Feasibility and performance analysis of sensor modeling in OPNET

Niranjan Krishnamurthi

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
Feasibility and Performance Analysis of Sensor Modeling in OPNET

by

Niranjan Krishnamurthi

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

Supervised by

Assistant Professor Dr. Shanchieh Jay Yang
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
August 2007

Approved By:

Dr. Shanchieh Jay Yang
Assistant Professor
Primary Adviser

Dr. Chance M. Glenn, Sr.
Director of The Laboratory for Advanced Communications Technology

Dr. Dhireesha Kudithipudi
Assistant Professor, Department of Computer Engineering
Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title: Feasibility and Performance Analysis of Sensor Modeling in OP-NET

I, Niranjan Krishnamurthi, hereby grant permission to the Wallace Memorial Library reproduce my thesis in whole or part.

Niranjan Krishnamurthi

Date
Dedication

To my parents and brother, Dr. Krishnamurthi, Dr. Rekha and Dr. Sudarshan.
I would like to thank Dr. Shanchieh Jay Yang, my primary adviser, for being a major source of inspiration and also a driving force behind this work. Much appreciation also goes out to Dr. Dhireesha Kudithipudi and Dr. Chance M. Glenn, Sr. for their contributions, which lead to the success of this project.
Abstract

The current surge in wireless sensor network research has led to the extensive use of network simulation software to simulate new and novel protocols. Analysis via simulation allows for the validation of protocols before being fully deployed in physical networks. Though superior in modeling RF signals and network operations, current wireless network simulation software lack fully functional sensor models that replicate a sensor’s interaction with the environment. Realistic and specific sensor models do exist, however very little has been incorporated into commonly used network simulators. It is questionable whether the typical approach of modeling sensed data as random processes offers anything close to reality. It is, therefore, the goal of this work to investigate the feasibility and benefit of developing user-configurable sensor models in a widely used network simulation software called OPNET.

This work examines various types of sensors and categorizes them based on the way the sensors may be modeled in OPNET. Sensors are grouped into four distinct categories as follows. The first group, called ambient sensors, is used to categorize sensors that sense information in its surroundings such as ambient temperature and humidity. The second group of sensors are called self-characterizing sensors. These type of sensors are used to help a wireless node identify its physical properties, including acceleration and internal temperature. Sensors that help a wireless node identify external objects are called object-characterizing sensors. This group is further subdivided into two sections, intrusive and non-intrusive sensors. An intrusive object characterizing sensor sends out energy in some waveform and interprets data from the waveform reflected back. Sensors in this category include ultrasonic and sonar sensors. Finally, non-intrusive sensors characterize an object
passively, such as that done by infrared heat sensors. Leveraging the well-defined Wireless Module in OPNET, an infrared thermal imaging sensor module, typically used for detection and tracking, will be developed and examined. Results via simulation will demonstrate the benefit of utilizing the developed sensor model as compared to traditional approaches used in network simulation. Feasibility and benefits for modeling other sensor types, such as velocity and global positioning sensors will also be examined.
Contents

Dedication ........................................ iii
Acknowledgments ................................ iv
Abstract ........................................... v
1 Introduction .................................... 1
2 Physical Sensors ................................. 6
   2.1 Sensor Categorization .................... 6
   2.2 Background on Temperature Sensors .... 10
      2.2.1 Thermocouple based Temperature Sensors 10
      2.2.2 Infrared Temperature Sensors .......... 10
   2.3 Background on Infrared Radiation ....... 13
      2.3.1 Electromagnetic Radiation .......... 14
      2.3.2 Heat - Electromagnetic Radiation Relationship 16
      2.3.3 Infrared Temperature Sensor Technology 22
      2.3.4 Thermal Imaging Sensors ......... 24
3 OPNET Simulation Software .................. 26
   3.1 OPNET Architecture ...................... 28
      3.1.1 Model Specification ................. 28
      3.1.2 Modeling Communications with Packets 31
      3.1.3 Data Collection and Simulation .... 32
      3.1.4 Analysis ................................ 32
   3.2 Wireless Module ........................... 33
      3.2.1 Wireless Objects in the Network Domain 33
      3.2.2 Wireless Objects in the Node Domain .... 34
      3.2.3 Wireless Pipeline Stages .......... 38
4 Thermal Imaging Module ................................................. 44
  4.1 Overview ................................................................. 44
  4.2 Packet Format ......................................................... 45
  4.3 Pipeline Stage Modifications ....................................... 46
    4.3.1 Pipeline Stage 0: Receiver Group ......................... 47
    4.3.2 Pipeline Stage 1: Transmission Delay ...................... 47
    4.3.3 Pipeline Stage 2: Link Closure ............................ 48
    4.3.4 Pipeline Stage 3: Channel Match ......................... 48
    4.3.5 Pipeline Stage 4: Transmitter Antenna Gain .............. 48
    4.3.6 Pipeline Stage 5: Propagation Delay ...................... 49
    4.3.7 Pipeline Stage 6: Receiver Antenna Gain ................. 49
    4.3.8 Pipeline Stage 7: Receiver Power .......................... 49
    4.3.9 Pipeline Stages 8 through 12 .............................. 50
    4.3.10 Pipeline Stage 13: Error Correction ..................... 51
  4.4 Infrared Emitting Object Module .................................. 51
    4.4.1 heat_gen Process Model .................................... 52
    4.4.2 heat_radio_tx Transmitter Model ......................... 55
    4.4.3 heat_ant_tx Antenna Model .............................. 57
  4.5 Thermal Imaging Sensor Module ................................... 57
    4.5.1 heat_ant_rx Antenna Model ............................. 57
    4.5.2 heat_radio_rx Receiver Model ............................ 58
    4.5.3 heat_proc Process Model .............................. 59
  4.6 Marker and Ambient Temperature Modules .......................... 74
    4.6.1 Marker Module ........................................... 74
    4.6.2 Ambient Temperature Module .............................. 77

5 Thermal Imaging Module Results ..................................... 80
  5.1 Capabilities .......................................................... 80
    5.1.1 Image Resolution ........................................ 80
    5.1.2 Viewing Capabilities ...................................... 83
    5.1.3 Infrared Properties ....................................... 83
    5.1.4 Emissivity Properties .................................... 83
    5.1.5 Sensor Viewing Limits ................................... 88
  5.2 Performance .......................................................... 88
    5.2.1 Environment Settings ..................................... 92
List of Figures

2.1 The Wyse Bridge before being submerged. ............................ 8
2.2 Image of the Wyse Bridge obtained with side-scan sonars. ........ 9
2.3 Pyroelectric infrared sensor element [3] ................................ 11
2.4 Microbolometer element [14]. .......................................... 12
2.5 Groups of the electromagnetic spectrum divided by wavelength [5]. . 16
2.6 Example heat radiation curves of 3 stars [7]. ............................ 17
2.7 Black body curves for two different temperatures [26]. ............. 21
2.8 Image depicting how Wien’s Law of Displacement and Stefan-Boltzmann Law are derived from Planck’s Radiation Curve [4]. ............ 22
2.9 Image of a section of a microbolometer array [14]. .................... 25
3.1 Simulation Project Cycle [19]. ............................................ 28
3.2 Different types of connections used in a node model [19]. .......... 30
3.3 Multiple transmitters and receivers sharing the same antenna module (a.0) [19]. ................................................................. 35
3.4 Isotropic antenna pattern [19]. ............................................ 36
3.5 Directional antenna pattern with lobes [19]. ............................ 37
3.6 $\phi, \theta$ coordinate system to represent an antenna pattern [19]. ..... 37
3.7 The stages that encompass the wireless pipeline [19]. ............. 39
4.1 Packet format used in the thermal imaging sensor module. ........ 46
4.2 Difference between infrared penetrable and impenetrable objects. . 47
4.3 Code excerpt of changes made to the Receiver Antenna Gain pipeline stage. ............................................................... 49
4.4 Portion of the Receiver Power pipeline stage code that was commented out. ............................................................... 50
4.5 Portion of the Error Correction pipeline stage code that was edited. . 51
4.6 Node Model of the infrared emitting objects. .......................... 51
4.7 Process Model of the infrared emitting objects. ........................ 52
4.8 Infrared emitting object attributes that may be edited by a user. .... 53
4.9 List of emissivity values already predefined in the model. .......... 54
4.10 Portion of the ss_packet_generate() function code. ............ 55
4.11 Edited Transmitter model attributes. ............................ 56
4.12 Node Model of the thermal imaging sensor objects. ................. 57
4.13 Edited Receiver model attributes .................................. 59
4.14 Process Model of the thermal imaging sensor objects. .............. 60
4.15 Thermal imaging sensor attributes that may be edited by a user. .......... 61
4.16 Code excerpt showing the effect of emissivity on temperature calculations. 62
4.17 3-dimensional view of pointing \( \phi \) angle. ......................... 64
4.18 2-dimensional view of pointing \( \phi \) angle. .......................... 65
4.19 Pointing \( \theta \) angle. ................................................. 66
4.20 Translation into x-y coordinates. ..................................... 67
4.21 Radius angle is angle between center and outer edge of object. ........ 68
4.22 Half view angle provides maximum viewing space for each half of the sensor matrix. 69
4.23 Thermal imaging sensor attributes that may be edited by a user. .......... 70
4.24 Visual representation of object rendition on the sensor pixel matrix. .... 71
4.25 Process to calculate the values for \( Y_{\text{min}} \) and \( Y_{\text{max}} \). .................. 72
4.26 Formula to find x and y coordinates of any point on a circle. .......... 73
4.27 PGM image format example. ........................................ 74
4.28 Example network scenario where implementing a Marker Module is beneficial. 75
4.29 Sensor node module attributes to be edited to use a marker. .......... 76
4.30 Ambient temperature module attributes to be edited. ................. 78
4.31 Error messages corresponding to invalid or missing marker and ambient temperature modules. 79

5.1 Effect of resolution on the image quality. ............................. 81
5.2 Effect of distance on object size in captured image. .................. 82
5.3 Effect of distance and size on captured images. ....................... 84
5.4 Effect of distance and size on captured images. ....................... 85
5.5 Environment visualization from different angles. ..................... 86
5.6 Object rendition differences due to penetrability. .................... 87
5.7 Object rendition disparity due to emissivity values. .................. 89
5.8 Object rendition disparity due to emissivity values. .................. 90
5.9 Object rendition disparity due to emissivity values. .................. 91
5.10 Sensor viewing limits test scenario. ................................. 92
5.11 Near-far rendition of object by sensor_0 of scenario depicted in Figure ?? ... 93
List of Tables

2.1 Common emissivity values [18]................................. 18
2.2 Definition of variables for Planck’s Law...................... 19
2.3 Black body temperature ranges and corresponding wavelength ranges [7].... 20
4.1 Network Domain components of the Thermal Imaging Sensor Module........ 44
5.1 System profile of simulation computer............................. 93
Chapter 1

Introduction

Motivation

Over the past decade, advancements in technology have allowed for the use of wireless sensor networks to not only be feasible but extremely desirable in many situations. From the military, to weather detection and even day-to-day home security, the need for these sensors have only increased. The basis for the need of any type of sensor networks is data collection. Data gathered by sensors are translated into information by different algorithms based on needs. A good example of this would be weather prediction. Barometric measures obtained from sensors placed at different geographic locations are sent to a central processing system that can predict the direction of rain and so on. Of course, the precision of this system not only depends on the processing algorithm but also the data gathered. Increasing the sensor density in the same network coverage allows for the algorithm to have more samples to base its prediction on. This is where wireless sensor networks have the upper hand over wired networks. Due to decreasing costs and smaller hardware components, it has become practical to deploy wireless sensor networks liberally in situations such as this. With the realization of its practicality, comes the possibility of using wireless sensor networks in many other situations. As with other engineering research, the availability of simulators have allowed wireless sensor network research to thrive and be expedited. Some of the more well-known general network simulators are OPNET [19] and ns2 [28]. These network simulators are considered general because both provide network components such as ethernet connections, MAC layers and so on. The user is then allowed to implement their
own protocol into the simulator. Both simulators come with wireless networking modules, allowing researchers to implement and analyze new and novel wireless sensor network protocols. Analysis via simulators before deployment in physical networks is important for many reasons, the more crucial ones being cost and complexity. By removing the uncertainty and expense of hardware components, a researcher is allowed the luxury of focusing on the underlying network algorithm. Though superior in modeling RF signals and network operations, these simulation software lack fully functional sensor modules that replicate a sensor’s interaction with the environment. The current workaround to this problem is to model sensed data as pseudo-random processes, though the validity of these approaches is questionable. Realistic and specific sensor modules do exists in different simulation environments such as MATLAB and Orcad PSPICE but very little has been done to incorporate these models into a network simulator. Hence it is the goal of this work to investigate the feasibility and benefit of developing user-configurable sensor models in a widely used network simulation software.

**Background Work**

Sensor modeling is not a new concept and has been a growing research area lately. Advancements in simulator technology and computer speeds allow for the creation of good sensor models. Of course, the main motivation to perform sensor modeling is to develop new and innovative sensors while keeping costs down. A new sensor that can be fully modeled in simulation before entering production saves a manufacturer time and money. There are an endless number of sensor models created by researchers and companies, most of which focus on electrical circuitry modeling.

However, to the best knowledge of the author, the modeling of thermal imaging sensors has been lacking. One of the few models available for reference is an infrared sensor model that is an integral part of a simulation software called SYTHER [32]. As a training simulator, SYTHER’s purpose is to generate realistic synthetic images of scenes made of landscapes and targets observed by a wide range of sensors in any IR waveband and in any
environmental condition. The infrared sensing module of this simulator divides the infrared sensing process into three distinct components: optics, scanners and detectors. The model is complex with many considerations taken into account for each component of the infrared sensing process:

- **Optics**
  The optics component simulates the workings of a physical sensor lens. There are three main features to this component. First is a geometric function that simulates image formation in the presence of geometric distortion. The optics component also includes a radiometric function to account for optical radiometric effects. Some of the effects accounted for in this function include wavelength dependent variations in optical transmittance, image edge darkening from the collection of incident radiance through the entrance pupil and vignetting due to radiation stopped by optical components different from the entrance pupil. The final function of the optics component is a blurring function which simulates optical effects which make an image point look larger, as a blur, on the detector plane.

- **Scanners**
  The purpose of the scanner module is to accurately simulate the movement of the detector array field of view in terms of the whole sensor’s field of view by moving an elementary area of the image captured over a fixed position detector.

