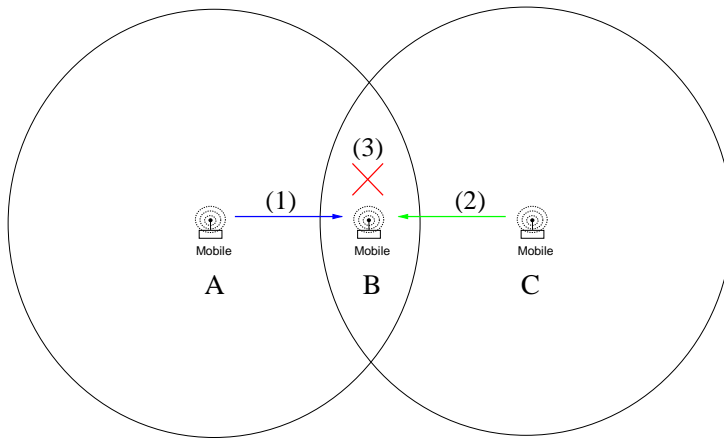


Figure 5.8: Hidden station problem of CSMA

Host A is communicating with B and C is not sensing A, and communicates with B. Now interference occur at B i.e. collision and this what is referred to as the hidden station problem. This problem can be resolved by introducing RTS/CTS reservation handshaking. This hidden station problem is demonstrated in the curve of Test2-C between the distances of 125 m and 130 m.

5.5.3 Test3

Test3 is intended to show that CSMA (carrier sense multiple access) uses exponential back-offs and this is done by exponentially increasing the number of hosts in the network. As the average back-off duration of each host increases exponentially, the average number of back-offs increases linearly as is shown in Figure 5.9 below.

