Thesis Report

Ashwini Shamsundar

Technische Universiteit Delft

TU Delft

Modeling and Performance Evaluation of the Thread Protocol

Thesis submitted in partial fulfillment of the degree of

Master of Science

in

Electrical Engineering
(Telecommunications and Sensing Systems)

By

Ashwini Shamsundar
4521331

Monday 13th November 2017

Network Architectures and Services Group
Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
The Netherlands

In cooperation with

Delft University of Technology

PHILIPS

Dr. ir. Fernando A. Kuipers

Dr. Xiangyu Wang
Dr. ir. Esko Dijk

ॐ

श्री गणपतये नमः ॥

अज्ञानतिमिरान्धस्य ज्ञानाञ्जनशालाकया ।
चक्षुरुन्मीलितं येन तस्मै श्रीगुरवे नमः ॥

Om

Salutations to Lord Ganapati (Remover of Obstacles)

Salutations to noble teachers who remove the darkness of ignorance from our blinded (inner) eyes through the light of knowledge

Dedicated to

All my teachers who taught me at different stages of life

Contents

SUMMARY

Internet of Things (IoT) as the name suggests is a network of interconnected devices connected to the internet. There are lot of 802.15.4 network technologies in the market for current implementation of IoT, however, they have critical problems which is preventing IoT to become fully successful. Some of the issues are lack of interoperability, high power requirements, incompetency to use IPv6 communications and single point of failure systems. These issues led to the development of new networking technology called Thread. Thread is an IPv6-based wireless networking protocol built on open standards such as IEEE 802.15.4 and 6LoWPAN. It is designed for secure, low power 802.15.4 wireless mesh networks.

In this thesis, it is investigated whether Thread will be suitable for large-scale IoT lighting applications for Building and Home Automation systems.

1 Introduction

The Internet-of-Things (IoT) refers to a network of interconnected heterogeneous devices and systems connected to the Internet. The world has become more self-aware, due to intelligent connected systems. They provide convenience to remotely access, sense and manage everything from houses, buildings, transport to cities, health and disasters. Massive development in Wireless Sensor Networks (WSN) and Internet protocols have led to emergence of IoT for example, in Building Automation Networks.

A building automation network is a dedicated network of connected devices in a building. Connected devices range from smart displays, load control devices to smart lighting systems and intelligent metering systems, including software applications to monitor and control them. Building with systems are often referred to as smart buildings. Professional lighting systems are an essential part of smart buildings. In addition to energy efficiency, it is required that smart and connected lights have low maintenance, self-diagnose problems and will be able to wirelessly communicate when defective. These functionalities make smart buildings and homes more livable, safe and enjoyable [1]. Furthermore, with lighting being associated to the Internet of Things, Lighting-as-a-Service is evolving rapidly making professional lighting systems essential part of Building Automation Systems (BAS) [2].

With the advent of WSN, it became more advantageous to use them in building automation applications due to higher sensor mobility (providing flexible coverage), lowered installation costs and increased spatial resolution [1] when compared to wired sensor networks. V. C. Gungor et al. in [3] describe resource constraints, security issues, connectivity constraints, scalability issues (large-scale deployments) as some of the challenges for using industrial WSN. In order to combat these challenges, various protocols and standardizations for wireless sensor networks emerged at different layers of the Open System Interconnect (OSI) model, specifically for building automation[4] [5].

The Institute of Electrical and Electronic Engineers (IEEE) and Internet Engineering Task Force (IETF) provides WSN standards for internet protocols such as IEEE 802.15.4[6], 6LowPAN [7] [8], MPL [9] and RPL [10]. Competitive wireless technologies in 802.15.4 standard for WSN have one or more critical issues such as lack of interoperability, high

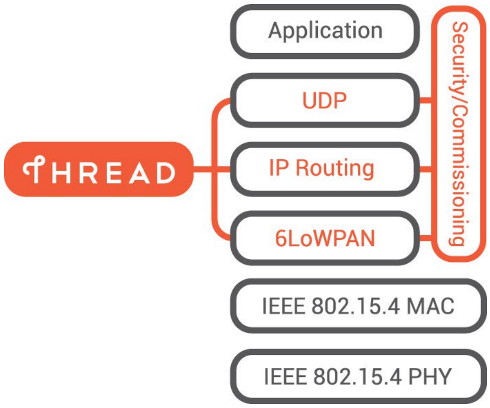


Figure 1.1: Thread Network Stack

power requirements, inability to communicate using IPv6, security issues and single point of failure systems. These challenges are preventing IoT using WSN to become a fully standardized technology.

THREAD PROTOCOL

The need to overcome the issues in one technology led to the development of new networking technology called Thread [11]. It is an industry-wide IPv6 protocol suite for IoT. Based on IEEE 802.15.4-2006 radio [6] and the IETF RFC 6LoWPAN standard [7], it promises large scale (mesh) network support and certified interoperability mainly suitable for Home Automation (HA) and Building Automation Systems (BAS). Figure 1.1 shows the different layers in the Thread network stack.

Thread addresses the issues of using IPv6 multicast for seamless end-to-end connectivity and interoperability to control (large) groups of actuators, for:

- low latency (< 200 ms) , and
- synchronous group actuation (< 50-100 ms)
- high mesh scalability due to efficient network capacity usage (250-node Thread network).

Furthermore, Thread can operate groups of actuators independently from the specific network technology used (Thread, Wi-Fi, LAN, etc.) and independently from network topology used. Additionally, Thread allows a node to automatically join the ‘best’ nearby mesh network and allows network managers to assign a node to a specific mesh without disrupting the application.

A TYPICAL THREAD NETWORK

A typical Thread network (see Figure 1.2) consists of the following components:

1. Border Router: Connects the Thread network to Internet and other adjacent networks. May also be responsible for commissioning and router management.

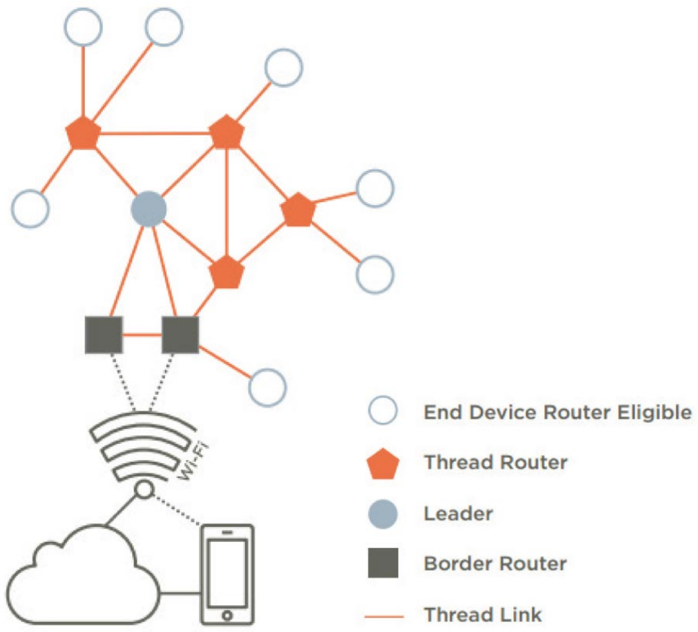


Figure 1.2: Thread Network and its components

2. Router: Responsible for providing joining, routing and security services in the network.

3. Router Enabled End Devices (REED): Non-routers in the network who behave as end devices and are hardware-capable of upgrading to Routers when required by the network.

4. Sleepy end devices: They are host devices who communicate only through their parent router.

1.1. PROBLEM STATEMENT

This work aims to answer whether the Thread wireless standard is suitable for large-scale professional IoT lighting applications, mainly in Home and Building Automated Systems. Deployment of large-scale IoT lighting systems in Building Automation Systems such as government and company buildings and educational institutions have a large number of lighting nodes typically in the order of hundreds to several thousands. A single building can contain multiple thread mesh networks with up to 250 nodes per mesh network. Through modeling and configuring of Thread based lighting system scenarios, the project conducts Thread performance evaluations in operational phase with different use cases.

Key performance indicators are essential to quantitatively determine Thread's aptness in lighting applications and to compare with other competitive protocols. In professional wireless lighting systems, packet loss rate, latency and stability of the network topology are the Key Performance Indicators (KPIs) and this project focuses on these KPIs only. The statistics defined based on KPIs (directly/indirectly) serve as the basis of our performance study of Thread.

The investigation comprises two main studies. They are highlighted as follows.

1. **Performance analysis of Thread Network model**. The study consists of modeling the Thread Network layer in two stages – Thread Multicast protocol for Low-Power and Lossy networks (MPL) model and full Thread Network model, respectively. The first stage comprises of modeling the MPL [9] according to Thread specification [11]. The second stage consists of modeling selected aspects of Thread Network layer consisting of network formation and router selection according to Thread specification [11]. Deployment scenarios with single source packet (seed node) in a rectangular network for node sizes of 100 and 250 deployed 3m apart and temporary power outage scenarios in partial and entire network determine the impact of latency on the network's performance. Number of Router-REED role changes during network formation scenarios will help in determining time taken for the network to converge. Unicast and MPL multicast packet loss rates in a network with/without overlapping broadcasts in single and multi-seed network models will also be analyzed.

2. **Comparison of Thread network model performance with existing technology**.

   The study consists of comparing MPL multicast performance of Zigbee network model with Thread Network model. MPL multicast packet loss rates in a network with or without overlapping broadcasts in single and multi-zone networks models are determined. The node scenario used is 101 nodes (with 1 seed node). The deployment scenario is used to compare latency in both models. Zigbee modified implementation model by Philips is used for this analysis, whose details are proprietary and cannot be disclosed.

   These studies will help determine if Thread is truly suitable for large-scale professional IoT Lighting applications in smart buildings.

1.2. PROPOSED APPROACH

In order to truly answer the suitability of Thread protocol for large scale professional IoT lighting applications, features in Thread network which help in determining protocol suitability are finalized. Low latency and high reliability & stability are also the requirements for BAS.