- **Detectors**
  The detectors act as the sensor pixels. The detectors have two functions, the first of which is a geometric function. The purpose of this is to simulate detector action on irradiance reaching the image plane. The second is a radiometric function which simulates conversion of the thermal radiation into electrical signals by the detector. Included in this function are electrical noise and detector responsivity non-uniformity.

Another less comprehensive infrared sensor model is NVSIM [12]. This simulator requires as an input an image file that will be converted into a simulated thermal image.
based on user inputed sensor specifications. Taken into consideration are optics, sampling, noise as well as electrical circuitry gain.

Both these simulators, however, do not fulfill the objective of this project and that is to integrate a sensor module into a networking simulator for use by sensor networks.

Objectives

The lack of actual sensor modules in currently used network simulators leaves a huge void in the way simulation software can actually mimic real-life network events. In an attempt to fill in this void, this project’s objectives are as follows:

• **To create a usable and configurable infrared sensor module in OPNET**
  The infrared sensor module created in OPNET has black box properties whereby a user needs to only provide certain parameters and plug in the module in appropriate places in the simulation code. By providing this facility, it is hoped that the sensor module can be used without hassle by those in need of a proper infrared sensor module for their network simulation.

• **To be able to provide thermal imaging capabilities in OPNET**
  By combining the input of the infrared sensor module with certain algorithms, it is possible to provide a simple 2 dimensional false-colored image of the field of view captured by the sensor. This image is stored and visible to the end-user to understand what a node’s sensor is capable of viewing at any given time in the simulation. This information is then free to be used by individual algorithms created by the user.

• **To provide additional secondary sensor modules and an analysis of them**
  This project will also be able to provide users with additional, though simpler sensor modules that will be readily implementable in their sensor network algorithms. Such sensors include, but are not limited to, internal temperature, humidity and velocity sensors. The feasibility and limitations of simulating these sensors will also be discussed.
• **To show that the sensor modules accurately simulate the properties of a physical sensor**

The sensor modules would serve no purpose if it were to provide results that did not reflect real world implementations. Therefore, an important goal of this project is to provide proof of the accuracy of the simulation sensor module when compared to its physical counterparts.

• **To show the benefits of providing external inputs to a wireless sensor network algorithm**

The end goal of this project is to ultimately show that using these sensor modules is beneficial, in terms of simulation accuracy, to the end-user when compared to the traditional method of emulating sensor results.

• **To provide a guide on editing OPNET code**

The sensor modules created in this project will be built based upon the current structure of OPNET’s Wireless Module’s pipeline stages and other properties. A side-goal of this project is to provide a tutorial on how to edit the properties of these elements to create the proposed sensor modules.
Chapter 2

Physical Sensors

2.1 Sensor Categorization

Current technology has allowed for the development of a vast range of sensors, used to collect different types of data in a variety of ways for almost any situation. There are many different ways to categorize these sensors based on the particular sensing properties to be studied. For the purpose of this project, it is important to understand how data is gathered by a sensor as its implementation on a sensor node can vary widely depending on this factor.

The following four categories have been devised especially for nodes in sensor networks based on how data is gathered:

- **Ambient Sensors**
  Sensors in this category are typically used to sense information from the immediate surroundings of a sensor node to obtain information such as the ambient temperature, humidity, noise level and so on. By obtaining information from these sensors, a node is able to evaluate the atmospheric conditions of its surroundings. Object identification with this type of sensors is difficult if not impossible unless the external object to be detected is within very close range of the sensor and able to affect the sensor’s entire reading.

- **Self-characterizing Sensors**
Self-characterizing sensors are used to obtain information from within a node. Important applications of this type of sensors include internal heat sensing to detect if a node’s temperature reaches a critical level. Other sensors in this category include accelerometers, gyroscopes and global positioning sensors. Information from these sensors help a node understand internal characteristics that might affect its performance. Having said that, they are rarely used for primary data collection because the core purpose of sensor nodes is to collect external data from its environment.

It would be wise to note that the aforementioned sensors can only provide one data value for every sample taken. For example, a temperature sensor sampling a pack of ice placed next to hot coal will not be able to distinguish the difference between the two. It will only be able to provide a reading according to how each object influences the environment close to the sensor.

However, this is not the case for object characterizing sensors that have the ability to identify objects in its vicinity. This category is further divided according to the way data is obtained:

- **Intrusive Object Characterizing Sensors**

  Intrusive object characterizing sensors obtain data by first emitting a waveform and then analyzing the returning waveform that was echoed off of target objects. As such, sensors such as ultrasonic sensors and sonars are ‘intrusive’ or active, for they need to emit some form of energy that needs to be returned for sensing to take place.

  An example of an intrusive object characterizing sensor can be seen as in Figure 2.2\(^1\) which is an underwater side-sonar scan of a submerged bridge. The actual picture of the bridge before being submerged in water can be seen as in Figure 2.1\(^2\).

- **Non-intrusive Object Characterizing Sensors**

\(^1\)http://en.wikipedia.org/wiki/Image:Lakemurray-wyse_ferry_Bridge_sonar.jpg
The other type of object characterizing sensors are those that only read in information, such as electromagnetic waves, without having to emit a waveform beforehand. Hence, sensors in this category, including infrared temperature sensors, are categorized as ‘non-intrusive’ or passive as no form of energy needs to be emitted by the sensor for sensing to take place.

Object characterizing sensors have the capability to distinguish between different objects based on different factors such as the sensor’s resolution and the objects’ physical properties. Consequently, infrared temperature sensors can be expanded into thermal imaging sensors whereas side-scan sonars can be used to map entire regions of a seabed.

![Figure 2.1: The Wyse Bridge before being submerged.](image_url)

Of all the types of sensors depicted above, a model for non-intrusive object characterizing sensors (in specific, thermal imaging sensors) was selected to be the primary sensor module in this project for the following reasons:
The need to be informed of objects surrounding a node

Due to the lack of physical sensor modules in OPNET, sensor node implementations are not able to detect objects in its surroundings without external help, such as packets from other nodes or GPS data.

Thermal imaging sensors, along with proper algorithms, can help a node identify and distinguish objects in its surroundings in a stand-alone manner. This ability allows simulated sensor nodes to be autonomous.

Relatively new sensing technology

Thermal imaging sensor technology is relatively new outside of military applications. As such, there are many uses for it that have just recently been implemented or are currently being researched, such as production line defect detection [15], weather forecasting and so on. With this and future work, it is the hope of the author that this sensor module will open gateways to new, novel and creative ideas for wireless sensor networks to take advantage of infrared sensing capabilities.

Compatibility with OPNET’s Wireless Module

OPNET’s wireless module is based on the transmission of packets from one node to its surroundings in a unidirectional manner. It is up to receiving nodes to obtain this information, interpret it as needed and possibly generate packets of its own to be transmitted. Of all the four sensor categories mentioned above, non-intrusive
object characterizing sensors most closely resemble this trait of OPNET’s wireless module. In the context of thermal imaging sensors, infrared radiation represented by packets are broadcasted by thermal objects and detected by sensors. As the radiation is interpreted from multiple sources, thermal images of the sensor’s surroundings are created. A detailed overview of this concept is given in Section 4.1.

2.2 Background on Temperature Sensors

2.2.1 Thermocouple based Temperature Sensors

A very common type of temperature sensors are thermocouples which convert thermal potential difference into electrical potential difference to measure temperature. In 1821, Thomas Johann Seebeck [35] discovered that any conductor subjected to a thermal gradient will generate a voltage. By measuring this effect, known as the Seebeck effect, a direct correlation can be made between the change in voltage and temperature.

Because thermocouples do not measure the absolute temperature but rather the temperature difference between two points, there needs to be a known reference temperature at one of the points, called the cold junction. When doing so is not feasible, some other type of thermally sensitive device needs to be used to generate a voltage for the cold junction. This process is known as cold junction compensation.

2.2.2 Infrared Temperature Sensors

Commercial use for thermal imaging technology began with the creation of photon detection sensors that needed to be cooled by liquid nitrogen to 77 K. Unfortunately, to achieve this operating temperature, cryogenic coolers were needed along with a startup time of about 10 minutes [17]. There were issues with size, maintenance and cost that hindered these sensors from being used widely beyond research fields.

In the 1980’s, the United States Department of Defense provided Honeywell and Texas
Instruments classified contracts to develop uncooled thermal sensors with hope to obtain small form-factor sensors with very short initialization time. Research by both companies led to the development of the pyroelectric infrared sensor by Texas Instruments and the microbolometer by Honeywell.

- **Pyroelectric Infrared Sensor**
  
  [23] mentions that the technology behind this type of sensors is based upon induced pyroelectric effects near the phase transition of a ferroelectric element. These sensors consist of a thin wafer of dielectric crystal that is poled and oriented. The pyroelectric crystal acts as a capacitor that frees electric charges when exposed to a change in temperature that can then be measured and interpreted to provide temperature information. An example of a pyroelectric infrared sensor element can be seen as in Figure 2.3.

- **Microbolometer**
  
  As noted in [14], a microbolometer element consists of a two-layer structure with a grid of vanadium oxide or amorphous silicon heat sensors on top of a grid of silicon. As infrared radiation from a specific wavelength range strikes the top layer, its electrical resistance changes. This resistance change can then be measured and
interpreted to provide temperature information. An example of a microbolometer element is shown as in Figure 2.4.

![Figure 2.4: Microbolometer element [14].](image)

Both sensor types have their advantages and disadvantages while also being better suited for certain types of applications. More information on these sensors can be obtained from Infrared Solutions [14], C. A. Marshall et al [29], C. C. Chang et al [3], C. Hanson and H. Beratan [10] and A. Hossain and M. Rashid [13].

By 1992, the US government declassified the use of these products for commercial use, although under tight restrictions. From that point on, the development of thermal sensors for different commercial uses has advanced tremendously.

Infrared sensors used for temperature detection have many advantages over traditional thermocouples, including:

- **The ability to measure temperature from afar**
  
  Infrared temperature sensors do not need to have direct contact with the object being measured in order to provide a reading. As long as there is an unobstructed path for infrared radiation from the target object to reach the sensor, a reading can be obtained. This is different from traditional thermocouple based temperature sensors that need to be in close proximity with the target object to provide a reading that is accurate.
• **Higher resolution**

One of the major limitations of traditional thermocouple based temperature sensors is their accuracy which is typically more than ± 1°C [37] due to factors such as cold junction compensation. On the other hand, infrared temperature sensors can provide an accuracy of around ± 0.1°C, a tenfold improvement.

• **Fast temperature readout**

Once initialized, infrared temperature sensors are able to provide temperature readings almost instantaneously as they are based on detecting infrared radiation at a constant wavelength. Thermocouples, however, need to ensure that the sensor inputs have stabilized before providing temperature information.

### 2.3 Background on Infrared Radiation

Temperature is defined as a number that is related to the average kinetic energy of the molecules of a substance. When measured in Kelvin degrees, the number is directly proportional to the average kinetic energy of the molecules. Heat, on the other hand, is used to symbolize the amount of energy transferred from one body to another because of temperature differences between the two. The symbol for heat is $Q$ with a SI unit of joule. Two things may happen when heat goes into a substance:

- **A raise in temperature:**

  An influx of heat can speed up the molecules of a substance, thereby increasing its kinetic energy. As Kelvin temperature is proportional to the average kinetic energy of the molecules of a substance, a factor increase in kinetic energy will cause the same factor increase in temperature.

- **A change of state:**

  When not increasing kinetic energy, an influx of heat causes a change in potential energy. This happens when ice melts into water. The temperatures of the ice moments
before melting and water moments after being formed are equal because of constant kinetic energy.

There are three ways for heat to move from one place to another, as mentioned in [2]:

- **Conduction:**
  Conduction occurs when molecules crash into each other. Heat is conducted from one side of a material to another if there is a difference in temperature.

- **Convection:**
  Convection is the transfer of energy by currents within a fluid, including gas. Convection occurs in the atmosphere and ocean where energy is transferred from hotter areas to colder ones.

- **Radiation:**
  Any object above 0 K emits some type of electromagnetic radiation from its surface due to its temperature. This type of radiation includes both infrared and visible light.

Only heat transferred as radiation will be of importance to this project as infrared sensors use this property to interpret an object’s temperature.

### 2.3.1 Electromagnetic Radiation

James Clerk Maxwell [36], the formulator of Maxwell’s equations, described electromagnetic radiation as a self-propagating wave in space with electric and magnetic components. These components oscillate at right angles to each other and to the direction of propagation, while being in phase with each other. Electromagnetic radiation is classified into types according to its wavelength.

In the order of decreasing wavelength, these groups are:

- **Radio Waves**
  Radio waves are primarily used for the transmission of data by radio antennas. The
wavelength of radio waves can range from hundreds of meters to about one millimeter. Mobile phones as well as wireless networks use radio waves to transmit data.

- **Microwaves**
  Microwave is a generic name given to waves that have wavelengths between that of radio and infrared radiation. This range includes ultra-high frequency (UHF), super high frequency (SHF) and extremely high frequency (EHF) signals.
  There are many uses for microwave signals including: microwave ovens, telecommunication (WiMAX) and also radar technology.

- **Infrared Radiation**
  Infrared radiation encompasses electromagnetic radiation with wavelengths between that of microwaves and visible light. Infrared radiation spans 3 orders of magnitude in wavelengths that can range from approximately 1 mm to 750 nm.

- **Visible Light**
  Visible light is defined as the portion of the electromagnetic spectrum that is visible to the human eye. As with the other spectrums, there are no exact boundaries to the wavelengths of visible light although the human eye can typically respond to wavelengths from 700 nm to 400 nm.

- **Ultraviolet Radiation**
  Ultraviolet radiation encompasses wavelengths shorter than that of visible light but larger than x-ray. This radiation is sometimes referred to as black light as it is invisible to the human eye but visible to some animals. The many uses of ultraviolet light include sterilization in the medical field, mineral analysis and as black light for security purposes (for example, monetary bills containing UV sensitive ink that can only be seen under black light).

- **X-Rays**
  The term x-ray refers to the region of the electromagnetic spectrum with wavelengths
in the range of 10 nm to 0.01 nm. The primary use of X-rays are for diagnostic radiography and crystallography.

• **Gamma Rays**

Gamma rays are electromagnetic radiation produced by radioactive decay or other nuclear or subatomic processes such as electron-positron annihilation.