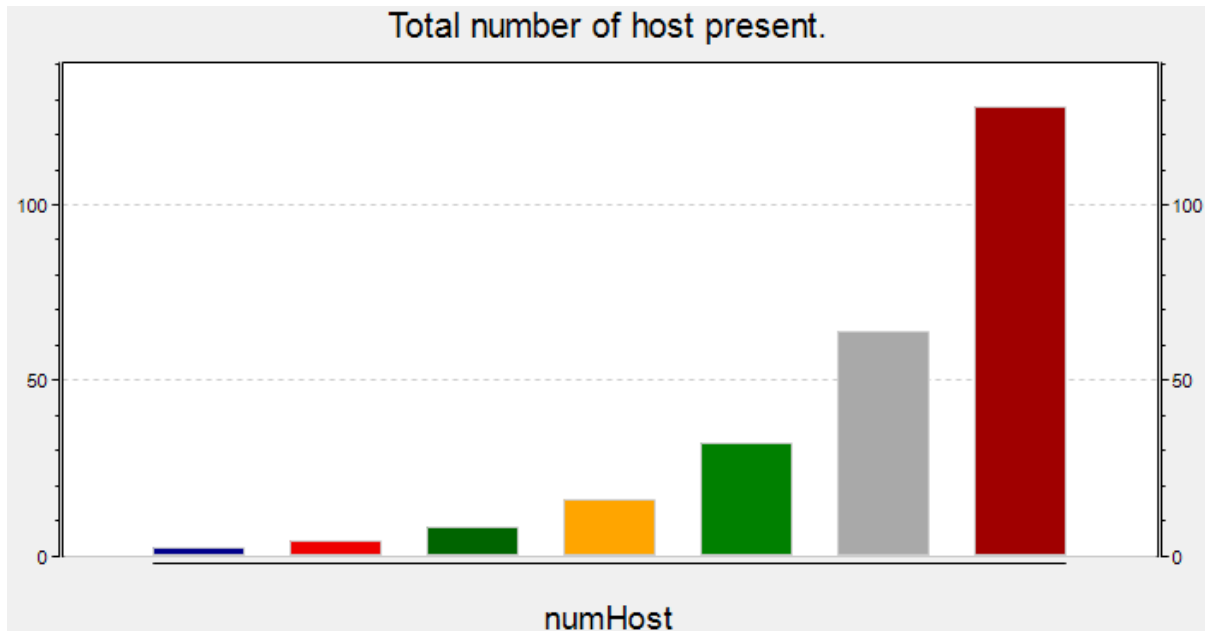


Figure 5.9(a): Number of hosts per simulation

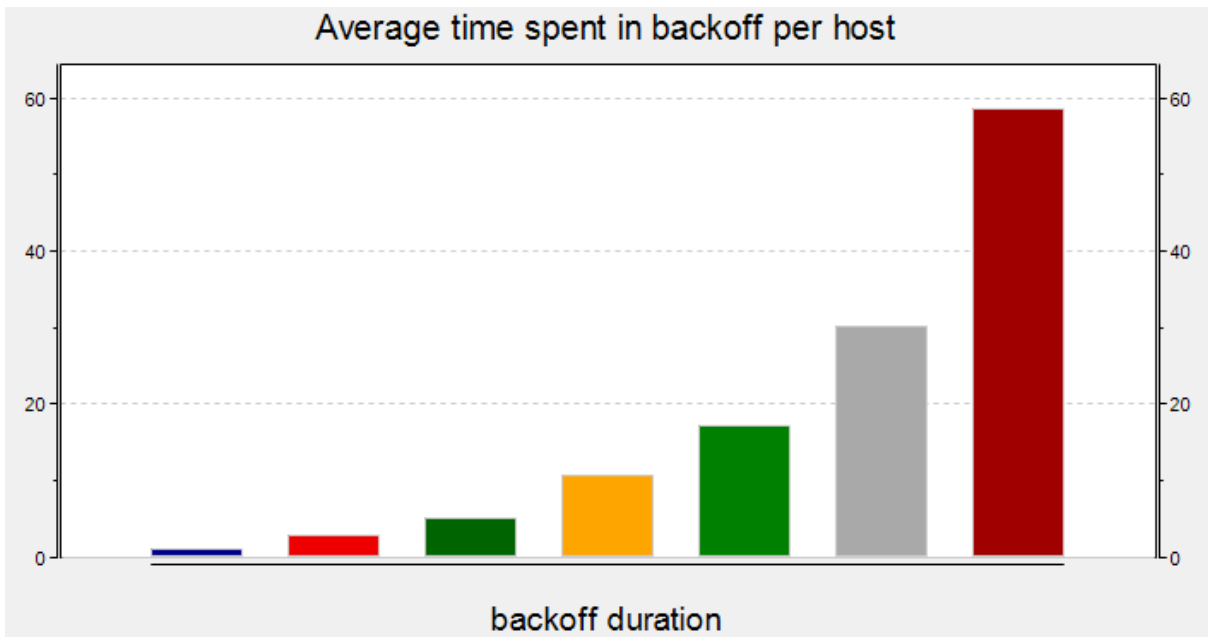


Figure 5.9(b): Time spent in Back off

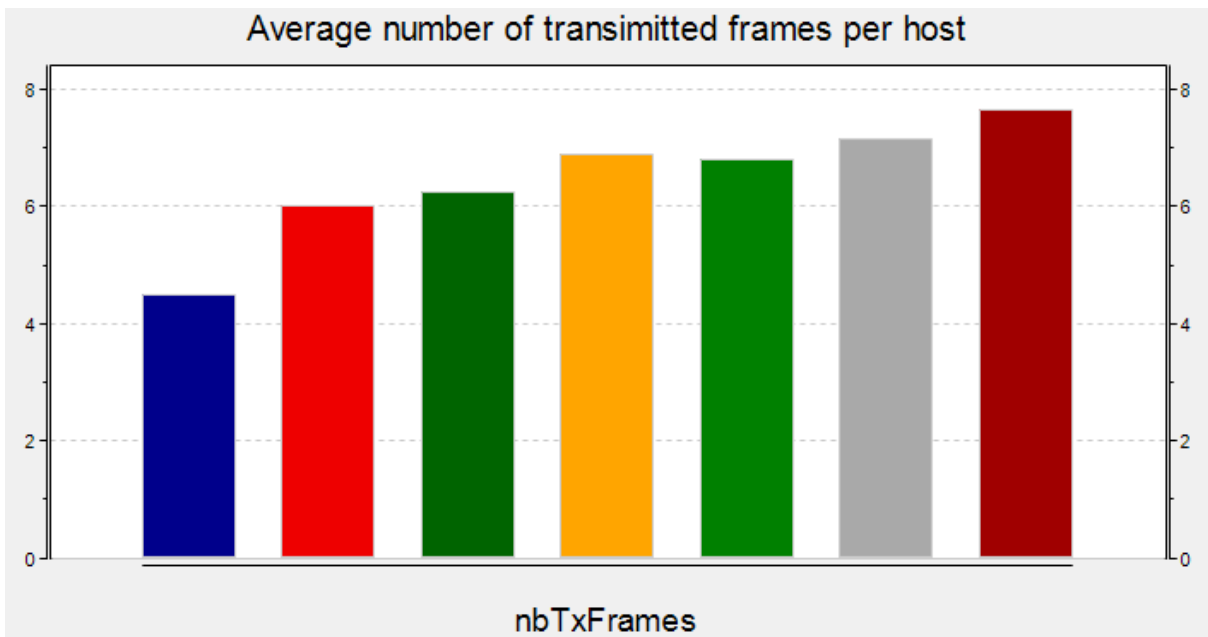


Figure 5.9(c): Transmitted Frames per simulation.