Key Performance indicators corresponding to the requirements of BAS are as follows:

- Packet loss rate: Packet loss rate in a Thread network is defined as the ratio of number of packets lost in the network to the total number of packets transmitted in the network. It is measured through Packet Delivery Ratio (PDR).

- Latency: Latency is the time delay in the network. In this study, latency is defined as the time taken for a transmitted packet to reach end node.It is measured through Multicast End-to-End delay and Multicast Forwarding delay.
- Network convergence (stability): Network Convergence is defined as the time taken for Thread network to form such that there is no frequent REED-Router roles interchange anymore for certain period of time.

The Thread stack is modeled in OPNET 14.5 tool using C programming language. OPNET is a network simulation tool set of Riverbed [12], which provides a suite of protocols and technologies to model, design, simulate and analyze communications networks. OPNET stands for OPtimized Network Engineering Tools. It is generally used to investigate the following aspects in communication networks [13]: (a) Application performance management; (b) Planning; (c) Engineering; (d) Operations; and (e) Research and Development.

OPNET can be used explicitly in Discrete Event Simulation (DES) mode or in hybrid simulation mode. In hybrid simulation mode, the network is partially with DES for accuracy and partially modeled mathematically.

Explicit DES mode is used for large-scale simulation. DES makes it possible for more accurate and realistic simulation as it generates remarkably detailed, packet to packet model to predict network activities. DES simulations are known to provide very accurate results for both simple and complex protocol [13]. Since this project is for large-scale scenarios, explicit DES mode is used for modeling and simulations.

Although the Graphical User Interface (GUI) in OPNET makes it simple to start with when compared to other network simulators, the OPNET Modeler complexity, lack of support materials and online assistance from Riverbed makes it difficult to use.

The research question is divided into sub-questions and will be answered through key performance indicators as follows:

1. Performance of unicast and multicast forwarding (MPL):
   (a) What is the packet loss rate in a network with/without overlapping broadcasts having multiple zones?
   (b) What is the packet loss rate in a single/multi zone network having overlapping broadcasts?
   (c) How will the radio link quality impact the network performance?
   (d) How will unicast/multicast routing impact the network performance?
   (e) How will combined unicast and multicast combination traffic impact network performance?
2. Latency:
   (a) Will the network converge? If yes,
      i. How long does it take to converge (convergence time)?
      ii. Will you have reliable (unicast and multicast) communication between the nodes during the process of convergence? If yes, for how long?

   (b) Impact of each scenario on convergence time
      i. Power on/off scenarios:
         - How will powering on different parts of networks at different time intervals (for example, half of network is turned on first at 0sec, other half after 10 sec) impact latency?
         - How will switching off different network parts at different time interval(for example, during power outage) impact latency?
      ii. Deployment scenarios:
         - How is router selection done during network formation and convergence?
         - Considering a new node in a network, what is the impact on it by the network to become a REED or Router?
         - How the Routers and REEDs interchange their roles in the network? How much time do they take?
         - What is the time taken for the network to be stable so that no further changes occur in the roles of nodes?
   (c) How long does it take for the lights to turn on after the switch is turned on?

3. Convergence
   (a) Topology of network
      i. How will the power on/off scenarios impact the topology formation of network?
   (b) How many role changes (Router and REED) are needed in the network before converging?
   (c) How many messages are transmitted in the network before converging?

1.3. OUTLINE

Chapter 2 describes the competing protocols and the work related to Thread. Chapter 3 gives a detailed description of the Thread protocol standard. Chapter 4 consists of modelling and implementation of the Thread network layer using Opnet 14.5 tool for different scenarios, assumptions and use cases. Chapter 5 consists of simulation results and detailed performance analysis for predefined scenarios along with various challenges faced. Chapter 6 concludes this thesis study and provides vistas for future work.

2

Related Work

2.1. Introduction

In this era of a digital and more connected world, various factors and demands triggered the emergence of wireless sensor networks (WSN). A major factors include

1. Escalation of computing and electronic devices in the lives of people due to swift advancement in semiconductor technology and miniaturization.
2. Exponential growth in the processing power of micro-controllers.
3. Advancement and convergence of wireless communications, digital electronics and electro-mechanical systems technology.
4. Signal sensing and conditioning integration into small sensor nodes capable of measuring and storing data through complex processing techniques.
5. Meteoric growth and advancement of wireless technologies, mainly for low-power and short range applications. [14]

A wireless sensor network is a distributed network of wireless sensor nodes with different topologies in order to enable sensing and actutation. These wireless sensor nodes are small in size and capable of communicating with each other within a short radio range. Since they are wireless, mobility can be easily supported as well. As a consequence of these striking features on the wireless sensor nodes, there is a increased adoption of WSN in commercial, industrial and professional applications [15].

Due to the increasing applications and usage, IEEE standardized physical and medium access layer for WSNs called the IEEE 802.15.4 (Low Rate Wireless Personal Area Network) standard [6]. This standard defines the operation of low - rate wireless personal area networks. While this standard defines how nodes may communicate, it does not define how to network them. Networking these nodes is an important and challenging aspect as the network needs to be low power while enabling reliable communications over lossy wireless channels. There are several networking standards such as ZigBee, Z-Wave and Thread. Zigbee and Z-wave will be described in the next section.

2.2. PROTOCOLS IN WSN

This section describes ZigBee and Zwave in the context of Home Area Networks (HAN) and Building Area Networks (BAN).

2.2.1. ZIGBEE

Zigbee is an open wireless standard defined by the Zigbee alliance [16] for low power and lossy WSN. Zigbee builds on the physical layer and media access control defined in IEEE standard 802.15.4, in order to create a wireless mesh network. The Zigbee specification defines the network layer and application layers. The Zigbee network layer natively supports both star and tree networks, and generic mesh networking. Every network must have one coordinator device. Within star networks, the coordinator must be the central node. Both trees and meshes allow the use of Zigbee routers to extend communication at the network level. The network layer handles the functions of addressing, neighbor discovery and routing. Typically, a variant of ad-hoc on-demand distance vector routing (AODV) protocol is used for routing data.

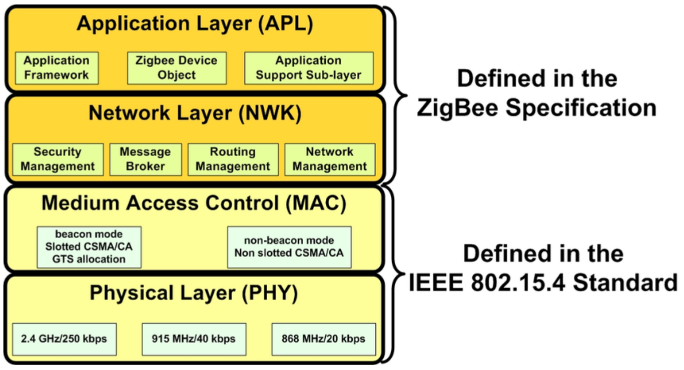


Figure 2.1: ZigBee network stack

The Connected Lighting Alliance (TCLA) has declared Zigbee 3.0 to be the preferred standard for lighting automation in connected home and buildings [17]. Figure 2.1 shows a Zigbee stack for a building network.

2.2.2. Z-WAVE

Z-Wave is another wireless standard for smart homes developed by Z-Wave alliance [18]. Z-wave provides application level inter-operability between the devices. The protocol stack of Z-wave has 3 layers: radio layer, network layer and application layer. Figure 2.2 provides an overview of the Z-Wave stack.

Z-Wave creates a wireless mesh network. This architecture uses source routing and can support up to 232 nodes on a single network.

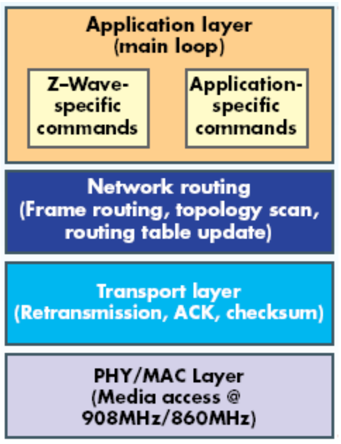

Application layer (main loop)
Z-Wave-specific commands
Application-specific commands
Network routing (Frame routing, topology scan, routing table update)
Transport layer (Retransmission, ACK, checksum)
PHY/MAC Layer (Media access @ 908MHz/860MHz)

Figure 2.2: Z-Wave network stack

2.3. COMPARISON OF COMPETITIVE PROTOCOLS WITH THREAD

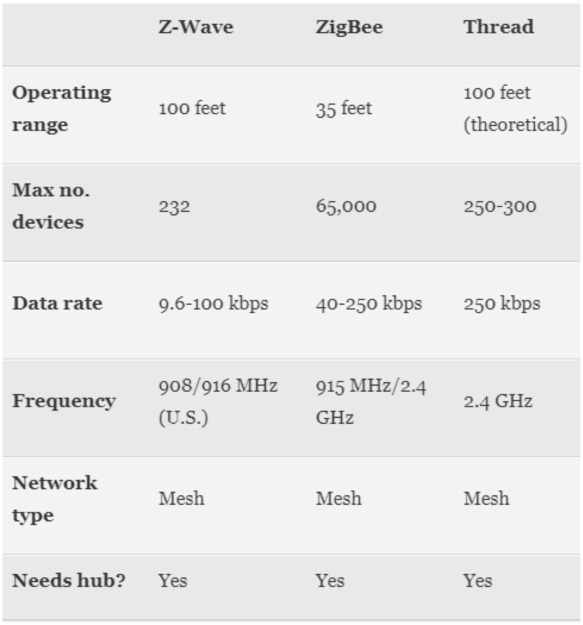
In the previous section, introduction to different wireless protocols used in IoT applications were discussed. In this section, various parameters such as operating range, topology type, security, interoperability, etc. of these protocols will be compared with that of Thread.