Because of its powerful nature, gamma rays are used in the medical industry to sterilize medical equipment by killing bacteria. Gamma rays are also used to treat some types of cancer in a medical procedure called the gamma-knife surgery.

Figure 2.5 shows a summary of the electromagnetic spectrum groups.

![Electromagnetic Spectrum](image)

Figure 2.5: Groups of the electromagnetic spectrum divided by wavelength [5].

### 2.3.2 Heat - Electromagnetic Radiation Relationship

All elements that emit heat also emit some amount of electromagnetic radiation. The intensity of the radiation as a function of the wavelength and temperature can be plotted and
is called Planck’s Radiation Curve. Figure 2.6 shows the radiation curve of three different stars; Spica, the Sun and Antares, as obtained from [7].

![Heat Radiation Curves of 3 Stars](image)

Figure 2.6: Example heat radiation curves of 3 stars [7].

The intensity of the radiation at any given wavelength depends on the object’s temperature and emissivity. The emissivity of an object is a measure of the object’s ability to absorb and radiate energy. It is measured as the ratio of the energy radiated by the object when compared to an ideal black body at a given temperature and wavelength. Because it is a ratio, the value of emissivity can only be within the range of 0.0 to 1.0. An example emissivity value is that of concrete which is 0.94 meaning that at a given temperature, concrete is able to emit 94% of the radiation when compared to a black body at the same temperature. Table 2.1 shows the emissivity values of a few other common materials at a given temperature. As can be seen, the values vary widely depending on the material itself.

The emissivity of a target object plays a huge role in the accuracy of a sensor, as will be explained further on. [34] suggests that numerous factors affect the emissivity value of a material:

- **The actual composition of the object’s surface:**

Thin coatings such as oxidation of the object’s surface can affect its emissivity value.
<table>
<thead>
<tr>
<th>Material</th>
<th>Reference Temperature (°F)</th>
<th>Emissivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concrete</td>
<td>32-2000</td>
<td>0.94</td>
</tr>
<tr>
<td>Mercury</td>
<td>32</td>
<td>0.09</td>
</tr>
<tr>
<td>Sand</td>
<td>68</td>
<td>0.76</td>
</tr>
<tr>
<td>Water</td>
<td>100</td>
<td>0.67</td>
</tr>
<tr>
<td>Gravel</td>
<td>100</td>
<td>0.28</td>
</tr>
<tr>
<td>Ice, smooth</td>
<td>32</td>
<td>0.97</td>
</tr>
<tr>
<td>Ice, rough</td>
<td>32</td>
<td>0.98</td>
</tr>
</tbody>
</table>

Table 2.1: Common emissivity values [18].

- **The roughness of the object’s surface:**
  A rougher surface will have a larger emitting area and this could be attributed to a change in emissivity.

- **The geometric shape of an object:**
  For example, emissivity values drop for convex surfaces with increased curvature because they spread their radiation over a wider angle. The opposite holds true for concave surfaces.

- **The viewing angle from normal:**
  Reflective surfaces do not emit radiation equally at all angles. Therefore emissivity values for objects composed of these materials drop rapidly at any angle beyond 45°.

- **Foreign material in the line of view between the target object and the sensor:**
  Rain, smoke, steam and so on absorb radiation and thus lessen the amount of radiation received by the sensor.

Max Planck [30] stated that a black body is an entity that absorbs all electromagnetic radiation that falls onto it. External radiation can neither pass through nor reflect off a black body. At the same time, a black body can theoretically radiate every possible wavelength of energy. The amount and type of electromagnetic radiation emitted is in direct relation with its temperature. The temperature is also related to a black body’s color. As mentioned
in [8], black bodies at a temperature below around 700 K (426.85 °C) produce very little radiation at visible wavelengths and appear black. However, these entities begin to produce radiation at visible wavelengths starting with red, going through orange, yellow, and white before ending up as blue as the temperature increases.

Black bodies are theoretical elements but are replicated in labs by creating a small hole entrance to a larger cavity. Light entering the hole is trapped because it would have to reflect off the walls of the cavity multiple times and is thus absorbed before managing to escape. If this cavity is then heated, the spectrum of the hole’s radiation will be continuous and will not depend on the wall material in the cavity.

The law governing this relationship is called Planck’s Law. Although it can be expressed in terms of the radiation frequency or wavelength, only the formula relating to the wavelength will be used for our purposes. The equation, as obtained from [26], is as follows:

\[ S_{\lambda} = \frac{E^2\pi^2h}{\lambda^5} \frac{1}{e^{\frac{h\lambda}{kT}} - 1} \]

The definition of the variables are as in Table 2.2:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
<th>SI Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_{\lambda} )</td>
<td>Radiation Spectral Intensity (energy per unit time per unit surface area per unit solid angle per unit frequency)</td>
<td>( J\cdot s^{-1}\cdot m^{-2}\cdot sr^{-1}\cdot Hz^{-1} )</td>
</tr>
<tr>
<td>( h )</td>
<td>Planck’s constant</td>
<td>( 6.6260693 \times 10^{-34} ) J·s</td>
</tr>
<tr>
<td>( c )</td>
<td>Speed of light</td>
<td>( m\cdot s^{-1} )</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>Wavelength</td>
<td>( m )</td>
</tr>
<tr>
<td>( k )</td>
<td>Boltzmann constant</td>
<td>( 1.3806505 \times 10^{-23} ) J·kelvin(^{-1} )</td>
</tr>
<tr>
<td>( T )</td>
<td>Temperature</td>
<td>kelvin</td>
</tr>
<tr>
<td>( E )</td>
<td>Emissivity</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2.2: Definition of variables for Planck’s Law.

From Planck’s Law and Planck’s Radiation Curve, two characteristics can be observed as a black body’s temperature increases:
• The peak intensity of the radiation increases

• The wavelength at which the peak is produced decreases

Both these characteristic can be observed from Figures 2.7(a) and 2.7(b) below where the temperature of a 6000 K black body is compared to that of a 3000 K black body. It should be noted that the wavelength can vary throughout the electromagnetic spectrum based on the object’s temperature. Table 2.3 summarizes the black body temperatures necessary to give a peak for emitted radiation in various regions of the spectrum. These ranges are approximate as there are no exact bounds between the different types of electromagnetic waves as explained previously.

<table>
<thead>
<tr>
<th>Region</th>
<th>Wavelength (cm)</th>
<th>Black body Temperature (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radio</td>
<td>&gt;10</td>
<td>&lt;0.03</td>
</tr>
<tr>
<td>Microwave</td>
<td>10 - 0.01</td>
<td>0.03 - 30</td>
</tr>
<tr>
<td>Infrared</td>
<td>0.01 - 7 × 10^{-6}</td>
<td>30 - 4100</td>
</tr>
<tr>
<td>Visible</td>
<td>7 × 10^{-9} - 4 × 10^{-9}</td>
<td>4100 - 7300</td>
</tr>
<tr>
<td>Ultraviolet</td>
<td>4 × 10^{-5} - 10^{-7}</td>
<td>7300 - 3 × 10^{6}</td>
</tr>
<tr>
<td>X-Ray</td>
<td>10^{-7} - 10^{-9}</td>
<td>3 × 10^{6} - 3 × 10^{8}</td>
</tr>
<tr>
<td>Gamma Rays</td>
<td>&lt; 10^{-9}</td>
<td>&gt; 3 × 10^{8}</td>
</tr>
</tbody>
</table>

Table 2.3: Black body temperature ranges and corresponding wavelength ranges [7].

Another law that governs black body radiation is Wien’s Law of Displacement. As mentioned in [27], this law states that the relationship between the temperature \( T \) of a black body, and wavelength \( \lambda_{max} \) at which the intensity of the radiation it produces is at a maximum is:

\[
\lambda_{max} = \frac{b}{T}
\]

Whereby \( b \) is Wien’s displacement constant with a value of \( 2.897 \ 7685 \times 10^{3} \) m K. From this relationship, it can be seen that as a black body’s temperature increases, the position of \( \lambda_{max} \) decreases. Another fact to note from Wien’s Law is that the displacement under all radiation curves is always the constant \( b \).
Figure 2.7: Black body curves for two different temperatures [26].
Finally, the third law related to black bodies is Stefan-Boltzmann’s Law which states that the total energy radiated per unit area per unit time $j^\ast$ (in watts per square meter) by a black body is related to its temperature $T$ (in Kelvins) and the Stefan-Boltzmann constant $\sigma$ as follows [24]:

$$j^\ast = \sigma T^4$$

Whereby $\sigma$ has a value of $5.6704 \times 10^{-8}$ watt m$^{-2}$ K$^{-4}$.

In short, taking the derivative of Planck’s Law to find the peak of the radiation curve produces Wien’s Law of Displacement. On the other hand, integrating Planck’s Law over all wavelengths produces Stefan-Boltzmann Law to obtain the total power radiated by the black body. This is depicted in Figure 2.8, which is an adaptation from [4].

![Figure 2.8: Image depicting how Wien’s Law of Displacement and Stefan-Boltzmann Law are derived from Planck’s Radiation Curve [4].](image)

**2.3.3 Infrared Temperature Sensor Technology**

As mentioned in the previous section, the radiation emitted from a black body is only in the infrared region for temperatures ranging from 30 K to 4100 K. This roughly translates to
-243 °C to 3827 °C. Temperatures beyond this range cannot be detected by infrared sensors as the Planck’s Radiation Curve would have shifted beyond the scope of the infrared range. However, this is not an issue as for all practical purposes, the detectable temperature range is adequate.

[38] mentions that infrared temperature sensors are calibrated in laboratories using a lab-constructed black body as a reference point. When being calibrated, the sensor assumes an emissivity level of 1.0, in tune with the ‘perfect’ black body it is being calibrated to. The precision of the sensor depends on the effectiveness of the simulated black body.

An infrared temperature sensor works on the basis of Planck’s Law, which contains three variables; \( S_\lambda, \lambda \) and \( T \). By obtaining the values of two of the variables, the numerical value for the third one can be calculated. There are generally two ways for infrared temperature sensors to collect data, both applying the same fundamental concepts but applied slightly differently. They are:

- **Single Wavelength Infrared Temperature Sensors**
  A single wavelength infrared temperature sensor detects infrared radiation only at a certain wavelength. Then, because of the calibration performed in lab, a temperature readout can be obtained for different radiation intensities detected by the sensor.

- **Dual Wavelength (Ratio) Infrared Temperature Sensors**
  A dual wavelength infrared temperature sensor measures the radiation intensity of the incoming infrared radiation at two different wavelengths instead of just one. The ratio of the intensities calculated are then calibrated against a lab black body. As the ratio changes, so does the calculated temperature of this type of sensor.

For both temperature sensors, the ability to accurately measure the temperature of objects are closely related to how accurately the object’s emissivity is known. An object with an emissivity of 0.7 emits only 70 percent of the energy of a black body. Therefore, unless this is taken into account, the indicated temperature reading will be lower than the object’s actual temperature for a sensor calibrated to an emissivity of 1.0.
There are quite a few advantages to using a dual wavelength infrared temperature sensor when compared to a single wavelength one, including:

- **Less sensitive to emissivity variations**
  Because ratio infrared temperature sensors calibrate their signal ratio against temperature, the temperature obtained also depends on the ratio of emissivity at the two different wavelengths. This allows for ratio temperature sensors to be less sensitive to emissivity variations. Single wavelength infrared temperature sensors, on the other hand, calibrate their internal measurement signal level directly to temperature and are highly influenced by emissivity levels. Because of this advantage, ratio infrared sensors are preferred when measuring temperatures of materials with frequently changing emissivity levels.

- **Less sensitive to obstructed infrared radiation**
  A ratio infrared temperature sensor is less susceptible to physical obstructions that might block incoming infrared radiation. If some physical structure partially obscures the sensor’s field of view by some fraction, both detector signals are reduced by the same fractional amount, leaving the ratio unchanged and therefore not influencing the final temperature reading.

  Using the same reasoning, ratio infrared temperature sensors are less sensitive to objects smaller than the full field of view, atmospheric effects such as smoke, haze or dust obscuring the object, or dirt contaminating the sensor’s front lens; assuming such factors influence the readings at both wavelengths equally.

**2.3.4 Thermal Imaging Sensors**

When applied to thermal imaging sensors, each pixel of the thermal imaging camera is represented by a separate infrared sensor element. Combined with proper signal processing and algorithms, a final thermal image can be produced. The optics involved in thermal
imaging is very similar to that of CCD digital cameras. Figure 2.4 shows a part of a microbolometer array that is used as a thermal imaging sensor.

Figure 2.9: Image of a section of a microbolometer array [14].
Chapter 3

OPNET Simulation Software

OPNET’s user manual gives the perfect overview of its capabilities, stating that “OPNET provides a comprehensive development environment supporting the modeling of communication networks and distributed systems. Both behavior and performance of modeled systems can be analyzed by performing discrete event simulations. The OPNET environment incorporates tools for all phases of a study, including model design, simulation, data collection, and data analysis.” [19].

Some of the key system features of OPNET are:

• **Object orientated programming**
  All systems created in OPNET consist of objects with configurable attributes. These objects belong to classes that can be derived from other classes to provide more specific support for particular applications.

• **Specialized in communication networks and information systems**
  By providing constructs related to communications and information processing, OPNET is able to provide high leverage of networks and distributed systems.

• **Hierarchical models**
  Because OPNET enforces a hierarchical order, it naturally parallels the structure of actual communication networks.

• **Graphical specification**
Most OPNET models can be created using graphical editors to allow for intuitive mapping from the modeled system to the OPNET model specification.

- **Flexibility to develop detailed custom models**
  OPNET’s programming language is a high-level hybrid language called ‘Proto-C’. By providing this language with extensive support for communications and distributed systems, the simulation environment allows realistic modeling of all communications protocols, algorithms and transmission technologies.

- **Automatic generation of simulations**
  Almost all code and specifications implemented in OPNET are compiled automatically into executable discrete-event simulations implemented in the C programming language, letting the user focus on implementing the simulation and obtaining the required results.

- **Application-specific statistics**
  Built-in performance statistics along with user specified application statistics provide a robust way for the user to collect data on the simulation taking place.

- **Integrated post-simulation analysis tools**
  Simulation output and statistic data can be represented graphically using tools provided by OPNET.

- **Interactive analysis**
  The included interactive debugger program allows for analysis of all OPNET simulations. With this tool, a user is able to debug code thoroughly and systematically.