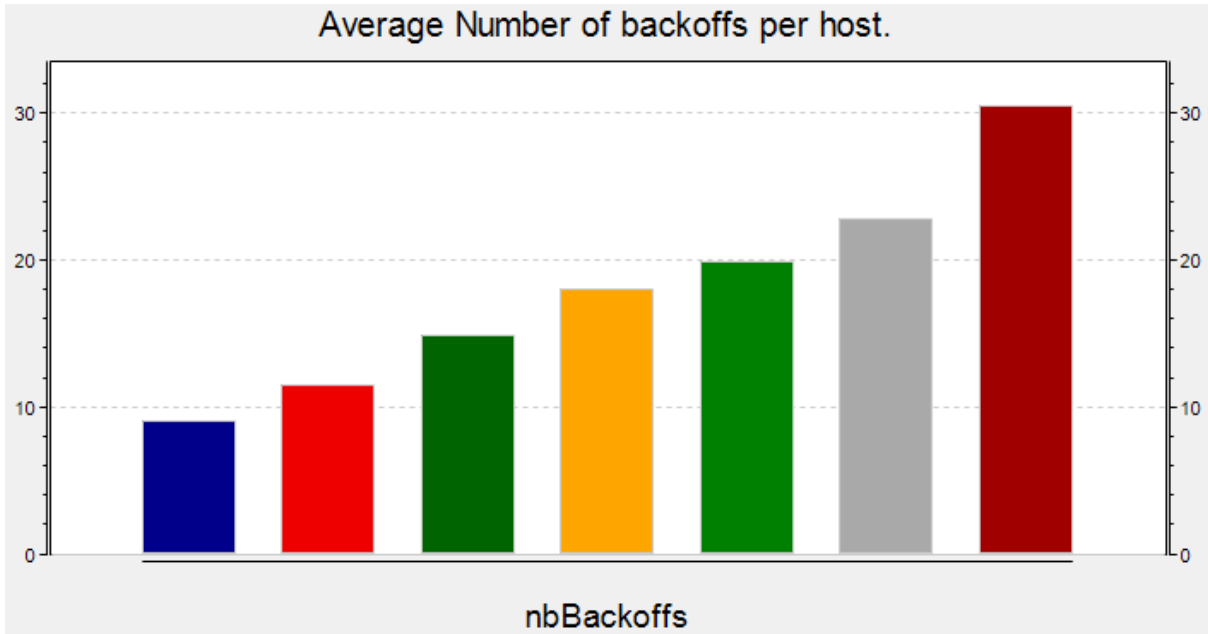


Figure 5.9(d): Back off number per host per simulation

Figure 5.9: Simulation results of Test3

Figure 5.9 shows four histograms from the results of Test3. Figure 5.9(a) is the histogram of exponential increase in number of hosts present. This is the base center of Test3 as the number of hosts are increased per each run and the other parameters like the average number of back-off per host per each run of simulation, average number of frames transmitted per host per simulation run and the average time spent in back-off per host per simulation run. The results of seven simulation runs are plotted in a histogram as depicted by Figure 5.9. Figure 5.9(b) shows that the average time spent in back-off per host increases exponentially as the total number of hosts present. The Figure 5.9(c) shows that the average number of back-offs per host increases almost linearly with each run of the experiment and this is due to exponential back-offs. Finally Figure 5.9(d) shows the average number of transmitted frames per host with bars between 5 and 10 frames per host.

Conclusion and future work

6.1 Conclusions

The ZigBee sensor network field is an interesting field as far as low rate wireless personal network is concerned. It has got far much to offer for applications in which lower bit rate is not an issue, and cost less compared to other wireless network domains.

This thesis portrays a simulation of ZigBee using OMNeT++. The OMNeT++ is a very interesting simulator as it tries to mimic live network communication of ZigBee nodes using animations. The OMNeT++ simulator is also a complicated simulator when you have to create your own nodes and program them using C++, but most of the work is already done in the OMNeT++ frameworks. You don't necessary need to know how to code in C++ before you can do a simulation using OMNeT++.

6.2 Future work.

So many works have been going on, on wireless sensor networks, aiming to make the most efficient and less costly wireless sensor network with lower power consumption. Our work serves as a guide for those who want to create his own simulations using OMNeT++ and experiment with real time measurement. Most of ZigBee features such as the coordinator control and the peer to peer network topologies can be simulated with OMNeT++ simulator.

A prototype of ZigBee network with ZigBee devices can be used to do real time experiments and the results could be compared with those obtained by simulation.

References

- [1] C. Wei, *et al.*, "On the performance of TCP over throughput-optimal CSMA," in *Quality of Service (IWQoS), 2011 IEEE 19th International Workshop on*, 2011, pp. 1-9.
- [2] S. Farahani, *ZigBee Wireless Networks and Transceivers*: Newnes, 2008.
- [3] *OMnet++ Source*. Available: <http://www.omnetpp.org/>
- [4] C. Feng, *et al.*, "Performance Evaluation of IEEE 802.15.4 LR-WPAN for Industrial Applications," in *Fifth Annual Conference on Wireless on Demand Network Systems and Services, 2008. WONS 2008.* , 2008, pp. 89-96.
- [5] "MiXiM source."