Figure 2.3 gives comparison between these protocols. As seen from the figure, Thread and Zigbee standards comply with the requirements of Lighting automation in IoT applications. However, Thread provides IPv6 connectivity and device to device interoperability unlike Zigbee. It will be investigated later in this thesis whether Thread is better than Zigbee for multicast scenarios. Additionally, Thread is not designed to replace any technology but to work in coexistence with other technologies in IoT systems.

2.4. THREAD PROTOCOL RELATED WORK IN LITERATURE

Lan et al. [19] performed an experimental evaluation of Thread wireless protocol with 23 FRDM-KW24D512 nodes from NXP. The evaluation is with respect to signal coverage, unicast and multicast latency, reliability, and availability. All the results presented are empirical based on limited evaluation in one building.

From the results, it can be seen that the nodes running S-MAC have better coverage as S-MAC is deemed more light-weight than Thread. Furthermore, for unicast packets, only 5% of the packets reach within 100ms round-trip time for 5 hops. However, for 200ms round-trip time, 6 hops works well. Multicast traffic was shown to have been disseminated within 200ms and with a high reliability percentage of over 98% in a one-off study. An availability test was also done, which shows that the network is highly available.



	Z-Wave	ZigBee	Thread
Operating range	100 feet	35 feet	100 feet (theoretical)
Max no. devices	232	65,000	250-300
Data rate	9.6-100 kbps	40-250 kbps	250 kbps
Frequency	908/916 MHz (U.S.)	915 MHz/2.4 GHz	2.4 GHz
Network type	Mesh	Mesh	Mesh
Needs hub?	Yes	Yes	Yes

Figure 2.3: Comparison between different technologies

2.4.1. OPENTHREAD

OpenThread [20] is an open source implemenatation of the Thread networking protocol, released by *Nest Labs* [21]. It implements all the features defined in *Thread 1.1.1* specification and can be used for certification by the Thread group. It includes all Thread networking layers - IEEE 802.15.4 with MAC security, IPv6 and 6LoWPAN. Additionally, it implements device roles, Border Router, Mesh Link Establishment and Mesh routing. OpenThread can be ported directly to new hardware platforms.

Gonzalez H.G. [22] from *Universitat Politecnica De Catalunya* in 2017, studied the Thread protocol specifically for home automation. The study used OpenThread and conducted a series of tests using the OpenMote Development Kit. The thesis states, OpenThread to be developing with some missing features while many are validated only quite recently (in 2017). However, Gonzalez used an uncertified version of OpenThread 1.0 implementation for his study. Furthermore, only simple broadcast message tests were conducted for a small home automation network. Hence, the results are not very useful for the current study.

2.4.2. SIMPLIFIED MATLAB MODEL OF THREAD

A simplified model of Thread has been implemented in Matlab for basic simulation studies for usage within Philips Lighting. In this model, the routers and REEDs are implemented.

During the start of network formation, the router adopts the role of Border Router and all the other nodes connect with that node. As this is a simplified model, the nodes

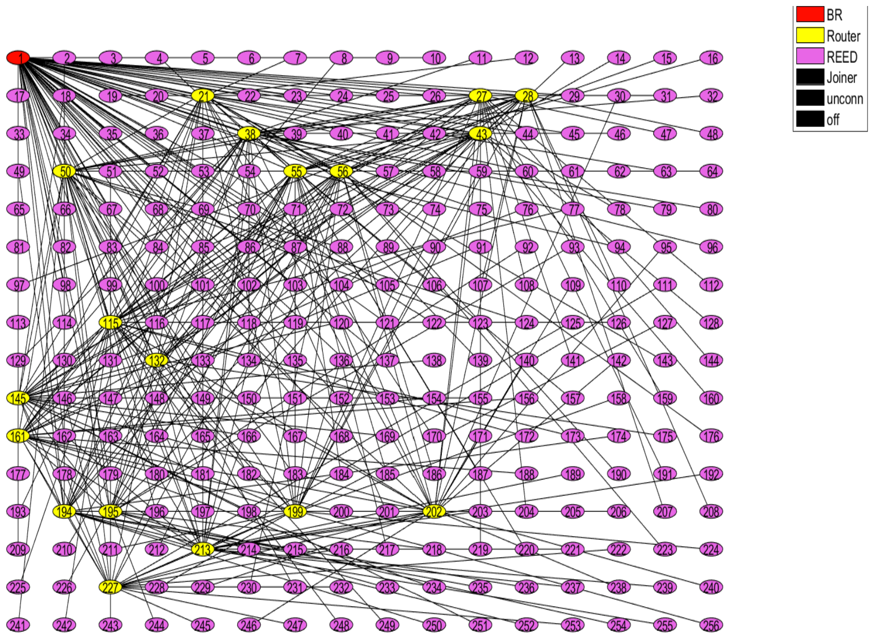


Figure 2.4: Thread network simulation using the Matlab model.

have a global knowledge of the network. Furthermore, the border router is not considered to connect to an external network. The nodes actively monitor and convert into router till there are 15 routers in the network, after this it becomes more careful and follows more elaborate procedure to become a router. The average number of routers in the network is in the range of 16-23. Maximum limit for number of routers is 32. A screenshot of a simulation using this Matlab model is shown in Figure 2.4.

In this simplified model, several aspects are not considered as listed here.

- Link quality between nodes is not considered.
- The transmission and propagation of the “packets" are not considered: It is assumed that the transmitted data are directly delivered to the destination.
- Leader role mentioned in the Thread specification is not considered

This model is the basis for Thread model implementation in OPNET Modeler 14.5. It is useful to determine the routers and REEDs in a network for a specific scenario. This information is used during MPL study of Thread network in OPNET 14.5.

3
Thread Architecture

3.1. Introduction

In order for IoT to be successful, right technology for multi sensing, embedded processing and connecting are required to create smart solutions for wide variety of applications. These solutions must include the right software, hardware and support structure to enable and bring the product to market quickly.

Thread is one of the mesh networking layer protocols adhering to the IEEE 802.15.4 standard. It implements IP addressing to the end node. In this chapter, Thread stack and architecture will be studied. Additionally, different components in the Thread network are also discussed.

3.2. Features of Thread

Some of the characteristics of Thread are as follows:

1. Reliable networks: Thread networks allow systems to self configure and fix routing problems. They use simple protocols for forming, joining and maintaining the networks.

2. Secure networks: Thread uses encryption standards, which are used in banking applications. The devices do not join the Thread network unless authorized and communications are encrypted.

3. Robust: No single point of failure due to mesh topology.

4. Low power: Thread supports battery-operated devices to be part of network. AA type batteries with suitable duty cycles are used such that the devices can operate for several years.

5. Fast time to market: Since Thread is based on open standards, it supports devices which work on IEEE 802.15.4 and 6LowPAN. It does not need special hardware.

3.3. THREAD COMPONENTS OVERVIEW

There are different types of devices in a thread network. Figure 1.2 shows these components. They are as follows:

1. Border Router: A Border Router is a gateway, which provides connectivity for thread network to access other adjacent networks such as Wi-Fi, Ethernet, etc. Additionally, Border Routers provide services for devices within the 802.15.4 network, including routing services for off network operations. A Thread Network typically contains one or more Border Routers.

2. Router: Routers provide routing services to network devices. Routers also provide joining and security services for devices trying to join the Thread Network. Routers are not designed to sleep. Routers can downgrade their functionality and become REEDs (Router-Eligible End Devices).

3. Router Enabled End Devices: REEDs can become routers but due to the network topology or conditions these devices are not acting as routers. As such, a REED is not a specific device type but a state of a routing-capable device when in the Thread Network. These devices do not forward messages or provide joining or security services for other devices in the network. If necessary, the network manages the transition of a device from REED to router without user interaction.

4. Sleepy End Devices: Sleepy End Devices (SEDs) are host devices. They communicate only through their parent router and cannot forward messages for other devices.

3.4. THREAD ROLES OVERVIEW

Devices participating in a Thread network can take up various roles depending upon their type and configuration of the Thread network join process. A Thread network join process is called Thread Commissioning. The different roles Thread devices are described in the subsequent sections.

1. Joiner: The device to be added by a human administrator to a commissioned Thread Network. This role requires a Thread interface to operate and cannot be combined with another role in one device. Most importantly, the Joiner does not have network credentials.

2. Joiner Router: An existing Thread router or REED (Router-Eligible End Device) on the secure Thread Network that is one radio hop away from the Joiner. The Joiner Router requires a Thread interface to operate, and may be combined in any device with other roles except the Joiner role.

3. Leader: The single distinguished device in any Thread Network Partition that currently acts as a central arbiter of network configuration state. The Leader requires a Thread interface to perform and may be combined in any device with other roles except the Joiner.

4. On-mesh Commissioner: A combined role for certain collapsed cases where the Commissioner has a 15.4 interface, and possesses the Thread Network Credentials. An On-mesh Commissioner is always both a Commissioner and its own Border Agent. An On-mesh Commissioner may also be a Joiner Router.

5. Native Border Agent : A specific term for a device that is serving the role of Border Agent for a Native Commissioner. Such a device needs only a IEEE 802.15.4 radio [6] to bridge between the unsecured 802.15.4 external neighbors and the secured Thread Network. As such, any Joiner Router MAY also serve as a Native Border Agent role.

6. Native Commissioner: A Commissioner or Commissioner Candidate that has the same interface used by the mesh. In the case of Thread, this would be an IEEE 802.15.4 radio [6]. Unlike an On-mesh Commissioner, a Native Commissioner does not possess the Thread Network Credentials.

3.5. ASSUMPTIONS AND OTHER DEFINITIONS

1. Thread devices must be authenticated and authorized to be part of a Thread network. This process is called joining or commissioning of nodes in the network.

2. Thread Network Partition: A connected group of nodes that operates independently of any other nodes in the network. Each Thread Network Partition has its own leader.

3. Two Thread network partitions are said to be part of same thread network if they have the same domain ID.

4. In any Thread partition,the maximum number of routers possible is 32 (maximum hopcount is 32).

3.6. THREAD STACK

Thread is an open IP based networking standard which is designed specifically for connected home appliances. It is based on the IEEE 802.15.4 MAC and physical layer operating at 250 kbps in the 2.4 GHz band. Thread uses simple protocol for forming, joining and maintaining the network.