- **Animation**
  OPNET provides a way for simulation runs to automatically generate animations of the modeled systems at various levels of detail including animation of statistics.

- **Cosimulation**
  Models created in other simulators such as MATLAB can be connected to models in
OPNET. These external models can represent network hardware, end-user behavior patterns and so on.

- **Application Program Interface (API)**
  By providing APIs for most of OPNET’s functions as an alternative to graphical specifications, automatic generation of models in user code is vastly simplified.

Almost all of these features will be used in the creation of the sensor modules for this project.

### 3.1 OPNET Architecture

OPNET’s architecture divides the simulation into three phases that form a cycle as in Figure 3.1:

![Simulation Project Cycle](image)

Figure 3.1: Simulation Project Cycle [19].

Each of these phases will be explained in detail below:

#### 3.1.1 Model Specification

Essentially, model specification involves the task of representing a system to be studied in OPNET. To better resemble an actual network, model specification is divided into a few domains:
Network Domain

The network domain defines the topology of a communications network. Entities in this domain are called nodes that are created using a node editor. Because of OPNET’s object orientated programming characteristics, different nodes may be based on the same node model but have unique features based on attributes set individually in each node.

Three different types of nodes are available in OPNET based on mobility traits. Fixed nodes are the default type of nodes offered in OPNET where the location of a node in the topology may not change once a simulation has begun. Mobile nodes can be assigned pre-defined trajectories that specify their position as a function of time throughout a simulation run. In the same manner, satellite nodes are assigned orbits that prescribe their motion. Both mobile and satellite nodes are only available with the optional Wireless Module installed in OPNET.

Communication with other nodes is vital for a node to complete its function in a network model. OPNET thrives in this aspect as it provides numerous ways for nodes to exchange information, including simplex and duplex point to point links, bus links as well as the capability for fixed, mobile and satellite nodes to communicate with each other using radio links with the installation of the Wireless Module.

Large networks may use a form of abstraction called a subnetwork to simplify network protocols and addressing. A subnetwork is a subset of a larger network’s devices that forms a network in its own right. This abstraction can be formed at an infinite amount of levels, where a network can be a subnetwork of a larger network while also being composed of smaller subnetworks.

Node Domain

The node domain provides for the modeling of nodes deployed in the network domain as mentioned previously. Nodes may correspond to various types of computing and communicating devices in the real world such as routers, bridges, terminals, file servers and in the case of this project, wireless sensor nodes.
Edited in the node editor, components in the node model are called modules; with some, such as transmitters and receivers offering a limited predefined amount of user changeable parameters and others such as processors, queues and external systems being highly programmable. The highly configurable modules are edited in the process editor.

Interactions between modules are provided by three different types of connections. Packet streams allow formatted messages called packets to be sent from one module to another. Statistic wires are used when one module needs to monitor the performance or state of another module. Finally, logical associations identify a binding between modules. These connections are shown as in Figure 3.2:

![Figure 3.2: Different types of connections used in a node model [19].](image)

**Process Domain**

User-programable elements in the node domain are processes that are edited using the process editor. These processes are similar to an executing software program.

A processor may contain multiple processes that execute within it. At the beginning of a simulation, only the root process executes and this process can then create new processes which in turn can create others.

In a manner similar to the procedure-call mechanism in a programming language, only
one process can be executed at any time. A process that is executing can invoke another
process which will then suspend the invoking process until it (the invoked process) has
blocked. Processes also respond to interrupts, which indicate the occurrence of events of
interest. A process that is provoked by an interrupt takes action in response to the interrupt
and then blocks.

The programming language used to program processes is called Proto-C. This is a
hybrid programming language combining graphical state-transition-diagrams, embedded
C/C++ language data items and statements, and a library of Kernel Procedures that provides
commonly needed functionality for modeling communications and information processing
systems.

External System Domain

When OPNET runs a cosimulation, its interface with the external code is an external system
definition (ESD), represented in the node model by an esys module. ESDs use esys inter-
faces in a black-box manner to send and receive data when communicating with external
code.

3.1.2 Modeling Communications with Packets

The most commonly used mechanism for information exchange is called a packet. Pack-
ets contain formatted information that can change dynamically while being stored by and
transferred between objects in each of the modeling domains mentioned previously. Each
packet contains three main areas: user-defined fields, pre-defined fields for accounting and
transmission data used by link models. OPNET’s structure allows each packet to be mod-
eled individually and have arbitrary contents in its fields. Data types that are supported by
the fields include numerical values, encapsulated packets as well as general data structures.
### 3.1.3 Data Collection and Simulation

Data collection and simulation are vital parts of a modeling effort as it allows one to obtain measures of a system’s performance and also make observations concerning a system’s behavior. Although customizable, most users will find the output automatically generated by OPNET simulations sufficient. These outputs can be divided into the following three categories:

- **Output Vectors**
  
  Output vectors represent time-series simulation data. They consist of a list of entries, each of which is a time-value pair.

- **Output Scalars**
  
  Scalar statistics are individual values that represent a metric of interest. They are often derived from vector statistics, as an average, peak value, final value, or other statistic. Typically, one value for each scalar statistic is recorded during a simulation; when many simulation instances are run, their scalar outputs are combined to form a graph.

- **Animations**
  
  OPNET simulations can generate animations that can be viewed during the run or afterwards. Several forms of predefined or automatic animations are provided including packet flows, node movements, state transitions and statistics.

### 3.1.4 Analysis

The final phase of simulation is used to examine the results collected during simulation. The Analysis Tool in OPNET allows the data collected to be represented in forms of graphs. These graphs are plotted in analysis panels that can be saved and recalled as analysis configurations for future references.
3.2 Wireless Module

The optional Wireless Module in OPNET plays a vital role in the completion of this project and will be reviewed extensively in this section. This module allows for the simulation of networks that include moving nodes as well as radio communications.

3.2.1 Wireless Objects in the Network Domain

There are four basic types of nodes provided by OPNET in the wireless module:

- **Mobile subnetwork**
  Similar to a fixed subnetwork, except that it can move as specified by a user-defined trajectory.

- **Satellite subnetwork**
  Similar to a fixed subnetwork, except that it moves automatically on an assigned orbit.

- **Mobile node**
  Similar to a fixed node, except that it can move as specified by a user-defined trajectory.

- **Satellite node**
  Similar to a fixed node, except that it moves automatically on an assigned orbit.

**Trajectories**

A trajectory is a path specification for a mobile site’s motion during a simulation. In OPNET, a trajectory can be defined as either segment-based or vector-based. Segment-based trajectories define movement using a series of pre-defined points whereas vector-based trajectories define movement in terms of bearing, ground speed and ascent rate.
Orbits

Satellite sites use an orbit to specify their location during a simulation. An orbit is the path around the earth along which the satellite site moves during simulation. If a subnetwork contains a satellite site with an assigned orbit, the parent subnetwork’s location and size do not affect the satellite’s orbital path. This is different from fixed and mobile sites, whose positions are often defined relative to their parent subnetworks.

3.2.2 Wireless Objects in the Node Domain

The wireless objects in the node domain allow a user to create wireless networks by supplying a node with fundamental wireless networking hardware components. By connecting user created queues and processes appropriately with these modules, the basis for wireless communication is formed. These modules are:

Radio Transmitter Module

The purpose of radio transmitter objects are to act as exit points of a node for packets transmitted on radio links. Each of the transmitter’s multiple possible channel objects attempt to access appropriate radio receiver channels in remote nodes using radio links. Transmitters can also have multiple input streams, corresponding to a separate radio channel and each connected to a separate packet stream originating at other modules in the node. Due to the fact that radio transmitters are considered as data sinks from the perspective of other modules within a node, they cannot have outgoing packet streams.

Radio Receiver Module

Radio receiver modules play the opposite role of radio transmitter modules because they act as entry points in a node for packets received on radio communication links. As with transmitters, they can contain multiple channel objects and have multiple output streams, each one corresponding to a separate radio channel and each connected to another module
in the node. Also, due to the fact that radio receivers are considered to be data sources from the perspective of other modules within a node, they cannot have incoming packet streams.

**Antenna Module**

The antenna module is used to simulate an actual antenna connected to a node. Because its properties are specified externally to the transmitter or receiver modules, an antenna module may be used by multiple transmitter and receiver modules in the same node and even used in other nodes. Radio transmitters and receivers not connected to an antenna module automatically default to an antenna with equal gain in all directions.

Only connections from transmitter and receiver modules are allowed to an antenna module in the form of a packet stream. The sharing of the same antenna module by multiple transmitters and receivers as mentioned above is depicted in Figure 3.3:

![Figure 3.3: Multiple transmitters and receivers sharing the same antenna module (a_0) [19.]](image)

The effect of the antenna module on radio transmission and reception of packets is determined by two factors: the antenna’s directional gain pattern and its target (the point in space where the antenna is to be directed). The pattern of the antenna is edited using the provided Antenna Pattern Editor whereas the target is specified by the target latitude,
longitude and altitude. These attributes can be dynamically changed during a simulation run.

The point of the antenna pattern being aimed at the target is specified by two final attributes; the pointing reference phi and pointing reference theta.

An antenna pattern depicts a three-dimensional object whose shape indicates the relative magnitudes of gain in each direction. The simplest form of an antenna pattern (and also, the default in OPNET) is the isotropic pattern which radiates power equally in all directions with a gain equal to 0 dB at all points. Logically, such a pattern would have a spherical shape as depicted in Figure 3.4:

![Isotropic Antenna Pattern](image)

Figure 3.4: Isotropic antenna pattern [19].

Besides the isotropic pattern, there are many types of directional antenna patterns that can be thought of as deformations of the sphere which represents an isotropic pattern. As shown in Figure 3.5, regions of the pattern that are relative maxima of the gain function are called lobes whereas other areas can be compressed toward the center of the sphere to indicate a gain of less than unity. A gain that has reached a relative minimum is called the antenna null.
An antenna pattern can be represented as a function of two angle variables called $\phi$ and $\theta$. As in Figure 3.6, the angle of $\phi$ varies from $0^\circ$ to $180^\circ$ whereas $\theta$ varies from $0^\circ$ to $360^\circ$. Together, all directions in terms of vectors departing from the center of the pattern can be specified.
3.2.3 Wireless Pipeline Stages

The wireless pipeline stages form the backbone of the Wireless Module. The 14 pipeline stages are used step by step to determine if a packet from a transmitting node can reach any other given node in the simulation environment based on numerous factors such as distance, interference, signal match and so on.

The execution of the pipeline stages are done by different components: Stage 0 pre-sim by the simulator, Stage 1 during simulation by the radio transmitter and Stages 2 to 13 during simulation by the radio receiver. Figure 3.7 shows all the components of the wireless pipeline. A detailed explanation of each pipeline stage follows:

Pipeline Stage 0: Receiver Group

This pipeline stage is executed pre-sim time and not dynamically. The purpose of this stage is to populate each transmitter’s Receiver Group list. This list contains the receiver nodes that have the possibility of receiving transmission from a particular transmitting node. If OPNET decides that physical transmission will never be possible between two nodes, the receiving node is not added to this list. Later on in the simulation, only receiving nodes in a transmitter’s Receiver Group will execute its pipeline stages during a transmission.

In uncertain conditions that might change dynamically, (for example, a receiving node that is initially too far from the transmitting node, but moves dynamically closer to the transmitting node during simulation), OPNET takes the safe route and includes the receiver in the transmitter’s Receiver Group list.

The reason for executing this stage pre-sim is to allow OPNET to remove unneeded pipeline stage calculations from the simulation and therefore reduce the amount of redundant calculations done during simulation.

Pipeline Stage 1: Transmission Delay

This is the first stage in the wireless pipeline to execute dynamically. The purpose of this stage is to calculate the amount of time it takes for a packet to complete transmission; it
Figure 3.7: The stages that encompass the wireless pipeline [19].
is calculated as the time difference between the transmission of the first and last bit of the packet.

OPNET’s simulation kernel uses the result of this stage to signal an end of transmission event to the transmitter channel that is used to send the packet. With this event, the channel is free to send the next packet in its queue. The result is also used in conjunction with the propagation delay to calculate the time needed for the packet to reach its destination.

The result of this stage is only calculated once per packet in the transmitting node as it is obtained regardless of external variables (such as receiving node distance, speed of light and so on).

**Pipeline Stage 2: Link Closure**

Pipeline stages from this point forward are calculated at the receiving node, as their results vary dynamically, depending on the topological locations of the transmitting and receiving nodes.

The ability of a transmission to reach a receiver is referred to as closure by OPNET. The only purpose of this stage is to check if a transmitting signal can affect a receiver physically in any way, regardless of the signal and receiver attributes. The computations of this stage are mostly based on physical conditions such as node distances and signal occlusion by the surface of the earth or other objects. If link closure for a particular transmitter-receiver pair is not possible, then the computation of pipeline stages for that particular packet is halted from this point forth to reduce redundant calculations.

**Pipeline Stage 3: Channel Match**

After validation of link closure, the simulation checks the transmitting signal for channel match with the receiver. Three distinct outcomes are possible from this stage, based on channel attributes such as bandwidth and upper and lower frequencies:

- **Valid**

  Only packets that are considered compatible with the receiver channel are put in this
category.

- **Noise**
  
Packets that are not compatible with the receiver channel but still able to affect the channel are put in this category.

- **Ignored**
  
Packets that do not fit in either of the above two categories are ignored and discarded as they would no longer have an effect on calculations of further pipeline stages.

**Pipeline Stage 4: Transmitter Antenna Gain**

Pipeline stage 4 is used to calculate the transmitter antenna gain as seen from the receiving node. The calculation is performed based on the direction of the vector between the receiving and transmitting nodes and the antenna gain pattern used by the transmitting node. The value obtained from this stage will be used in other stages later on in the pipeline.

**Pipeline Stage 5: Propagation Delay**

The purpose of this pipeline stage is to calculate the time difference between the end of transmission and the beginning of reception of a packet from a transmitter to a receiver. Factors involved in the calculation include: the length of the packet, the distance between the transmitter-receiver pair and the speed of light.

**Pipeline Stage 6: Receiver Antenna Gain**

The computation of the receiver antenna gain is identical to that of stage 4 but done on the end of the receiving node’s antenna instead.