Devices cannot join the thread network if they are unauthorized and the communications are unencrypted. An elliptic curve variant of J-PAKE (EC-JPAKE) using NIST P-256 elliptic curve is used for authentication and key agreement. The topology type used is mesh. Due to this, there is no single point of failure.

The devices in network are powered with AA type batteries with suitable duty cycles which are used such that the devices can operate for several years.

Figure 3.1 shows the Thread network layers corresponding to the standards used in each layer of the stack. The following sections discuss functions at each layer. Concepts relevant to this thesis study are discussed in detail.

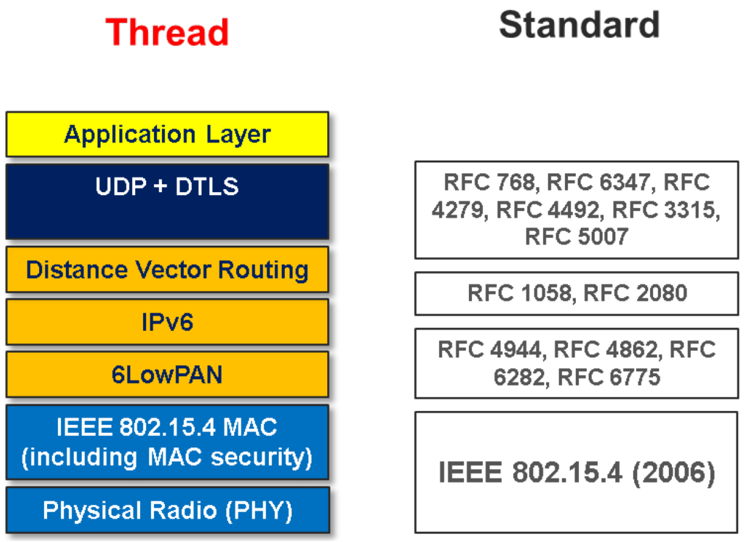

Thread
Standard
Application Layer
UDP + DTLS
Distance Vector Routing
IPv6
6LowPAN
IEEE 802.15.4 MAC (including MAC security)
Physical Radio (PHY)
RFC 768, RFC 6347, RFC 4279, RFC 4492, RFC 3315, RFC 5007
RFC 1058, RFC 2080
RFC 4944, RFC 4862, RFC 6282, RFC 6775
IEEE 802.15.4 (2006)

Figure 3.1: Thread network stack along with standards used at each layer

3.6.1. IEEE 802.15.4

IEEE 802.15.4 standard is suitable for home networking and for wireless sensor network applications. It defines the physical layer and Media Access Control(MAC) layer of the Open System Interconnect (OSI) model. The data rates offered are 250 kb/s, 40 kb/s and 20 kb/s. The standard is mainly used for star, mesh and peer-to-peer operations. This fully handshaked protocol provides good data transfer reliability. Additionally, it supports low latency devices and has low power consumption. It operates at the frequency of 2.4 GHz ISM band with 16 channels, in 915MHz ISM band with 10 channels and in European 868 MHz band with 1 channel.

The physical layer is used for transmission/reception of packets over physical medium with activation and deactivation of radio trans-receiver. Receiver energy detection is used as an estimate of received power within bandwidth of channel. It is also intended to be used by network layer for channel selection.

Link quality indication is done in physical layer to characterize the strength/quality of received packet. It is implemented using energy detection (ED), SNR or a combination of both. Clear channel assessment is done using techniques such as energy detection above ED threshold, Carrier Sense only and carrier sense with energy above threshold.

The MAC layer in 802.15.4 standard is responsible for contention management of data, error correction, channel acquisition and addressing.

3.6.2. 6LOWPAN

Low power wireless personal area network (LoWPAN) is a simple low cost network which provides wireless connectivity to devices with limited power that conform to the IEEE 802.15.4 standard. The devices used are usually limited in their computational power, energy availability and memory. IPv6 over Low-power Wireless Personal Area Networks

(6LoWPAN) layer provides adaptation between IP layer and 802.15.4 MAC layer. IPv6 packet is fragmented when it is passed from IP layer to 802.15.4 MAC layer. Packets are reassembled into IPv6 format when passed to IP layer from 802.15.4 MAC layer.

6LoWPAN is suitable for low powered (mainly battery run) devices having low bandwidth and requiring small packet size. The maximum size of physical layer packet is 127 bytes. Different data rates which can be used with 6LoWPAN are 250 kbps for 2.4 GHz, 40 kbps for 915 MHz and 20 kbps for 868 MHz which is also supported by IEEE 802.15.4 physical layer. Additionally, 6LoWPAN supports star and mesh topologies.

6LoWPAN Layer is defined in IETF RFC 4944 [7] and RFC 6282 [8]. Summary of functions in 6LoWPAN adaptation layer are as follows:

- LOWPAN_IPHC header compression and decompression is specified in IETF RFC 6282. It reduces overhead of the large IPv6 header. By selecting among no compression, stateless compression (i.e., SAC = 0, DAC = 0), or stateful compression (i.e., SAC = 1, DAC = 1) modes, and choosing the needed address compression mode (i.e, SAM and DAM), packet sizes change accordingly.

- A node with IPHC compression drops the packet if received context ID is different than its own. This impacts how the received statefully compressed 6LoWPAN packets are processed.

- Page 14 of [23] states a link- local address is sufficient to communicate with nodes in direct radio communication. But a routable address is required to communicate with devices that are multiple hops away. Therefore, three compression methods are assumed as follows:

  - Compression Mode 1 (CM1): 64-bit needed to complete the address
  - Compression mode 2 (CM2): 16-bit needed to complete the address
  - Compression mode 3 (CM3): 0-bit needed to complete the address (IID derived from MAC)

  This results into the following possible cases:

  1. Stateless IPHC
     - Link local : Source Address and Destination Address use CM1,CM2, CM3 (statically decided).
     - Global: Source Address and Destination Address fully specified (128 bits) and CM is ignored.
     - Multicast: Destination Address can be 128, 48, 32 or 8 bits, based on the type of address (for now left undefined).
  2. Stateful IPHC
     - Global: Source Address and Destination Address use CM1,CM2, CM3 (statically decided)
     - Multicast: Unicast-Prefix-based IPv6 Multicast addresses (48 bits)

In Chapter 4, attributes configured and implemented according to Thread specification are discussed.

3.6.3. MESH LINK ESTABLISHMENT

Mesh Link Establishment (MLE) protocol enables a node to periodically multicast the estimate of the quality of links to the neighboring nodes. It allows the node to dynamically configure and secure radio links, and detect unreliable links before configuring them. Thereby, efficiently estimating link reliability without two way message exchanges. MLE messages are transported using single-hop link local unicasts and multicasts between Routers.

All routers in a Thread network periodically exchange single-hop MLE advertisement packets which contains link and path cost information of all neighboring routers. All the routers update the path cost information to any other router in the network through these messages. Routers can dynamically make a selection on the next most suitable route to the destination if a route is no longer usable. This enables the routers to determine other routers who are dropped off from the Thread network.

The relative signal in dB received above the noise floor is defined as the link margin. The link quality in each direction is based on the link cost on incoming messages from that neighboring device. This incoming link margin is mapped to a link quality from 0 to 3. One-way link quality is shared in the MLE advertisement message, and the nodes calculate the two-way link quality after receiving the MLE unicast response. A minimum of two-way link quality is taken and the corresponding value is mapped with the Table (see Table 3.1) to calculate the link cost.

Link margin	**Link quality**	**Link cost**
> 20dB	3	1
> 10dB	2	2
> 2dB	1	4
≤ 2dB	0	∞

Table 3.1: Link metrics

MLE in the specification also defines link configuration, parameter dissemination and neighbor detection in Thread network. Additionally, it defines Type-Length-Value (TLV) command formats to network formation and maintenance including parent selection, address solicitation, and router management.

MLE PARENT SELECTION

Unconnected node in a network sends a multicast Parent Request message. This message is sent to a link-local all routers multicast address (FF02::2) with a hop limit of 255. The following TLVs are present in the Parent Request packet format:

- Mode TLV
- Challenge TLV
- Scan mask TLV (See Figure 3.2)): R and E bits determine whether only active routers in the network should respond to the parent request or all the routers and REEDs in the network should respond.

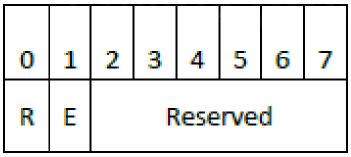


Figure 3.2: Scan Mask message format

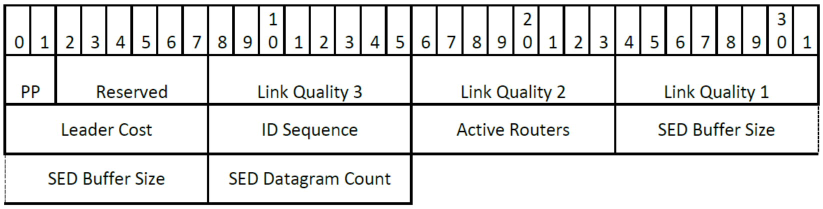


Figure 3.3: Connectivity TLV format

- Version TLV.

Once an active router or REED (depending on Scan Mask TLV setting) receives the parent request message, parent response unicast message is sent out to the sender node provided the following conditions are not satisfied:

- It has not available child capacity (if max child count minus child count would be equal to zero) OR
- It is disconnected from its Thread Network partition (that is, it has not received an updated ID sequence number within LEADER_TIMEOUT seconds) OR
- Its current routing path cost to the Leader is infinite (it is not connected to the network).

Parent response unicast message consists of the following packet format.

- Connectivity TLV
- Link Margin of sender node (see Figure 3.4)
- Quality of sender node (Outgoing Link quality in sender's node view).

The connectivity TLV message format is shown in Figure 3.3. The fields of the message are as follows.

- PP - Parent Priority
- Link Quality N (see Table 3.1 - The number of neighboring device with which the sender shares a link of quality N.

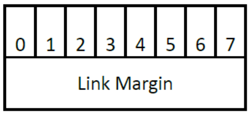


Figure 3.4: Link Margin TLV

- Leader Cost - The sender's routing cost to the Leader. This field is zero if the sender is the Leader itself. If the actual routing cost is infinite (normally represented as zero), no message is sent.
- ID Sequence - The most recent ID sequence number received by the sender.
- Active Routers - 8-bit unsigned integer indicating the number of active Routers in the sender's Thread Network Partition.
- SED Buffer Size - Optional 16-bit unsigned integer indicating the guaranteed buffer capacity in octets for all IPv6 datagrams destined to a given SED.
- SED Datagram Count - Optional 8-bit unsigned integer indicating the guaranteed queue capacity in number of IPv6 datagrams destined to a given SED.

In addition to this Source Address TLV is also sent. Leader Data TLV, Link-layer frame Counter TLV, MLE frame Counter TLV, Response TLV, Challenge TLV and Version TLV are part of the parent response unicast message.

The parents' response is received at the sender node. It consists of Link Margin TLV. Through RSSI of parent's response the link Margin and link quality (incoming Link quality in sender's view) is calculated and the node forms a Link Set tuple for each Router and REED as shown below:

(L_router_id, L_link_margin, L_incoming_quality, L_outgoing_quality, L_age)

where, L_router_id is the Router ID assigned to that neighbor; L_link_margin is the measured link margin for messages received from the neighbor; L_incoming_quality is the incoming link quality metric as calculated from; L_link_margin; L_outgoing_quality is the incoming link quality metric reported by the neighbor for messages arriving from this Router, and L_age is the elapsed time since an advertisement was received from the neighbor.

The 2-way Link Quality is calculated at the sender's node as

$$min(\text{Incoming Link Quality}, \text{Outgoing Link Quality}).$$

A parent with better parent priority and 2-way Link Quality is preferred. If two parents have same 2-way Link Quality, the one with higher Link Quality 3 neighbors in Connectivity TLV is preferred. If multiple parents have same number of Link Quality 3 neighbors, the child selects one of the parent provided that the number of child for that parent doesn't exceed the child threshold (10 for all other routers and 64 for Border Router).

Once the sender node selects its parent, it sends confirmation to that parent through unicast Child ID request message and it is assigned a child ID by the parent. Through this, the sender node knows its parent, though the 2-way Link quality, the sender node can also make an estimate about its direct neighbors and have a table based on that.

MLE Neighbor Detection

A router sends Link Request MLE message to establish a link to a neighboring router. The Link Request message consists of Source Address TLV, Challenge TLV, Leader Data TLV, Version TLV, and TLV Request TLV: Link Margin. The neighboring router responds with Link Accept and Request MLE message, where the neighboring router accepts a requested link and request a link with the sender of the original request if it satisfies all the conditions. The Link Accept and Request message consists of Source Address TLV, Leader Data TLV, Response TLV, Link-layer Frame Counter TLV, Version TLV, Link Margin TLV, and the optional fields including MLE Frame Counter TLV, Challenge TLV, and TLV Request TLV: Link Margin.

In case the conditions are not satisfied, neighboring router responds with Link Reject MLE message with Status TLV.

3.6.4. Network Layer

Thread defines network layer with IPv6 addressing architecture, multicast addressing and forwarding. Thread protocol defines two scopes of addresses for multicast transmissions - Link-Local scope and Realm-Local scope.

- Link-Local scope refer to all the Thread interfaces- Border router, Routers, REEDs, Sleepy End devices, which can be reached with single radio transmission.

- Realm-Local scope refer to all the Thread interfaces within a Thread network. Thread implements Multicast Protocol for low power and lossy networks (MPL), for Realm-Local scope multicast messages.

MPL is implemented as described in RFC 7731 [9] and RFC 6206 [24] with certain modifications. There are different Realm-Local scope addresses which Thread interfaces should be subscribed to. They are as follows:

1. FF03::1 - This is realm-local all-nodes multicast address. All the thread interfaces must be subscribed to it. [11]

2. FF03::2 -This is realm-local all-routers multicast address. All the Border Routers, Routers and REEDs must subscribe to it.

3. FF03::FC - This is realm-local ALL_MPL_FORWARDERS address. This address is to forward multicast packets outside the scope of realm-local. Hence, all the devices must subscribe to and process the multicast messages sent to this address.

Thread MPL considers all the realm-local addresses as MPL domain address. This facilitates Thread devices to forward multicast messages to any arbitrary address within realm-local scope without IP-to-IP encapsulation.

MPL PROTOCOL IN THREAD

This section discusses the MPL protocol used in Thread specification. MPL protocol defined in [9], cater IPv6 multicast forwarding in resource constrained networks. Thread MPL protocol works on Trickle algorithm [24] to periodically transmit multicast messages. There are two strategies in MPL to propagate messages - Proactive and Reactive forwarding.

- Proactive forwarding: In this forwarding mechanism, the forwarders schedule MPL message transmissions using Trickle algorithm. The forwarder will be unaware if its neighboring nodes have received the message or not [9]. Message forwarding is terminated after MPL messages are transmitted limited number of times.

- Reactive forwarding: In this forwarding mechanism, MPL link local- multicast control messages are transmitted according to Trickle algorithm . The forwarder has the ability to discover the messages which are not received by its neighbors. It then triggers trickle process only for those messages.

MPL in Thread uses only proactive messages to propagate messages. More on MPL and Trickle algorithms will be discussed in Chapter 4.

3.6.5. TRANSPORT LAYER

Thread implements User Datagram Protocol (UDP) as in RFC 768 [25] and RFC 1122 [26]. It is to be noted that Thread does not omit checksums when using RFC 6282. Implementation of TCP is optional in Thread devices but RFC 793 [27] and RFC 1122 can be used to implement TCP on Thread devices.

4

Thread Model Implementation

4.1. Introduction

A smart connected lighting network (called scenario) is designed over a Thread network model comprising of N Thread nodes as luminaries,

$$\text{where } N \in \{4, 10, 20, 32, 64, 100, 125, 150, 200, 256, 512, 1000\} \tag{4.1}$$

Thread network model presented in this chapter is a discrete event simulation program that models Thread protocol stack, shown in Figure 3.1, in OPNET Modeler 14.5 tool. Figure 4.1 shows a Thread network scenario with 256 Thread nodes.

Thread network model consists of Thread nodes, implementing the communication protocols necessary to deliver data throughout the network. Each Thread node is comprised of Thread node model as shown in the Figure 4.2. It consists of simplified higher-layer protocol stack modeled by a traffic source and a sink, a network layer implementing MPL and network formation protocols, a 6LoWPAN adaptation layer and IEEE 802.15.4 MAC connected to radio transmitter and receiver modules.

Following sections in this chapter details layer-wise process models implemented in the Thread node model. Some important assumptions, to be noted before proceeding to the next section are as follows:

1. A node is assigned a unique integer number called node ID, which is valid at all levels of the model.

2. Both IP and MAC addresses coincide with the Node ID.

3. All Thread nodes are assumed to be commissioned into the network through Mesh Commissioning Protocol. The protocol itself is not implemented in this model.

4. All nodes are considered to be unconnected in the network until they become a Thread router or child (REED) of a Thread router in the network.

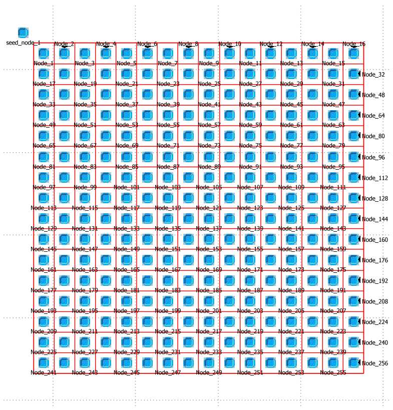


Figure 4.1: Grid placement of nodes and single seed node for a 256 node Scenario

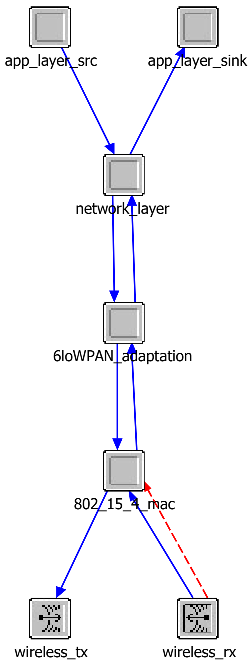


Figure 4.2: Node model implemented in OPNET

Table 4.1: Physical Layer parameters

Physical Layer Parameters	Assigned Values
Packet reception-power threshold	-85 dBm
Path loss exponent (γ)	3
Transmit Power	0 dBm
Radio range (calculated)	30.8175m

4.2. IEEE 802.15.4 PHY AND MAC

A simplified version of IEEE 802.15.4 Medium Access Control (MAC) and Physical Layer (PHY) specifications have been implemented in this work. The implementation provides channel access and (optionally acknowledged) frame delivery for nodes within communication range.

4.2.1. IEEE 802.15.4 PHY

Standard IEEE 802.15.4 transmitter and receiver modules of OPNET are chosen as the base model for this implementation. The standard model and its attributes are modified according to Thread specification 1.1.1. The modeled protocol operates by transmitting data at 250 kb/s on Channel 26, with a center frequency of 2480 MHz and a bandwidth of 5MHz. Table 4.1 shows different parameters which can be configured in Physical layer and the values assigned for each. This is calculated according to the following path loss equation:

$$P_r = P_t \left(\frac{c}{4\pi f}\right)^2 \left(\frac{1}{d}\right)^\gamma \tag{4.2}$$

where, P_r = Received power (in Watts) at each node
P_t = Transmitted power (in Watts) from each node
f = transmitted frequency (in Hz)
c =speed of light in vacuum (m/s)
γ =path loss exponent

Here we assume the antenna gains $G_{at} = G_{ar} = 1$ and reference distance $d_0 = 1$m. Hence, the variables are not shown in the equation.