**Pipeline Stage 7: Receiver Power**

The purpose of stage 7 is to calculate the receiver power of an incoming packet. The information obtained from this calculation can be used later on in the Signal-to-Noise ratio
calculations. Whereas a transmitting antenna’s transmit power can always be set to a pre-defined value, the receive power of an antenna will always vary depending many factors. The calculation is based on the formula below:

\[ P_R = \frac{P_T \cdot G_T \cdot G_R \cdot \lambda^2}{4 \pi^2 d^2} \]

Whereby:

- \( P_R \): Receive Power.
- \( P_T \): Transmit Power; obtained from the transmitting node’s configuration.
- \( G_T \): Transmitter Gain; obtained from pipeline stage 4.
- \( G_R \): Receiver Gain; obtained from pipeline stage 6.
- \( \lambda \): Distance travelled by the signal in one period.
- \( d \): distance between the two nodes.

Pipeline Stage 8: Background Noise

The purpose of this stage is to represent the effect of all noise sources except for other concurrently arriving transmissions. Examples of noise include thermal noise, emissions from neighboring electronics and noise from alien devices.

Pipeline Stage 9: Interference Noise

The purpose of this stage is to calculate the noise accumulated from other valid or invalid packets on a valid packet that is currently being received. OPNET maintains a noise accumulator variable in each valid packet that accumulates the receive power of incoming interfering packets. When a packet completes transmission, its receive power is subtracted from accumulators of all other packets still being received so that the final value in the accumulator represents the current noise level. In this stage, OPNET also calculates the number of collisions experienced by an incoming packet.
Pipeline Stage 10: Signal-To-Noise Ratio

Pipeline stage 10 is executed to compute the current power signal-to-noise ratio result for an arriving packet. The calculation is performed based on values calculated from pipeline stages 7, 8 and 9.

Pipeline Stage 11: Bit Error Rate

Pipeline stage 11 calculates the received packet’s bit error rate based on the signal-to-noise ratio obtained from the previous stage.

Pipeline Stage 12: Error Allocation

The purpose of the error allocation stage is to estimate the number of bit errors in a packet segment where the bit error probability has been calculated and is constant.

Pipeline Stage 13: Error Correction

The final pipeline stage is used to determine whether or not the arriving packet can be accepted and forwarded via the channel’s corresponding output stream to one of the receiver’s neighboring modules in the destination node. This is usually dependent upon a few factors, including if a packet has experienced collisions, the result computed in the error allocation stage, and the ability of the receiver to correct the errors affecting the packet (hence the name of the stage). Based on the outcome of this stage, the simulation kernel will either destroy the packet, or allow it to proceed into the destination node. In addition, this result affects error and throughput statistics collected for the receiver channel.

Once a packet has been successfully processed into the receiver node, a user is free to use, manipulate and extract data from the packet in user defined processes and queues.
Chapter 4

Thermal Imaging Module

4.1 Overview

The heart of this project is the thermal imaging sensor module that allows wireless sensor nodes to create thermal images of heat radiating objects around it. This module contains four components in the network domain, and they are stated as in Table 4.1. The purpose and workings of each element will be discussed thoroughly in sections to follow.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌫️</td>
<td>Thermal Imaging Sensor</td>
<td>Nodes that capture thermal images of their surroundings</td>
</tr>
<tr>
<td>🌡️</td>
<td>Infrared Emitting Object</td>
<td>Nodes that emit infrared radiation in all directions</td>
</tr>
<tr>
<td>🔔</td>
<td>Sensor Pointing Marker</td>
<td>Used as pointing directions for sensors</td>
</tr>
<tr>
<td>🌄</td>
<td>Network Ambient Temperature</td>
<td>Nodes that allow the user to define the network area’s ambient temperature</td>
</tr>
</tbody>
</table>

Table 4.1: Network Domain components of the Thermal Imaging Sensor Module.
4.2 Packet Format

As previously explained, the thermal imaging sensor module implemented in this project is based on the structure of OPNET’s wireless module. As such, the medium of transmission for infrared radiation are packets. This section explains the structure of the packets that are created and broadcasted by the infrared emitting objects. The basic format for the infrared packet is shown in Figure 4.1. There are six components to an infrared packet:

- **infrared source**
  This attribute is used by the transmitting infrared heat radiating object to store its node object ID. This ID will be used later on by the receiving thermal imaging sensor module.

- **infrared heat**
  The transmitting object inserts its current temperature into this field. This value is set in the SI unit °Celcius.

- **infrared emissivity**
  As mentioned in Section 2.3.2, the emissivity of an object’s material plays a vital role in how accurately a thermal imaging sensor can determine the object’s temperature. For future calculations in the thermal imaging sensor module, the transmitting infrared heat radiating object inserts its emissivity value into this portion of the packet.

- **sphere_rad**
  In this project, the shapes of the infrared heat radiating objects are limited to spheres. This portion of the packet is used to store the sphere’s radius, in meters, that will be used in determining object sizes from the point of view of receiving thermal imaging sensors.

- **pointing theta** and **pointing phi**
  Both these values are obtained from OPNET’s default code during pipeline stage 6
and are stored dynamically into the packets by an edited version of this pipeline stage model as explained in Section 4.3.7.

• **average_values**

Because not all materials react in the same way to infrared radiation, this project implements a way for some objects to be ‘translucent’ to infrared radiation while emitting their own heat. This component of the infrared packet acts as a boolean value to determine if the object is translucent or solid to infrared radiation passing through its shape. An example of this can be seen as in Figure 4.2 whereby the purple colored solid sphere is impenetrable to other infrared radiation whereas the orange colored translucent sphere is penetrable.

![Figure 4.1: Packet format used in the thermal imaging sensor module.](image)

### 4.3 Pipeline Stage Modifications

As mention previously in Chapter 3.2.3, OPNET’s wireless pipeline contains 14 distinct stages. Modifications were not necessary for all pipeline stages, however explanations will be given for all stages, stating the reasons they were or were not modified. No pipeline stage may be skipped for the wireless module to work and therefore a model needs to be specified for every stage, even if it is unneeded.
4.3.1 Pipeline Stage 0: Receiver Group

The network scenarios that are used for this thermal imaging sensor module do not exceed 3 kilometers in length. In such situations, where the network area is relatively small compared to the surface of the earth, OPNET automatically includes all receivers into a transmitter’s Receiver Group. Therefore no changes were necessary to this pipeline stage.

4.3.2 Pipeline Stage 1: Transmission Delay

Transmission delay statistics are usually used in the wireless module along with propagation delay statistics to show how long it takes for packets to reach a receiver from the transmitter. This can be vital for some time critical data that needs to be transmitted and received by a certain amount of time.

However, because the packets sent from an infrared emitting object in this module needs to mimic real infrared radiation by making the packets appear continuous, transmission delay statistics would not serve any purpose other than to show the developer how long it...
takes for an infrared packet to be transmitted and use this value to change other parameters as necessary. As such, no changes were made to this pipeline stage.

### 4.3.3 Pipeline Stage 2: Link Closure

OPNET provides two link closure models that may be used in this stage. The first, called `dra_closure`, is the default model and performs the operations mentioned in Section 3.2.3. The other, called `dra_closure_all` automatically allows closure for all transmitter-receiver pairs.

The second model was used as is in this pipeline stage because it was desired for every sensor to be able to detect all infrared radiation initially before being filtered at a later point.

### 4.3.4 Pipeline Stage 3: Channel Match

As with the reasoning applied in the previous section, the channel match stage was edited to ensure that all infrared radiation within visible sight can be sensed by the thermal imaging sensors.

The model for this stage was edited to ensure that the only factors taken into account when calculating a channel match are the matching of the transmitter-receiver frequencies and bandwidth. Detailed explanations on these parameters will be given in Sections 4.4 and 4.5.

### 4.3.5 Pipeline Stage 4: Transmitter Antenna Gain

Proper infrared radiation has many properties that are similar to visible light including propagation characteristics. Unlike radio communications, infrared signals are not supposed to degrade with distance unless there are foreign components such as dust or moisture in its path. As of yet, modeling of foreign components has not been introduced into this module.

With this reasoning in mind, the amount of gain seen by a thermal imaging sensor when receiving an infrared packet is unimportant. The model for this pipeline stage was left
untouched as a future implementation of foreign components in the transmission medium might need to use OPNET’s transmitter antenna gain pipeline model.

### 4.3.6 Pipeline Stage 5: Propagation Delay

As with the reasoning given for pipeline stage 1, propagation delay is not an important factor to consider in the transmission of infrared packets. This pipeline stage was left intact for debugging and future development purposes.

```c
// Add the value of tx_phi and tx_theta into the current packet
op_pk_nfd_set(pkptr, "pointing_phi", tx_phi);
op_pk_nfd_set(pkptr, "pointing_theta", tx_theta);
```

Figure 4.3: Code excerpt of changes made to the Receiver Antenna Gain pipeline stage.

### 4.3.7 Pipeline Stage 6: Receiver Antenna Gain

Similar to that of pipeline stage 4, the model for this pipeline stage was left untouched save for additional code to store information obtained from this stage. The model use in this stage calculates the information of the transmitting node’s pointing phi and pointing theta values in relation to the sensor, both of which will be explained in greater detail in Section 4.5. Because these values are needed by the thermal imaging sensor process module, they are stored into designated storage areas of the transmitting infrared packet that passes through this pipeline stage. An excerpt of this code is shown in Figure 4.3.

### 4.3.8 Pipeline Stage 7: Receiver Power

As with the gain pipeline stages, receiver power is not a value that needs to be considered in the transmission of infrared packets. This pipeline stage was left as is except for a portion of the code that was removed. As shown in Figure 4.4, this portion of the code locks the
receiver channel onto a single incoming packet. This implementation is not plausible for
the thermal imaging module where information from multiple infrared source objects need
to reach a sensor.

```
{  
    /* The receiving node is enabled. Get */  
    /* the address of the receiver channel. */  
    rx_ch_obid = op_td_get_int (pkptr, OPC_TDA_RA_RX_CH_OBJID);

    /* Access receiver channels state information. */  
    rxch_state_ptr = (DrxA_Rxch_State_Info *) op_ima_obj_state_get (rx_ch_obid);

    /* If the receiver channel is already locked, */  
    /* the packet will now be considered to be noise. */  
    /* This prevents simultaneous reception of multiple */  
    /* valid packets on any given radio channel. */  
    if (rxch_state_ptr->signal_lock)
        op_td_set_int (pkptr, OPC_TDA_RA_MATCH_STATUS, OPC_TDA_RA_MATCH_NOISE);
    else
    {  
        /* Otherwise, the receiver channel will become */  
        /* locked until the packet reception ends. */  
        rxch_state_ptr->signal_lock = OPC_TRUE;
    }
}
```

Figure 4.4: Portion of the Receiver Power pipeline stage code that was commented out.

### 4.3.9 Pipeline Stages 8 though 12

Along with many of the previous pipeline stages, the role of calculating background noise,
interference noise, signal to noise ratio, bit error rate and error allocation is to determine if
a transmission’s signal strength has degraded in the process of reaching intended receivers.
Noise accumulated from the medium and other packets are taken into account in these
stages.

However, as mentioned previously, it is the intention of the module to allow all infrared
packets within a sensor’s field of view to reach a sensor node’s receiver regardless of any
other interfering particles. Due to the fact that these stages only calculate values but do not
perform any operations to destroy packets, they were left as is.
4.3.10 Pipeline Stage 13: Error Correction

This final pipeline stage makes use of the values determined in stages 8 though 12 and destroys packets that are deemed unfit for reception. Therefore, it was necessary to edit the code to ensure the reception of all infrared packets. The portion of the code that was edited out is shown in Figure 4.5.

```c
/* obtain the error correction threshold of the receiver. */
ecc_thresh = op_tc_get_db1 (pkptr, OPCODE_RA_ECC_THRESH);

/* Obtain length of packet. */
packlen = op_pk_total_size_get (pkptr);

/* Obtain number of errors in packet. */
num_errs = op_tc_get_int (pkptr, OPCODE_RA_NUM_ERRORS);

/* Test if bit errors exceed threshold. */
if  (packlen == 0)
    accept = OPCODE_TRUE;
else
    accept = (((double) num_errs) / packlen) <= ecc_thresh ? OPCODE_TRUE : OPCODE_FALSE;
```

Figure 4.5: Portion of the Error Correction pipeline stage code that was edited.

4.4 Infrared Emitting Object Module

Figure 4.6: Node Model of the infrared emitting objects.

Figure 4.6 shows the three elements that make up the infrared emitting objects’ node model. This section will explain each component in detail.
4.4.1 *heat-gen* Process Model

This process model is used by the node to generate the infrared packets that are transmitted. The states within this process model are shown in Figure 4.7.

![Process Model of the infrared emitting objects.](image)

**Figure 4.7: Process Model of the infrared emitting objects.**

- **Init**

  The Init state is used primarily to obtain user defined attributes. The attributes that may be changed by a user to affect the thermal imaging module are shown in the highlighted portion of Figure 4.8. The object’s altitude is an important factor to consider as OPNET’s network topology editor only allows latitude and longitude changes when moving node icons in the topology area. To change a node’s height, the altitude attribute needs to be used. The *Start Time* attribute is used to tell the node when the first packet should be sent out. Its unit is in Seconds.

  The rest of the attributes are sent out in the generated infrared packets so that receiving thermal imaging sensor nodes may decipher the characteristics of a particular infrared emitting object.
Figure 4.8: Infrared emitting object attributes that may be edited by a user.

When clicked, the Object Emissivity attribute gives the user a list of emissivity values of some common materials such as in Figure 4.9. The values were obtained from Electro Optical Industries Inc at [20].

- **Transition from Init to Generate**
  The state transition from Init to Generate occurs when the first self generated interrupt triggers, set by the Start Time attribute. During this transition, an infrared packet is generated by the ss_packet_generate() function and sent out of the process’s output stream. An excerpt of this function’s code is shown in Figure 4.10.
• **Transition from Init to Stop**

  The state transition from *Init* to *Stop* only occurs if there was an invalid value entered as the *Start Time* attribute. An example of an invalid value would be a start time of negative value or a start time that is greater than the simulation run time.

• **Generate**

  This state is used to set the next interrupt to generate a packet. The next interrupt time is based on a constant inter-arrival time that does not need to be changed by the user. The value has been set large enough to ensure that a thermal imaging sensor will always receive at least one packet from an object in sight during a capture period. Once a new self-interrupt has triggered, the state transitions back to itself, generating a new packet with the same `ss_packet_generate()` function that was used before. If for some reason, an invalid time has been set for the next interrupt, the state will...
transition to the *Stop* state.