Various pipeline procedures are created to establish link between the wireless transmitter and receiver modules with 802.15.4 MAC layer through different models. All the pipeline procedures are standard defined by OPNET Modeler tool with modifications according to Thread specification. The pipeline procedures for all the models in wireless transmitter module are listed below:

1. dra_rxgroup : This pipeline procedure is capable of determining the possibility of radio interaction between a given transmitter channel and a given receiver channel.

2. dra_txdel : This pipeline procedure is capable of computing transmission delay associated with the transmission of a given packet.

3. thread_dra_chanmatch: This pipeline procedure is capable of dynamically determining the ability of a given radio transmitter channel to reach a given radio receiver channel.
4. dra_propdel: This pipeline procedure is capable of computing the propagation delay associated with the transmission of a given packet toward a given receiver.

Pipeline procedures for all the models in the wireless receiver module are as follows:

1. dra_ragain: This pipeline procedure is capable of computing antenna gain associated with the receiver's antenna for a particular incoming radio transmission.
2. dra_power: This pipeline procedure is capable of computing the received power level for an incoming radio transmission.
3. dra_bknoise : This pipeline procedure is capable of computing background noise affecting incoming radio transmissions.
4. dra_innoise: This pipeline procedure is capable of computing interference noise affecting a particular incoming radio transmission.
5. thread_dra_snr : This pipeline procedure is capable of computing signal to noise ratio for a particular incoming radio transmission.
6. thread_dra_ber : This pipeline procedure is capable of computing the expected bit error rate for an incoming radio transmission.
7. dra_error : This pipeline procedure is capable of computing the number of bit errors in a segment of an incoming radio transmission.
8. thread_dra_ecc : This pipeline procedure is capable of determining the acceptability of an incoming radio transmission.

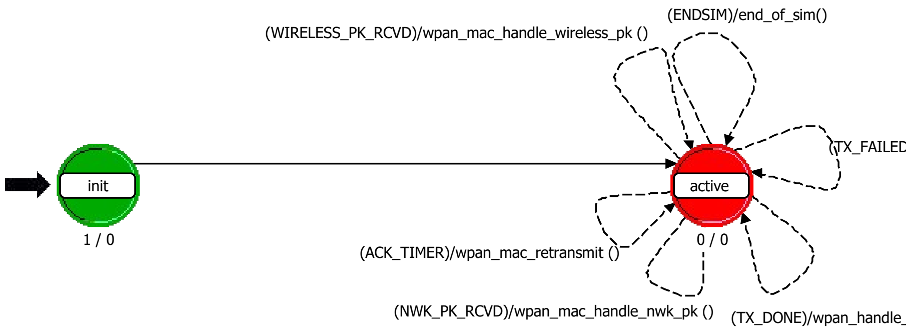
4.2.2. IEEE 802.15.4 MAC

The wireless medium is accessed through un-slotted CSMA CA mechanism. Beacon management, GTS management, frame validation, association and disassociation functionalities are omitted in the implementation. This OPNET process model was originally used for Zigbee model by Philips. Hence, some of the functionalities related to Zigbee are eliminated and statistics and functionalities related to Thread is added. Figure 4.3 shows process model of the simplified MAC model implemented in OPNET Modeler.

CSMA/CA mechanism is implemented as a child process to the simplified MAC model. It implements CSMA/CA algorithm for data transmission according to IEEE 802.15.4 specifications. The back-off exponent (BE) is reset for every new packet. Figure 4.4 shows the process model for CSMA/CA algorithm in the MAC layer.

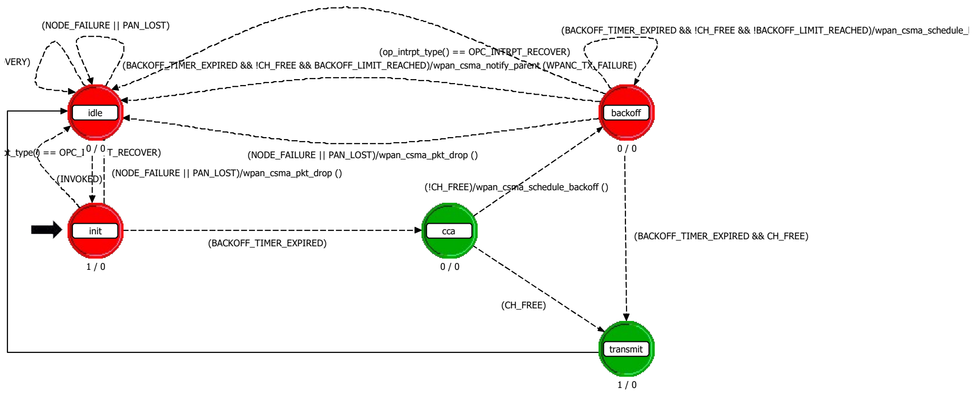
When a node finds the channel busy during the CSMA/CA procedure, the Number of Back-offs (NB) is increased. The back-off exponent (BE) is reset to initial value (Minimum Back-off Exponent) for every new packet. Minimum Back-off Exponent value is set to 3 and Maximum Number of Back-offs are set to 4 in this algorithm.

Features in simplified process model of MAC are as follows:



(ENDSIM)/end_of_sim()
(WIRELESS_PK_RCVD)/wpan_mac_handle_wireless_pk ()
(TX_FAILED
init
1 / 0
active
0 / 0
(ACK_TIMER)/wpan_mac_retransmit ()
(NWK_PK_RCVD)/wpan_mac_handle_nwk_pk ()
(TX_DONE)/wpan_handle_

Figure 4.3: Process model of IEEE 802.15.4 MAC in OPNET



(NODE_FAILURE || PAN_LOST)
(BACKOFF_TIMER_EXPIRED && !CH_FREE && !BACKOFF_LIMIT_REACHED)/wpan_csma_schedule_
VERY)
(op_intrpt_type() == OPC_INTRPT_RECOVER)
(BACKOFF_TIMER_EXPIRED && !CH_FREE && BACKOFF_LIMIT_REACHED)/wpan_csma_notify_parent (WPANC_TX_FAILURE)
idle
backoff
t_type() == OPC_I
0 / 0
T_RECOVER)
(NODE_FAILURE || PAN_LOST)/wpan_csma_pkt_drop ()
0 / 0
(INVOKED)
(NODE_FAILURE || PAN_LOST)/wpan_csma_pkt_drop ()
(!CH_FREE)/wpan_csma_schedule_backoff ()
init
cca
(BACKOFF_TIMER_EXPIRED && CH_FREE)
(BACKOFF_TIMER_EXPIRED)
1 / 0
0 / 0
(CH_FREE)
transmit
1 / 0

Figure 4.4: Process model of CSMA/CA algorithm in OPNET

- To simplify MAC addressing, PAN ID is not used, rather the MAC addresses are itself are unique to the whole network.
- Collision is modeled in the following way: The first packet arriving at the receiver is locked, all the others are considered as noise, regardless of their signal power level. Therefore, a collision is recorded when a packet is dropped after thread_dra_ecc pipeline stage, and it was disturbed by other concurrent packets. It should be noted here that other packets are counted as noise and not as a collision.
- In the acknowledgment (ACK) mechanism, the ACK wait duration is defined as the time duration for which the MAC will wait to receive ACK for a given transmission. If the ACK is not received during this duration, the MAC will re-transmit. The ACK wait duration is set as 0.05 seconds and the number of re-transmissions is set to 5.

Statistics collected at this layer are defined as follows:

1. Throughput (bits/sec): Total data traffic in bits/sec successfully received and forwarded to the higher layer by the 802.15.4 MAC.
2. Data received (packets and bits): Number(or size) of data traffic in packets (in bits) successfully received by the MAC from the physical layer.
3. Packet Size (bits): Size of transmitted packets.
4. Total Collisions (packets): Total number of collisions at the PHY receiver.

   Note: a collision is reported if a packet takes the lock and it is discarded due to other incoming packets. It does not count colliding packets not taking the receiver's lock.
5. Load (bits/sec): Load (in bits/sec) submitted to the 802.15.4 MAC by its higher layers in this node.
6. Delay (sec): Represents the end to end delay of all the packets received by the 802.15.4 MAC of this WPAN node and forwarded to the higher layer.
7. Queuing Delay (sec): Represents the queuing delay that packets from the network layer incurs at the MAC.
8. Queue Size (packets): Total number of packets outstanding in the MAC queue.
9. Total Channel Access Failures (packets): Sum of data packets which did not get the chance of accessing the medium, because maximum number of back-offs was reached.
10. Data Traffic Sent (packets and bits): Number (or size) of data traffic in packets (in bits) transmitted.
11. Retransmission Attempts (packets): Number of retransmission attempts until either packet is successfully transmitted or it is discarded as a result of reaching retry limit.

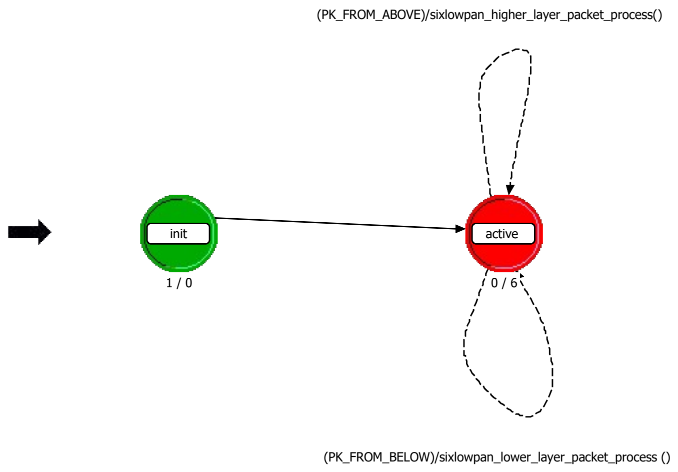

(PK_FROM_ABOVE)/sixlowpan_higher_layer_packet_process()
init
1 / 0
active
0 / 6
(PK_FROM_BELOW)/sixlowpan_lower_layer_packet_process ()

Figure 4.5: Process model for 6LowPAN in OPNET

12. Data Dropped (Retry Threshold Exceeded) (bits/sec): Higher layer data traffic (in bits/sec) dropped by the 802.15.4 MAC due to consistently failing retransmissions. This statistic reports the number of the higher layer packets that are dropped because the MAC couldn't receive any ACKs for the (re)transmissions of those packets or their fragments, and the packets' retry counts reached the MAC's retry limit.

13. Media Access Delay (sec): The total of queuing and contention delays of the data frames transmitted by the 802.15.4 MAC. For each frame, this delay is calculated as the duration from the time when it is inserted into the transmission queue, which is arrival time for higher layer data packets and creation time for all other frames types, until the time when the frame is sent to the physical layer for the first time.

4.3. 6LOWPAN ADAPTATION LAYER

6LoWPAN Layer is implemented according to IETF RFC 4944 [7] and RFC 6282 [8]. All Thread devices implement RFC 4944 and RFC 6282 [8] with modifications in addressing mode as described in Thread specification [11]. The Thread devices do not map IPv6 multicast addresses to IEEE 802.15.4 16-bit multicast addresses [11].Fragmentation is not implemented in the model. Figure 4.5 shows process model of 6LoWPAN adaptation layer in OPNET Modeler 14.5.

The layer consists of two main processes which implements required functions of 6LoWPAN. The processes are as follows:

- 6LoWPAN higher layer packet process: This process is triggered when ever a packet is received from the higher (IP layer). It implements IPHC compression function and collect data received in packets and bit/sec.

- 6LoWPAN lower layer packet process: This process is triggered when ever a packet

Table 4.2: Physical Layer parameters

6LoWPAN Attributes	Values
Fragmentation	disabled
Compression mode	IPHC Stateful
Address compression mode	CM3

is sent out of 6LoWPAN layer to lower MAC layer. It implements IPHC decompression function and collect data sent out of the layer in packets and bits/sec.

Table 4.2 summarizes compression mode, address compression mode and fragmentation chosen for 6LoWPAN Thread implementation.

Following statistics collected in this layer are defined as follows:

1. Data Traffic Sent (bits): Size of the compressed packets.
2. Data Traffic Received (bits): Size of the 6loWPAN packets received

4.4. NETWORK IP LAYER

Figure 4.6 shows the process model implemented for network IP layer. All the processes are categorized mainly into four, as follows:

1. MPL protocol in Thread
2. Network Formation in Thread
3. Router upgrade and downgrade
4. Neighbor detection

4.4.1. MPL PROTOCOL IN THREAD

As discussed in Chapter Chapter 3, the MPL works on Trickle algorithm to periodically transmit multicast messages. Thread MPL protocol implemented according to Thread specification in OPNET Modeler 14.5.

Thread uses only proactive forwarding of MPL and reactive forwarding is disabled, hence, Trickle algorithm [24] is used for MPL message transmissions. Thread MPL considers all the realm-local addresses as MPL domain address. This facilitates Thread devices to forward multicast messages to any arbitrary address within realm-local scope without IP-to-IP encapsulation. MPL parameters are defined by Thread MPL protocol and are shown in the table Table 4.3. They are passed to the network layer in the form of network layer attributes, as implemented in OPNET Modeler 14.5.

Trickle [24] is a very simple algorithm used for multicast propagation. It works on "inconsistency" in the network and is mainly used in density aware networks. Trickle works well both for sparse and dense networks. Figure 4.7 shows process model of Trickle implementation in OPNET Modeler 14.5.

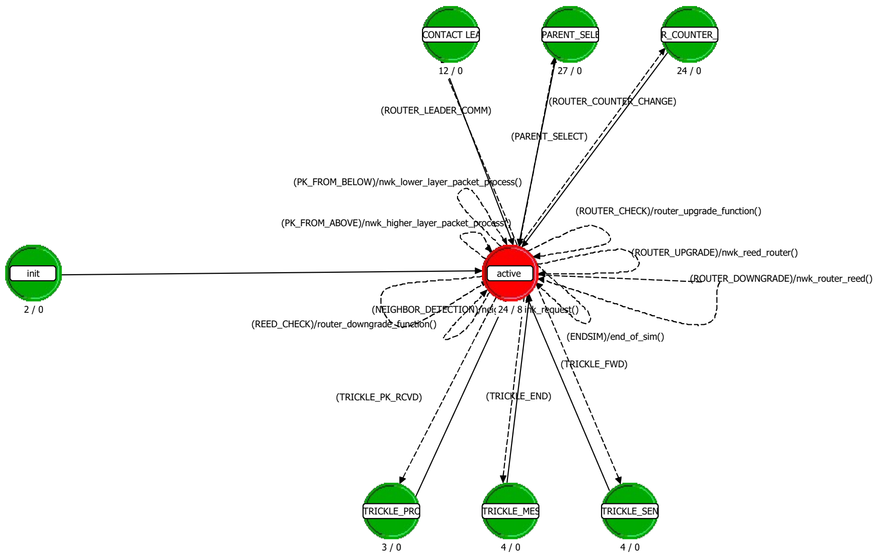


Figure 4.6: Process model of network layer in OPNET

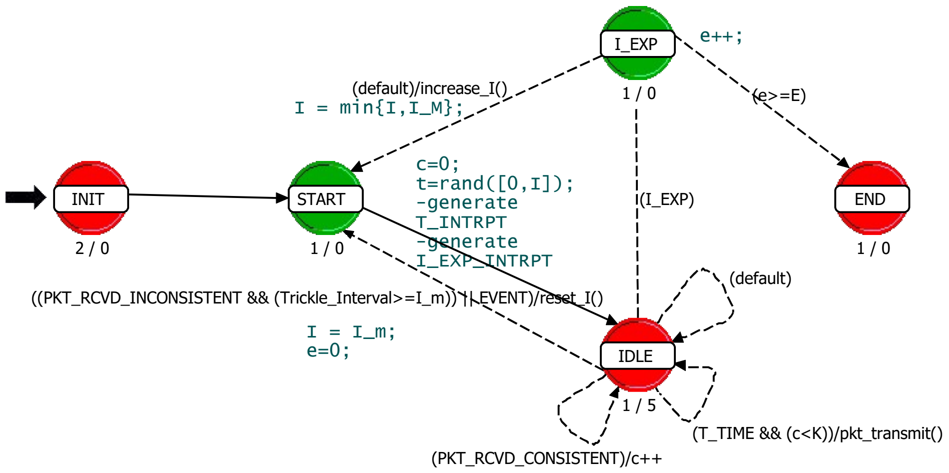


Figure 4.7: Trickle algorithm process model in OPNET 14.5

Table 4.3: MPL parameters Table

MPL Parameters	Assigned Values
DATA_MESSAGE_IMIN	64ms
DATA_MESSAGE_IMAX	64ms
DATA_MESSAGE_K	9999999999
DATA_MESSAGE_TIMER_EXPIRATIONS	2 for Routers and 0 for REEDs

Trickle algorithm starts with a time interval, I, that lies between the minimum time interval (DATA_MESSAGE_IMIN) and maximum time interval (DATA_MESSAGE_IMAX) as defined by Thread MPL protocol. Trickle maintains a parameter c, a counter, to monitor the number of times each MPL packet is forwarded.

A seed node generates a new MPL message in the form of an IPv6 packet and transmits it to the MPL domain address at uniform intervals of time. Each of these MPL messages have a sequence number. The "inconsistency" in the network occurs when there is a change in the sequence number of the received MPL message at a router. Typically, when ever a router receives next consecutive sequence numbered MPL message (when compared to the sequence number it already has), the router goes into inconsistent state and initiates trickle process. Routers receives the MPL message and checks for the sequence number.

If router receives MPL message with the sequence number of current trickle process, the message is discarded. If MPL message possessing next consecutive sequence number is received, new trickle process is started terminating existing trickle process. The parameters DATA_MESSAGE_IMIN, DATA_MESSAGE_IMAX are reset to default value and counter c is reset to 0.

The router stores new MPL message received. MPL message is forwarded only if c is less than redundancy constant K (which is defined by the MPL protocol) at a random transmission time t in the range (0,I]. After each forward, c is incremented. Additionally, the router interval remains same for the next iteration in the trickle process unlike in standard MPL protocol where, the interval is doubled in every iteration, till it reaches Imax. Hence, in Thread MPL protocol,

$$I_{min} = I_{max}$$

Furthermore, the iteration continues and router forwards each MPL message till it reaches a limit called DATA_MESSAGE_TIMER_EXPIRATIONS. In case of Thread MPL protocol, this is 2 for routers and 0 for non-routers (REEDs) as shown in the Table 4.3. Hence, it is expected in Thread that all routers forward each MPL packet exactly 2 times. Furthermore, DATA_MESSAGE_K is set to very high value as it should be infinity according to the specifications.

After the expiration time of current trickle process, a router waits for next sequence numbered MPL message. This process is continued throughout the simulation time.

4.4.2. NETWORK FORMATION IN THREAD

In this section, implementation processes for Thread network formation are discussed. Some processes are simplified from Thread specification.

A simple Thread network is the Border Router itself. It connects to external networks and Internet. Before starting the network formation process, the following assumptions have to kept in mind:

- A node is already commissioned into the network. It becomes REED after attaching to a parent.
- All nodes are router capable since the nodes are Luminaries, except seed nodes. Seed nodes are not router capable. They used in MPL scenarios for generating MPL multicast messages.
- Processes for Minimal Thread Device (MTD) are not implemented.
- There is single Border Router in the network.
- A node is connected to the network if it possess either a Router ID or a Child ID along with finite routing cost from Leader. Otherwise, the node is assumed as unconnected with routing cost to Leader as infinity.
- REED to Leader routing cost is 10. Router to Leader routing cost is 1.