- **Stop**
  
  The *Stop* state cannot be exited once entered. All interrupts intercepted while in this state are acknowledged and then ignored.

### 4.4.2 *heat_radio_tx* Transmitter Model

The *heat_radio_tx* transmitter model is identical to the default model provided by OPNET except for a number of altered attributes. The green highlighted portion of Figure 4.11 shows the attributes that were changed.

- **data rate (bps)**
  
  The data rate of transmission was increased from the default of 1,024 bps to 1,000,000 bps. The reason for doing so is to accelerate the transmission of packets from the infrared emitting objects so that they appears to be continuous when compared to the transmission of a typical radio link packet.

- **bandwidth (kHz) and min frequency (MHz)**
  
  The bandwidth and minimum frequency attributes relate to the portion of the electromagnetic spectrum that is used by this receiver model. The bandwidth’s value was left intact whereas the value for the minimum frequency was changed from the default of 30 MHz to 5 MHz. The numerical values of these attributes are unimportant...
since they have no effect on the later stages. However it is important that the spectrum used by the thermal imaging module does not overlap with that used by a user’s own wireless transmission module. If this is not checked, then the user will notice an immense increase in the noise seen by the wireless transmission module. The number of packets that a thermal imaging sensor receives will increase as well, causing an increase in computation time to destroy these foreign packets once received by its process model.

• closure model

The closure model was changed from OPNET’s default pipeline model to the version explained in Section 4.3.3.
• **chanmatch model**
  
The chanmatch model was changed from OPNET’s default pipeline model to the version explained in Section 4.3.4.

• **Other attributes**
  
All other attributes were left as is. Noticeably different is the transmit power value that was changed from the default of a 100 Watts to 1 Watt. The transmit power value plays no role in the reception of infrared packets by thermal imaging sensors. The value was changed to ensure the user is alert about this fact.

### 4.4.3 *heat_ant_tx* Antenna Model

All attributes of the antenna model were left at default with the antenna pattern being isotropic as in Figure 3.4.

### 4.5 Thermal Imaging Sensor Module

Figure 4.12 shows the three elements that make up the thermal imaging sensors’ node model. This section will explain each component in detail.

![Figure 4.12: Node Model of the thermal imaging sensor objects.](image)

### 4.5.1 *heat_ant_rx* Antenna Model

As with the antenna of the infrared emitting object module, the antenna attributes here are left unchanged.
4.5.2 *heat radio rx* Receiver Model

The *heat radio rx* receiver model is identical to the default model provided by OPNET except for a number of altered attributes. The green highlighted portion of Figure 4.13 shows the attributes that were changed.

- **data rate (bps)**
  The data rate of transmission was increased from the default of 1,024 bps to 1,000,000 bps. The reason for doing so is to accelerate the transmission of packets from the infrared emitting objects so that they appear to be continuous when compared to the transmission of a typical radio link packet.

- **bandwidth (kHz) and min frequency (MHz)**
  The bandwidth and minimum frequency attributes relate to the portion of the electromagnetic spectrum that is used by this receiver model. The bandwidth’s value was left intact whereas the value for the minimum frequency was changed from the default of 30 MHz to 5 MHz. The numerical values of these attributes are unimportant since they have no effect on the later stages. However, it is important that the spectrum used by the thermal imaging module does not overlap with that used by a user’s own wireless transmission module. If this is not checked, then the user will notice an immense increase in the noise seen by the wireless transmission module. The number of packets that a thermal imaging sensor receives will increase as well, causing an increase in computation time to destroy these foreign packets once received by its process model.

- **ragain model**
  The receiver gain model was changed from OPNET’s default pipeline model to the version explained in Section 4.3.7

- **power model**
  The receiver power model was changed from OPNET’s default pipeline model to the
version explained in Section 4.3.8.

- **ecc model**

  The error correction model was changed from OPNET’s default pipeline model to the version explained in Section 4.3.10.

![Edited Receiver model attributes](image)

Figure 4.13: Edited Receiver model attributes.

### 4.5.3 *heat Proc* Process Model

This process model is the primary engine for the thermal imaging module. It is used to obtain infrared packets from the receiver and interpret the data obtained before creating capture images. The states of this process model are shown in Figure 4.14. Details on each state and transition follows.
• **Init**

The Init state is run once during simulation initialization. The purpose of this state is to obtain user and node defined attributes while preparing the sensor model for subsequent captures.

Initially, the state attempts to obtain information from its designated marker and ambient temperature modules, both of which are explained in Section 4.6. If either attribute cannot be obtained, the code will immediately force the simulation to abort and displaying an error message. The marker position is used promptly to set the sensor node’s antenna target whereas the ambient temperature value is stored to be used repeatedly at later stages.

Following this, the state reads in the user defined attributes that are highlighted in Figure 4.15. As mentioned in Section 4.4.1, the altitude setting plays an important role in changing the 3-dimensional coordinates of a node in the topology. The Capture
Length and Sleep Length both refer to the amount of time the sensor should spend capturing images and sleeping. Typically, the capture time need not be changed from the default value of 0.1 seconds because the transmission rate of data from the transmitter to the receiver is very high, at 1,000,000 bps. The sleep length allows the user to specify the amount of time that should lapse between captures and should be changed according to the user’s needs.

The range of temperature that can be sensed is also user specified in Max Temperature and Min Temperature. These variables may be set to any value as long as the maximum is larger than the minimum. It should be noted however, that the thermal image code uses 256 shades of gray ranging from white to black to shade each picture. These shades will need to cover the whole temperature range and therefore accuracy of shading will drop progressively as the user increases the temperature range. Any infrared emitting objects not in this range cannot be detected by the sensor.
Max Viewing Angle defines the sensor’s viewing angle in degrees. A larger viewing angle would mean more coverage but at a cost of resolution as a larger viewing area causes objects to look smaller when the number of pixels in the sensor remains the same.

One of the important factors to consider in the sensor’s accuracy is the Sensor Emissivity setting. A disparity between the sensor’s emissivity and that of an infrared emitting object would cause an inaccurate portrayal of the object’s temperature. The highlighted area of Figure 4.16 shows that the emissivity difference is used as a ratio to alter temperature the gray shade of an object.

```c
// Calculate shade based on a linear scale
calc_shade = (LIGHTEST-DARKEST) / (max_temp-min_temp) * source_heat * source_em / sensor_emissivity;
```

Figure 4.16: Code excerpt showing the effect of emissivity on temperature calculations.

Image resolution can be changed using the Sensor Pixel Row / Column. The numerical value specified here creates a square matrix that will be used to store the images. The user should change this value to meet their needs and not exceed it as an increase in resolution also causes an increase in simulation time.

Finally, the last two user specifiable attributes allow a user to define when the sensor should start and stop during the simulation time. Any values entered that are invalid will cause the sensor to enter the Disabled state at the next self scheduled interrupt.

Before exiting, this state sets the next self interrupt for a capture event.

- **Transition from Init to Sleep**
  
  A transition is made from the Init state to the Sleep state if the start time specified by the user is different from 0.0. This causes the process to sleep until the next capture self-interrupt triggers.

- **Transition from Init to Disabled**
If the user specified an invalid start or stop time, a transition is made to the Disabled state at the next self-interrupt.

- **Transition from Init to Capture**
  If the start time was set to 0.0, the state would transition to the Capture state to perform the first image capture. During this transition, the function `ss.sleep_time()` is called to schedule a self-interrupt for the next sleep event. All incoming packets that are received before the sleep interrupt will be received by the sensor and processed to produce an image for the particular capture period.

- **Capture**
  The Capture state contains no code. It is used as a hold state while waiting for the next event.

- **Transition from Capture to New Data**
  All incoming packets received during the Capture state cause a transition to the New Data state.

- **Transition from Capture to Sleep**
  If a self-interrupt triggers to sleep, the transition is made to the Sleep state. During this transition, the next capture event is scheduled by calling the `ss.capture_time()` function.

- **Transition from Capture to Disabled**
  If the node’s stop time has been reached while in the Capture state, a transition is made to the Disabled state.

- **New Data**
  All packets received during the capture length are processed by the New Data state. Packet events are queued by OPNET’s simulation kernel and therefore, the amount of simulation time (as opposed to real time) required to process the packets received in this time frame is independent of packet arrivals. Initially the state checks if the
packet received is a valid infrared packet. This should not be an issue if the user fol-
lowed the guidelines stated previously about the bandwidth and minimum frequency
user defined attributes.

Once checked, the state compares the current packet source’s object ID with that 
of object IDs already in a list. Only packets not already in this list are processed
as they contain different data from those already in the list. Processing redundant 
repeated packets would only waste valuable resources. A packet identified as being
unique is then processed using the \textit{ss\_process\_element()} function which extracts the
information from the packet and also calculates the sphere’s midpoint in relation to 
the sensor’s field of view before storing these values as a new object in a vector.

![Diagram](image)

Figure 4.17: 3-dimensional view of pointing $\phi$ angle.
Calculating the sphere’s midpoint requires the use of the *pointing phi* and *pointing theta* values first obtained from Section 4.3.7. The *pointing phi* value refers to the angle between the infrared emitting node’s center and the Z-axis of the sensor. In a 3-dimensional plane, an infrared emitting object appearing on the first quarter of a sensor’s X-Y plane of view, would have a $\phi$ angle as shown in Figure 4.17. The distance between these two nodes, as provided by OPNET’s simulation kernel is labeled *Distance* in the same figure. The projection of this distance onto the X-Y plane is labeled $D’$. A 2-dimensional view of the same $\phi$ angle is shown in Figure 4.18.

![Figure 4.18: 2-dimensional view of pointing $\phi$ angle.](image)

The *pointing theta* value refers to the clockwise angle from the positive X-axis to the object’s center. Figure 4.19 shows this angle with the projection of *Distance*, $D’$ being the reference for the center of the node. The numerical value for $D’$ can be found using Figure 4.18 as a reference as:

$$D’ = \text{Distance} \times \sin \phi$$

At the same time, the length of the Z-axis when *Distance* is projected onto it is:
From these available information, it is possible to transform the object’s position into X and Y coordinates as in Figure 4.20 with the formulas below:

\[
x\text{-val} = D' \times \cos \theta \\
y\text{-val} = D' \times \sin \theta
\]

Once the x and y values have been obtained, their length in reference to the viewing angle needs to be calculated as objects outside of this angle need to be discarded. Another factor to consider is the radius angle which is the angle between the object’s center and outer edge as seen from the x-y plane. Depicted clearly in Figure 4.21, this angle is calculated as follows:
The viewing angle as specified by the user is divided into both halves of the X-axis. Therefore, the maximum viewing angle for the positive X-axis is half of the user set attribute, as in Figure 4.22. The process needs to now ensure that packets from all objects within this viewing limit are accepted as valid whereas the rest are discarded. This will need to include all objects who’s radius is still in the viewing angle although the node center is not, as depicted by the purple circle of Figure 4.23. Only when a circle’s radius is out of the field of view can its packet be discarded, as the green circle of the same figure shows.

Objects that are accepted need to have their x and y values translated into pixel values. This is done by finding the ratio between $D'$ and $D_{Max}$ and multiplying this value by the number of pixel rows or columns. Before the x and y values are stored, they are

\[ \text{radius angle} = \tan^{-1}\left( \frac{(D' + \text{radius})}{Z} \right) - \phi \]
off-set by the number of pixel rows to shift the origin to the upper left hand corner. This is done to allow easy writing to the matrix at a later stage.

Finally the packet’s object ID is added to the initial list to ensure that information from the same infrared emitting object is never reprocessed.

At the end of this state, a transition is made back to the Capture state. This transition does not require an interrupt. Once back in the Capture state, the process waits for the next interrupt which could be from a packet or self-generated.

• **Sleep**

Once a self-interrupted transition to the sleep phase occurs, the Sleep state creates an image using the data obtained from all the packets collected during the capture phase.

Each packet object stored in the vector by the New Data state represents a unique infrared emitting object that needs to be rendered on an image. These objects are initially sorted by distance and then inserted into the pixel matrix created in the Init state beginning with the furthest packet first. The reason for doing so is to ensure that in an event that objects overlap, the closer objects will always shade over further
ones. Once completed, the function, ss_insert_matrix() is called for each infrared emitting object using the object’s previously calculated x and y values, radius and distance. The radius of the object will need to be sized in proportion to its distance from the sensor. A realistic rendition of an object will need to reduce its size as the object moves further away from the sensor. The number of pixels that should be rendered for an object’s radius as a function of the distance and radius is as follows:

\[
\text{Radius Pixel} = \frac{(\text{Pixel Rows} \times \text{Radius})}{(\text{Distance} \times \tan(\frac{\text{ViewAngle}}{2}))}
\]

The following pseudo-code is then executed, with visual aid from Figure 4.24:

1. The coordinates for \(Y_{\text{min}}\) and \(Y_{\text{max}}\) are found by subtracting and adding \(\text{Radius Pixel}\) from the object center’s Y coordinate.

2. The values for \(Y_{\text{min}}\) and \(Y_{\text{max}}\) are tested to ensure they are within the bounds of the matrix. If not, change the values appropriately. Using Figure 4.25 as an example, the calculated value for \(Y_{\text{min}}\) for the object rendered is at the top of the object. However, due to being limited by the matrix size, the value for \(Y_{\text{min}}\)
is changed to the matrix limited value. The final $Y_{\text{min}}$ and $Y_{\text{max}}$ values refer to the range of rows in the matrix that is occupied by the object.

3. For every value of $Y$ in the bounds of $Y_{\text{min}}$ and $Y_{\text{max}}$, the corresponding value for $X_{\text{min}}$ and $X_{\text{max}}$ needs to be found as in Figure 4.24. Figure 4.26 shows that any point on a circle can be translated into $x$ and $y$ coordinates based on the circle radius, $L$, and the angle between the point and the $x$-axis. By deriving this formula, the values for $X_{\text{min}}$ and $X_{\text{max}}$ can be found as:

$$
x_{\text{min}} = \lfloor X_{\text{center}} - (\text{Radius Pixel}) \cdot \cos(\sin^{-1} \frac{Y - Y_{\text{center}}}{\text{Radius Pixel}}) \rfloor
$$

$$
x_{\text{max}} = \lceil X_{\text{center}} + (\text{Radius Pixel}) \cdot \cos(\sin^{-1} \frac{Y - Y_{\text{center}}}{\text{Radius Pixel}}) \rceil
$$

The floor and ceiling of the values are found respectively to ensure consistency in the matrix indexing. $X_{\text{center}}$ and $Y_{\text{center}}$ refer to the $x$ and $y$ coordinates of the object’s center. As with the $Y$ values, the values found here need to be checked for bounds to ensure that they are within the matrix’s limits. The final $X_{\text{min}}$ and $X_{\text{max}}$ values refer to the range of columns in the matrix that is occupied by the object.

4. All columns in the current row are then filled with the corresponding shade of the object. This value needs to be in the range of 0 to 255 inclusive. This value
Figure 4.24: Visual representation of object rendition on the sensor pixel matrix.

is found based on the temperature of the object and the detectable temperature range of the sensor as follows:

\[
Shade = \left(\frac{object\ temperature}{max\ temp - min\ temp}\right) \cdot \frac{255 \cdot object\ emissivity}{sensor\ emissivity}
\]

If the current object is infrared penetrable, then the algorithm needs to check if the current pixel has already been shaded by a previous object. If so, then the current shade value of the pixel is obtained and averaged with the new value before being replaced. If no object occupies the pixel, the shade is simply inserted into it.

At the end of this process, the matrix of integers will be filled with values ranging from 0 to 255 that as a whole make up an image of the captured event. The image
Figure 4.25: Process to calculate the values for $Y_{\text{min}}$ and $Y_{\text{max}}$.

is then stored in a PGM (Portable Gray Map) formatted file. This format is simple to use as the matrix integer information can be directly saved into the file without any conversions. An example of a PGM file opened using a text editor is shown in Figure 4.27(a) and the image itself is shown in Figure 4.27(b). The second row of the format, highlighted in green in the figure specifies the number of rows and columns in the image, corresponding to the pixel attribute set by the user. The third row, highlighted in red shows the number of shades (from 0 to 255) possible for the image. The file name format is as follows to help easily identify saved images once created:

$\text{ProjectName}_\text{ScenarioName}_\text{SensorName}_\text{SimulationTime}.pgm$

Once file save is completed, the vector used by the New Data state is cleared and the matrix is refilled with the ambient temperature values as done in the Init state.

- **Transition from Sleep to Capture**
Figure 4.26: Formula to find x and y coordinates of any point on a circle.

If a self-interrupt triggers to capture, the transition is made to the Capture state. During this transition, the next sleep event is scheduled by calling the `ss_sleep_time()` function.

- **Transition from Sleep to Destroy**
  Any packets received while the node is still sleeping will cause an interrupt to transition to the Destroy state.

- **Transition from Sleep to Disabled**
  If the node’s stop time has been reached while in the Sleep state, a transition is made to the Disabled state.

- **Destroy**
  All packets received during the sleep period will cause an interrupt that is serviced by the Destroy state. The only task performed by this state is to destroy these unneeded packets.
4.6 Marker and Ambient Temperature Modules

Two of the more simpler modules implemented in this project are the marker and ambient temperature modules. These modules are created to assist the user in setting up the network environment pre-sim.

4.6.1 Marker Module

As mentioned in Section 3.2.2, a node’s antenna module needs to have a pointing target that specifies the initial point in space where the antenna is to be directed. This point is changeable during the simulation run-time but needs to have a valid value at start time. The marker module provided in this project simplifies this process, especially when it is desired to have multiple thermal imaging sensors targeting the same point in space.

Without the use of this module, each thermal imaging sensor would need to be given a target latitude, longitude and altitude to point to. For a model such as in Figure 4.28, these
values will need to be manually entered into each and every one of the 8 thermal imaging sensors in the network. Not only is this task tedious, it is also strenuous as there is no easy way to determine a latitude-longitude-altitude global position pre-simulation.

One possible way to do this is to initially create a simulation that places an arbitrary node at the target and create a process in the node that calls the `opima_obj_pos_get()` Kernal Procedure to obtain these values. Once the values have been recorded, they can be entered into each sensor node before the actual simulation is run.

The marker module facilitates this process tremendously. Markers are places in desired locations on the network topology. It is only necessary to have one marker per target location as any number of thermal imaging sensors can refer to the same marker. Once a marker has been placed at the desired location as the X icon in Figure 4.28, the user only needs to specify the marker’s name in the sensor node’s attributes in the network domain as depicted by the green highlighted part of Figure 4.29. As OPNET users know, attributes set...
here can be done for multiple nodes of the same model simultaneously as shown in the red highlighted part of the same figure. Therefore, this process only needs to be done once per target position. During the simulation initialization, the sensor node will query OPNET’s kernal for the global position of the given marker and use the obtained values as a target for its infrared antenna module. Because the marker module serves no other purpose than to define a target, this module contains no process modules.

Figure 4.29: Sensor node module attributes to be edited to use a marker.

As the thermal imaging sensor module provided in this project is rudimentary, the sensor node’s target position is not changed after the initial setting. Thus, tracking of a moving marker has not been implemented, although this can easily be added to the module.
4.6.2 Ambient Temperature Module

A sensor node’s process module needs to accurately shade regions of its viewing angle that are not filled by infrared emitting objects in its sight. To do so, these areas should logically be shaded with the ambient temperature of the network area. There are multiple ways to let a thermal imaging sensor obtain this information, but it was decided that the most user-friendly and least computational intensive method should be used.

Hence the creation of the ambient temperature module. The module’s node model contains a variable called Temperature with a default value of 22°C, which is the typical value set for room ambient temperature. Of course, a user is free to change this value in the ambient temperature node’s attributes as shown in the green highlighted portion of Figure 4.30.

At simulation initialization, the thermal imaging sensor nodes in the network area look for an ambient temperature node module in the same topology. Once it is found, its ambient temperature value is queried and stored in each sensor node. Note that if two or more ambient temperature modules exist in the topology area, only the value of the first one placed in the topology is used. This is because OPNET assigns unique object IDs incrementally to each object set in the topology in the chronological order that it was created by the user. When a sensor node queries OPNET for the object ID of an ambient temperature module, only the object ID of the ambient temperature module with the smallest value is returned.

Both the marker and ambient temperature nodes need to be set in the network topology for the simulation to run. If either is missing, the simulation will stop with a message for the corresponding error as in Figure 4.31.
Figure 4.30: Ambient temperature module attributes to be edited.
Figure 4.31: Error messages corresponding to invalid or missing marker and ambient temperature modules.
Chapter 5

Thermal Imaging Module Results

This chapter of the documentation shows the capabilities of the thermal imaging module as well as simulation performance results.

5.1 Capabilities

The various capabilities of the thermal imaging module are shown in this section. Each capability is followed by sample captured images and a view of the scenario used in the demonstration.

5.1.1 Image Resolution

The thermal imaging sensor module allows a user to specify the resolution of images taken. The images captured are of a square size and therefore the resolution specified represents both the number of rows and columns of the sensor matrix. Figure 5.1 shows how an image taken by a sensor with a 400 by 400 pixel resolution compares with that taken by a sensor with a 100 by 100 pixel resolution. Notice that Figure 5.1(b) is the more pixelated image of the two.
Figure 5.1: Effect of resolution on the image quality.
Figure 5.2: Effect of distance on object size in captured image.
5.1.2 Viewing Capabilities

The distance of an object is directly correlated to its size as seen by the sensor. Figure 5.2 shows two sensors of the same configuration viewing an object of radius 0.5 meters from 10 meters and 5 meters away respectively. The object rendition produced by the closer sensor is roughly double that of the further sensor.

When a closer object is in the path of a further object as in Figure 5.3(b), its formation overlaps that of the further image, simulating how certain objects can block the path of infrared radiation. Using the same reasoning, it can be seen in Figure 5.4(b) that the closer larger object completely blocks the view of the smaller further object.

The image captured by a sensor will change as the sensor moves. Figure 5.5 shows the different captures made by a sensor that changes its altitude from the same as the infrared emitting objects to an elevated position 1 meter higher. An interesting observation is that the furthest object which is not viewable in Figure 5.5(a) can be seen once the sensor alters its position as in Figure 5.5(b).

5.1.3 Infrared Properties

Some infrared emitting objects may be penetrated themselves by other infrared radiation. In such cases, the sensor will be able to see ‘through’ the object to detect the infrared emitting object behind it. Figure 5.6(b) shows this clearly whereby the larger object closer to the sensor is set to be infrared penetrable and therefore allowing the smaller object further from sensor_1 to be detected.

5.1.4 Emissivity Properties

As mentioned in Section 2.3.2, the emissivity values of the sensor and the infrared emitting objects change the way a sensor interprets an object’s temperature. Shown in Figure 5.7, two objects of the same temperature are rendered differently. The sensor renders IR_source_1’s temperature accurately because it has the same emissivity as that object.
Figure 5.3: Effect of distance and size on captured images.

(a) Image captured by sensor_0.
(b) Image captured by sensor_1.
(c) Viewing capabilities test 2 scenario.
Figure 5.4: Effect of distance and size on captured images.
(a) Image captured at same altitude as objects.
(b) Image captured at a higher altitude than objects.

(c) Viewing capabilities test 2 scenario.

Figure 5.5: Environment visualization from different angles.
Figure 5.6: Object rendition differences due to penetrability.
However, due to the fact that the emissivity value of IR_source_0’s is exactly half of the sensor’s, it is only rendered at half its original temperature.

An extension of the previous test, Figure 5.8 shows the same two infrared emitting objects along with two sensors with different emissivity value settings. Both sensors are not able to detect objects with different emissivity settings accurately. Figure 5.8(a) shows that sensor_0 renders IR_source_1 as cooler than its actual temperature whereas sensor_1 is unable to render IR_source_0 at all due to the fact that its emissivity setting causes its temperature to appear beyond the range of the sensor.

Two objects of different temperatures can appear to look the same to the sensor due to emissivity values. Figure 5.9(b) shows 2 objects, IR_source_0 with an emissivity of 0.5 and a temperature of 400°C and IR_source_1 with an emissivity of 1.0 and a temperature of 200°C. However, as shown in Figure 5.9(a), the sensor renders both objects to appear at the same temperature due to the fact that IR_source_0’s emissivity setting causes the sensor to see its temperature as 200°C and not 400°C.

5.1.5 Sensor Viewing Limits

An infrared emitting object will be rendered by the sensor as long as it is within the viewing angle. Objects moving further away are rendered until they are too small to be shaded by a pixel area, at which time they disappear from the image. For sensor_0 in Figure 5.10, Figure 5.11 shows a portion of this rendition. The same applies for objects moving from one corner of the sensor’s viewing area to another, as depicted in Figure 5.12 (sensor_1 in Figure 5.10) where the object continues to be rendered as long as it is in the sensor’s field of view.

5.2 Performance

This section will show the burden of running simulations with the thermal imaging module implemented. For reference purposes, half of the simulations will include an OPNET
(a) Different shading for objects of the same temperature.

(b) Emissivity test 1 scenario.

Figure 5.7: Object rendition disparity due to emissivity values.
Figure 5.8: Object rendition disparity due to emissivity values.
Figure 5.9: Object rendition disparity due to emissivity values.
example wireless network module that transmits packets over large networks.

5.2.1 Environment Settings

All simulations were performed on the system specified in Table 5.1.

The basic network used to perform the performance simulations is shown in Figure 5.13. This network can be readily obtained from OPNET’s example networks by opening the Project Model called WLAN and switching to the scenario called large_network. The scenario description as given by OPNET [19] is as follows:
Figure 5.11: Near-far rendition of object by sensor_0 of scenario depicted in Figure ??.

Figure 5.12: Left-right rendition of object by sensor_1 of scenario depicted in Figure ??.

<table>
<thead>
<tr>
<th>Component</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>MS Windows XP Service Pack 2</td>
</tr>
<tr>
<td>Processor</td>
<td>Intel Core 2 Duo running at 2.33 GHz</td>
</tr>
<tr>
<td>Physical Memory</td>
<td>2 GB DDR2 SDRAM running at 667 MHz</td>
</tr>
</tbody>
</table>

Table 5.1: System profile of simulation computer.
Figure 5.13: Base network model used in conjunction with the thermal imaging module to evaluate performance.
• **Objective**

This is a 89-node WLAN network highlighting different WLAN algorithms and features specified in IEEE 802.11 and 802.11b standards, like various WLAN data rates, RTS/CTS frame exchange, data packet fragmentation and roaming.

• **Configuration**

The network consists of a central bridge, 8 APs (access points) connected to that bridge, whose BSS IDs are set as 0 to 7, and 10 WLAN stations in the BSS of each AP. The stations that are in BSS 0 initially rotate in clockwise direction and complete their circle towards the end of the simulation. Hence, they are expected to get connected to all existing APs while they are traversing their trajectory. Similarly, the stations that reside in BSS 6 initially visit each AP in counter-clockwise direction throughout the simulation. All the other stations are stable during the simulation.

Different APs use different WLAN data transmission rates: AP_0 and AP_4 1Mbps, AP_1 and AP_5 2Mbps, AP_2 and AP_6 5.5Mbps, and AP_3 and AP_7 11 Mbps. Additionally RTS/CTS frame exchange is enabled for AP_2, AP_3, AP_4 and AP_6, and fragmentation of large packets is enabled for AP_1, AP_3, AP_5 and AP_6. These parameters of station nodes are configured same way as they are configured for their initial APs.

Each station generates on average one packet for every 1.2 seconds that is randomly destined for another station in the network.

The transmission powers of the WLAN nodes are set to a value (2 mW) that is lower than the default value in order to reduce the coverage area of the APs so that these areas don’t overlap with each other significantly and a mobile STA is handed over to the next AP when it is around the mid-way between two APs.

The network was altered by removing all node mobility and adding the thermal imaging module to the node model as depicted in Figure 5.14. It should be stressed that the process
of interpreting data provided by the sensors for use by the wireless network is not implemented as this function is beyond the scope of this thesis. All that is shown here is how the process of providing data for a user’s wireless sensor network model with affect system performance. Infrared emitting objects are then added to the network scenario as needed. All infrared emitting object temperatures are kept at a constant 355 °C and with a radius of 10 meters. This large radius is used due to the fact that the network topology is vast, roughly 3 kilometers end to end, and the objects need to be large enough to be viewed by all sensors. All infrared emitting objects perform a square trajectory around the center of the network topology. This trajectory, which takes 8 movements to complete, is shown in Figure 5.15. The file save function in the thermal imaging sensor module was removed as a physical file save results in a simulation slowdown that is not in sync with the simulation time.

![Node model of test scenario.](image)

The performance evaluations are divided into two sections, one with the number of
Figure 5.15: Figure depicting trajectory performed by each infrared emitting object.
infrared emitting objects as the variable and the other with the number of sensors as the variable. The reason for doing so is to show which of the two components of the module cause a larger burden on the system performance. Each performance evaluation collects statistics for:

- **Total number of system events**
  OPNET’s simulation is based on the number of events created by components in the simulation run. Not all events take the same amount of time to execute but with holding many variables constant it can be shown that the number of events is directly related to the length of the simulation time.

- **Average system speed**
  OPNET’s average system speed is calculated based on the number of events the simulation is able to process per second. If the events are computationally intensive, then the average speed will drop.

- **Simulation run time**
  The most telling performance statistic is the real-time needed to complete the simulation. This statistic pays no regard to the computational intensity of individual system events or the number of events processed at a certain point in time, relying instead on the time between the start and end of simulation.

### 5.2.2 Varying Number of Infrared Emitting Objects

The first performance evaluation varies the number of infrared emitting objects while holding the number of sensors at a constant of 80 objects.

From Figure 5.16 it can be seen that as the number of infrared emitting object grows, there is a linear growth in the number of events created. This is due to the fact that the total number of infrared packets being broadcasted throughout the system is also increasing linearly. The total number of events in a system with a wireless module included only shows a shift upwards. This is due to the fact that the number of events generated by the
wireless module remains the same regardless of the number of infrared emitting objects as the wireless module is only implemented on the thermal imaging sensors and not the infrared emitting objects.

A first glance at Figure 5.17 is misleading as it shows the combined model to have a better average speed performance than the standalone model. However, what needs to be factored in when interpreting the data is that not all events take the same amount of time to execute. Because the events generated by the wireless module are less computational intensive, the average number of events that can be processed by the system increases, explaining the results shown by the graph.

The most unambiguous results can be inferred from Figure 5.18. Here it can be seen that there is a sup-linear growth in the simulation time as the number of infrared emitting objects are increased. This observation is due to the fact that increasing the number of infrared emitting objects necessitates more computations in the sensor modules as content from more objects need to be interpreted into images. Due to the fact that the number of events produced by the wireless module remains a constant, the simulation time of the combined model shows a fixed shift upwards in the graph.

5.2.3 Varying Number of Sensors

The second performance test varies the number of thermal imaging sensors while keeping the number of infrared emitting objects at a constant of 80.

Figure 5.19 shows the change in the number of system events generated when the number of thermal imaging sensors are varied. The number of events generated by each sensor module to produce images remains the same due to the fact that the number of infrared emitting objects is a constant. Therefore it can be seen that the increase in system events is linearly correlated with the increase in thermal imaging sensors for the standalone system. When the thermal imaging module is combined with the wireless module, an exponential correlation is observed. This reveals that the wireless module generates more events as the number of thermal imaging sensors is increased. One of the main reasons for this are
Figure 5.16: Number of infrared emitting objects vs. number of system events.
Figure 5.17: Number of infrared emitting objects vs. average system speed.
Figure 5.18: Number of infrared emitting objects vs. simulation run time.
extra collisions experienced by the wireless packets due to the increased number of packets generated throughout the system.

The average speed of the system when compared to the number of thermal imaging sensors is shown in Figure 5.20. When run in a standalone model, the thermal imaging module shows a gradual decrease in the average speed as the number of sensors are increased. When combined with the wireless module however, it can be seen that a generally better average speed is obtained. As with before, this goes to show that the events generated by the wireless module are less computationally intensive when compared to the thermal imaging module.

Finally, the clearest performance evaluation is shown by Figure 5.21. The amount of time taken to simulate the standalone system has a perfectly linear relation with the increase in the number of thermal imaging sensors due to the fact that the number of tasks performed by each sensor remains a constant. With the wireless module included, the graph changes into a sup-linear pattern, reflecting the increase in tasks from collisions caused by the increase in wireless packets.

### 5.2.4 Discussion of Results

From observing the graphs of both performance tests, it can be deduced that the processes of the thermal imaging nodes are more computational intensive when compared to the infrared emitting nodes. Adding wireless capabilities to the system shows that the thermal imaging module does not cause an excessive burden on the simulation time. Therefore, the results shown here prove that it is in fact quite feasible to implement the thermal imaging module integrated with a user specific wireless module while experiencing acceptable simulation performance.
Figure 5.19: Number of thermal imaging sensors vs. number of system events.
Figure 5.20: Number of thermal imaging sensors vs. average system speed.
Figure 5.21: Number of thermal imaging sensors vs. simulation run time.
Chapter 6

Examples of Other Sensor Modules

Two other simple sensor modules were implemented in this project to show the feasibility of doing so and also OPNET’s ability to provide data needed to create specific sensors. The two sensor modules created here are an ambient temperature sensor and a positioning and ground speed sensor.

6.1 Ambient Temperature Sensor Module

This sensor module simply allows a sensor to detect its surrounding ambient temperature. Using a version of the ambient temperature module in Section 4.6 that cycles its temperature from 12° Celcius to 32° Celcius, ambient temperature sensors plot graphs of changes in temperature. The only user editable variable in this model is the sampling interval of the sensor, shown in the green highlighted portion of Figure 6.1. The sampling interval states the number of seconds the sensor should wait before sampling the ambient temperature of the environment again.

The process model of this sensor uses the same function as the thermal imaging sensor to query the ambient temperature model for its current temperature. This is done periodically, according to the set sampling interval. The process model diagram is shown in Figure 6.2. For every query made, the model saves the data as a statistic to be graphed at simulation end. This is done using OPNET’s default statistic module.

To show the data created by the module, a network was filled with four sensors, with a
sampling interval of 20, 10, 5 and 1 second respectively. An ambient temperature marker was added to the network to vary the ambient temperature. Figure 6.3 shows the results produced by the sensors when simulated for 4 minutes and 10 seconds. Mobile_node_3 obtains temperature samples every 20 seconds. Due to the fact that its sampling interval is so large, this sensor does not properly reflect the fluctuation in ambient temperature, instead only showing the user a fluctuation between 32° Celcius and 21° Celcius. On the other hand, mobile_node_0, with a sampling interval of 10 seconds reveals the true variation in the ambient temperature. However, since there are only 5 samples taken per cycle, the user needs to interpolate the data provided to obtain the true change in temperature. This is easily done in this scenario but perhaps more difficult when the change in ambient temperature is random. Of course, the best sampling data is obtained from mobile_node_2 with the smallest sampling interval of 5 seconds. The data provided by this sensor is almost continuous, giving the user no reason to perform any type of data interpolation.

Although insignificant for this example, the amount of events generated by the sensor with the smallest sampling rate will obviously be higher than the rest. Therefore it is up to the user to decide the best sampling rate that will satisfy the simulation needs without unnecessarily over-sampling the environment.

6.2 Positioning and Ground Speed Sensor Module

The second type of sensor module implemented in OPNET provides global positioning sensing and ground speed capabilities to a node, working in the way a GPS system would. By querying OPNET’s simulation kernel for the node’s global position and current trajectory ground speed, the node is able to chart these changes over time.

The process model used for this sensor module is identical to that of the ambient temperature sensor module and is shown by Figure 6.2. At user defined intervals, the sensor’s process model makes use of OPNET’s op_ima_obj_pos_get() and op_ima_traj_info_get() functions that obtain the node’s global position and ground speed from the simulation kernel.
To demonstrate the capabilities of this module, a network model is filled with four positioning sensors. These sensors are implemented with a random mobility configuration that changes the nodes’ positions as the simulation time elapses. Figure 6.4 shows the network model of this test scenario. The changes in the nodes’ positions and speed are recorded with a sampling interval of 5 seconds. The simulation was run for about 30 minutes. The results of tracking the latitudes of these sensors are shown in Figure 6.5, with units of meters, showing how far away the node is from the top left corner of the network. Notice that latitudes of the sensors are different due to the random mobility,
Figure 6.2: Process model of the ambient temperature sensor.

with mobile_node_3 having the biggest change in its latitude. In the same way, tracking of altitude and longitude is also possible.

The corresponding ground speed of the sensors are shown in Figure 6.6. The ground speed has a unit of meters / second. The random mobility implemented in the network provided a random velocity to each node, and therefore not changing their acceleration. This is reflected in the graphs whereby all speeds remain constant after the initial acceleration. Of all the sensor nodes, mobile_node_3 has the highest velocity at 8 m/s. This goes hand-in-hand with the node having the largest change in latitude.
Figure 6.3: Effects of varying the sampling time on the ambient temperature sensor.
Figure 6.4: Network model of the positioning sensor module.
Figure 6.5: Latitude sampling of node positions.
Figure 6.6: Ground speed of nodes.
Chapter 7

Conclusions

The objective of this project was to leverage the abilities of OPNET’s wireless communications module to create a thermal imaging sensor that produces realistic, usable results that may be employed in combination with other modules to create networks with nodes that are robust and autonomous. The results provided in Chapter 5 show all the capabilities of the thermal imaging module including emissivity variations, object sizing based on distance, infrared penetration, sensor resolution and sensor viewing angle. Some of the other features of the module include variable capture times and user-configurable sensor and infrared emitting object altitudes.

Combining the results provided by the thermal imaging sensor module with a user-built wireless sensor network will allow the user to reap multiple benefits. One of the main benefits satisfies the core purpose of this project and that is to allow user-created sensor nodes to obtain important information from the environment in real-time that can be correlated with other sensors in the environment. This is done without aid from the simulation kernel or artificial random processes. Using this sensor module greatly increases the realism of a simulated network that performs object detection.

Nevertheless, there are ways to improve upon the current system. The images produced by the system currently are only in shades of gray. Most advanced physical thermal imaging systems apply false colors to the images to aid the user in identifying objects of different temperatures in an image. The application of false color in the images will require a few changes to the module. Firstly, the range of temperature that the sensors can detect will
need to be constant to better show that a certain color corresponds to a certain temperature. Besides that, the current file save format of .PGM will need to be changed into another format that supports colored data. The easiest way to change the file format would be to change it to a .PPM format that divides each column of information into three parts to represent red, green and blue values. More information on the .PPM format may be obtained from [11].

Another improvement that could be made to the module is the inclusion of other shapes for the infrared emitting objects. The inclusion of other simple shapes such as cubes or pyramids may allow for greater user flexibility in trying to distinguish between different objects in the simulation environment. However, more research needs to be performed on 3 dimensional rendering of these shapes. This is due to the fact that, unlike spheres, other 3 dimensional objects will need to be rendered differently by a sensor depending on the sensor’s angle of view in relation to the objects.

One of the most useful improvements to the module would be the ability to consider foreign particles in the air that affect the emissivity of infrared emitting objects. Due to these pollutants, the final images that will be produced by sensors will contain errors. There are a number of obstacles to overcome in implementing this feature. Firstly, a formula needs to be devised on how much effect a certain particle in the air has on emissivity values. Some of the factors to consider include the type of particle, the particle density, the total area of the network that is affected and most importantly, the effect of the particle on infrared radiation. Some research in this area has already been conducted by the Directorate for Technical Evaluation of the French Ministry of Defense. The Directorate has built a simulation software called CHORALE [9] that allows the user to create virtual and realistic multi spectral 3 dimensional scenes and generate the physical signals that are to be received by sensors including infrared and acoustic sensors. A module for this software allows for the simulation of emitted radiance and the transmission of any pre-computed obscurant cloud in a virtual battlefield. More information on this module may be obtained at [22].

Although robust, there are limitations to implementing the sensor module in OPNET.
The implementation of imaging algorithms such as object recognition requires intensive calculations that may be better suited for a dedicated mathematical programming language such as MATLAB. Besides that, advanced simulations of thermal imaging sensors will need to factor in hardware issues such as signal processing elements that can be better represented by a system such as MATLAB’s Simulink. Therefore, an ideal advancement in this project would be to implement the sensor module as a black-box system in a more mathematical specific programming language. This black-box system will then be integrated as part of the thermal imaging sensor module in OPNET and perform as usual in the network. The user interaction will be limited to the OPNET module to ensure minimum complications to the user. The black-box system will be integrated into the process model of the thermal imaging sensor. An example implementation would have every Capture and Sleep state make calls to the black-box system by passing appropriate information using OPNET’s APIs.

The implementations of the other two sensor modules show that it is indeed feasible and easy to implement simple sensor modules using OPNET’s built-in features. Of the categories mentioned in Section 2.1, the ambient temperature sensor is part of the ambient sensors category and the positioning and ground speed sensor is part of the self-characterizing sensor category. These sensor categories are the simplest to implement due to the fact that neither need to distinguish between different data in each sample. However, these examples go to show that the implementation of other simple sensors such as ambient humidity, ambient noise level, sensor acceleration and sensor internal temperature are possible. Although the information provided by such sensors may not still not allow a sensor to be completely autonomous, at a minimum they still allow for less dependance on random processes.

Of all the sensor categories discussed, the only type of sensors that were not implemented were those from the intrusive object characterizing sensor category. The implementation of these sensors is certainly plausible using the thermal imaging sensor module as a basis. In the implementation of sensors of this category, the objects to be detected need
not emit packets of their own. Instead, they would need to receive packets from the sensors and alter the packets appropriately before retransmitting the packet in the direction of the sensor. In the packet, the sensor will need to query the objects for attributes such as their distance from the sensor and size. Because sensors of this category (for example, ultrasonic sensors) do not have a property remotely close to the emissivity of thermal imaging sensors, any modeling of discrepancies between the data produced by the sensor and the actual object need to be inserted externally into the packet by the simulator.

All-in-all, the creation of these sensor modules in OPNET is a novel approach that has immense prospects to finally allow the proper and full integration of fully functional sensors into sensor networks.
Bibliography


Glossary

A
AP  Access Point. , p. 95.
API  Application Program Interface. , p. 28.

B
BSS  Base Station Subsystem. , p. 95.

C
CTS  Clear To Send. , p. 95.

E
ESD  External System Definition. , p. 31.

N
NVSIM  Night Vision Simulator. , p. 3.

R
RF  Radio Frequency. , p. v.

123
RTS Request To Send., p. 95.

S

SYTHER Synthesis of Thermal Images., p. 2.

W

WLAN Wireless Local Area Network., p. 95.