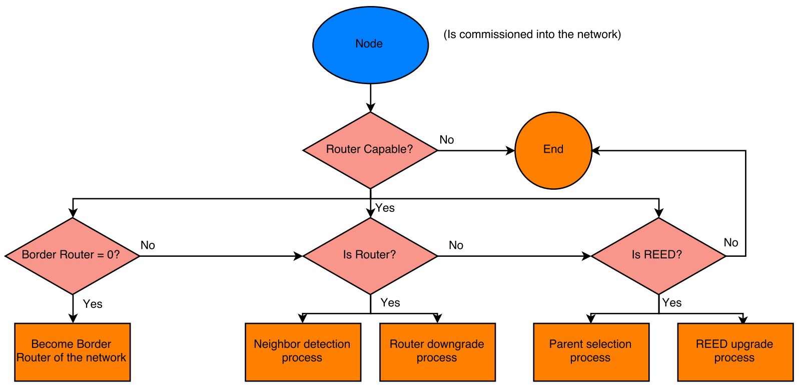


Figure 4.8: Processes associated with Network formation

Figure 4.8 shows the processes involved during network formation.The first process in the implementation starts with assignment of Border Router. In Chapter 3, different network components and roles were discussed. During network formation, each node transits into different roles. Various conditions are checked and decisions are made accordingly. They are summarized as follows: For a router capable node,

- If there is no Border Router in the network, node assigns itself as a Border Router.

- If there is already a Border Router in network, node assumes REED role and initiates parent selection process and REED - Router upgrade process.
- If the node becomes a Router through upgrade process, it initiates Neighbor detection process and Router downgrade process.

Border Router Implementation

When node becomes a Border Router, it assumes Router's, Border Router's and Leader's roles.Figure 4.9 depicts the processes associated with Border Router. It can be seen that Border Router has the processes which are implemented for Router except the upgrade and downgrade process.

As part of Leader's role, it handles router ID assignment to the nodes in network when they upgrade to Router. Additionally, Border Router also handles router ID release when the nodes downgrade, and reuses the ID. The size of Router ID list is 62 (almost twice of the maximum number of possible routers in the network) as per the specification. One

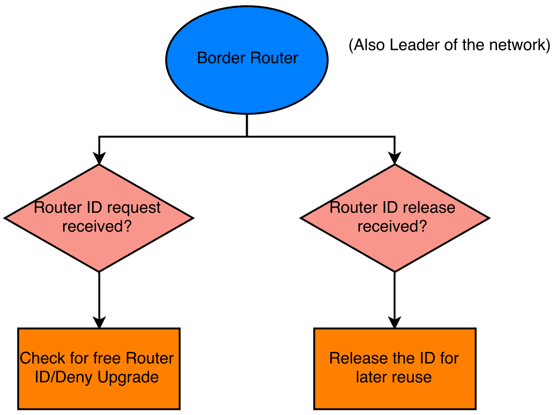

Border Router
(Also Leader of the network)
Router ID request received?
Router ID release received?
Check for free Router ID/Deny Upgrade
Release the ID for later reuse

Figure 4.9: Functions associated with Border Router

of the design decisions made was not to implement Router ID assignment section in Page 5-29, mentioned in Thread specification 1.1.1 [11]. Router ID assignment section defines Address Solicit Request, Address Solicit Response and Address Release Notification, which requires CoAP POST formatted with different configurations. However, CoAP protocol implementation was out of scope to this project. Hence, this communication was simulated with a delay of 90ms. That is, when ever a Router and Leader communicate for Router ID assignment or release, a delay of 90 milliseconds is inserted in network communication.

Router Implementation

A connected node is REED in the network. A node becomes Router when ever REED upgrades to Router role. Figure 4.10 depicts different processes associated with node as a Router.

The processes that are continuously checked by the router, through interrupts, are as follows:

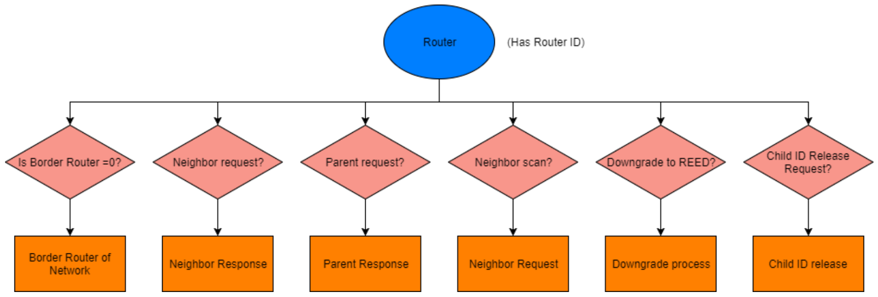


Figure 4.10: Processes associated with Router

1. A router periodically checks if the Border Router has failed in the network (number of Border Router in the network becomes 0). If there is no Border Router in the network, a router assigns itself as Border Router and broadcasts that information to all the nodes int he network.

2. Neighbor Request message:If a router receives Neighbor Request message from neighboring routers, it triggers the Neighbor Response process to respond.All the Neighbor detection processes are discussed in Neighbor detection section elaborately.

3. Parent request message: If a Router receives Parent Request message from a node, it triggers parent response process to respond provided the number of children is < MAX_CHILD_CAPACITY. MAX_CHILD_CAPACITY for Border Router is 64 and 10 for other routers.

4. Periodic neighbor scan interrupt will trigger Neighbor Request message to all the neighboring (who lie within the radio range) Routers.

5. When ever there is Router number change in the network (It can be due to upgrade or downgrade), the router triggers downgrade process, if it satisfies the following condition:

   Number of routers in the network > ROUTER_DOWNGRADE_THRESHOLD

   where, ROUTER_DOWNGRADE_THRESHOLD = 23

6. If a router receives Child ID release request message, then it triggers Child ID release process

REED IMPLEMENTATION

Unconnected node becomes REED when it attaches itself to most suitable (based on 2-way Link quality and Link cost) neighboring router in the network. The parent router assigns Child ID to the unconnected node, thereby adding it the network as REED.

Figure 4.11 shows different processes associated with REED. They are summarized as follows:

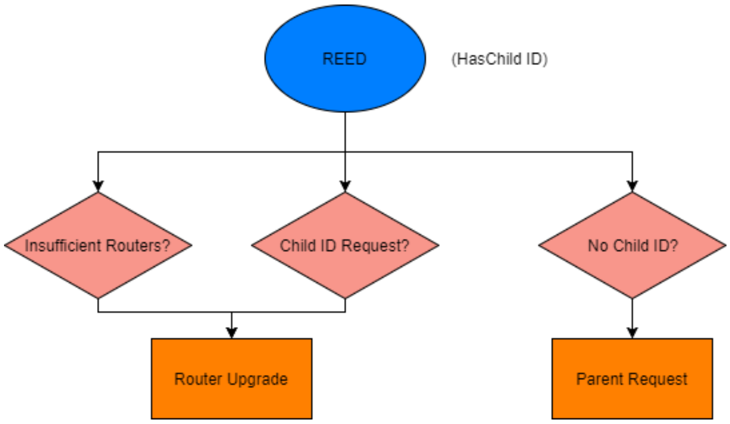


Figure 4.11: Processes associated with REED

- If the number of routers in the network is < 16 (ROUTER_UPGRADE_THRESHOLD) , the REED triggers slow REED to Router upgrade process. Otherwise, fast upgrade process is triggered.
- If Child ID request is received, REED to Router upgrade process is triggered immediately.
- If REED do not have Child ID (ID expiry or parent router failure), parent request message is triggered which is explained in parent selection process.

PARENT SELECTION PROCESS

Parent attaching process of unconnected node (routing cost to Leader is infinity) in the network comprises of four message exchanges as depicted in call flow diagram Figure 4.12. MLE parent request link-local multicast message is transmiited to all the nodes(Routers and REEDs)within its radio range by the unconnected node. This message is sent every t seconds, given by the following value,

$$t(sec) = rand[0.9, 1.1] \\ + \text{MLE\_MULTICAST\_RETRANSMISSION\_DELAY}$$

where, MLE_MULTICAST_RETRANSMISSION_DELAY = 5 seconds

The packet format used for Parent Request message is discussed in Chapter 3.

The message is transmitted to both Routers and REEDs in the network located within 1 hop by setting R and E values to 1 in Scan Mask TLV. The sender node waits for 1.25 seconds (MLE_PARENT_REQ_SCANMASK_RE_TIMEOUT) to receive parent response message from Router or REED.

The sender node waits for a time t seconds to receive parent response, before restart-

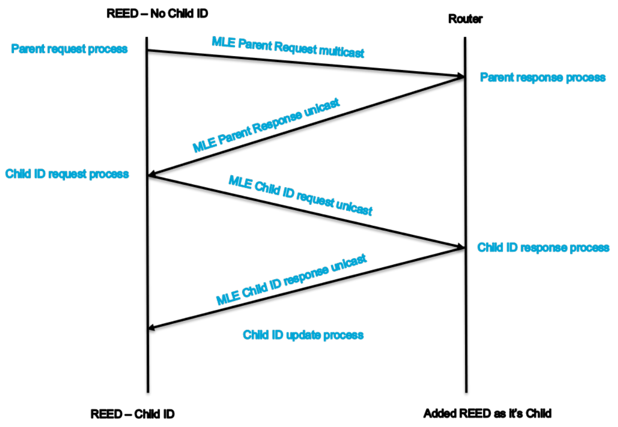


Figure 4.12: Call flow of Parent Selection process

ing parent selection process again. Here,

$$\begin{aligned} t(sec) = & \, rand[0.9, 1.1] \\ & + \text{MLE\_MULTICAST\_RETRANSMISSION\_DELAY} \\ & - \text{PARENT\_REQUEST\_WAIT} \end{aligned}$$

where, PARENT_REQUEST_WAIT = 2 seconds

All the Routers and REED receiving the multicast message, respond to sender node with MLE Parent Response unicast message along with Link Quality information of the incoming multicast message. This message is transmitted after a random time selected in the range [0,MLE_PARENT_RSP_ROUTER_JITTER],

where, MLE_PARENT_RSP_ROUTER_JITTER = 1 second.

Once the sender node receive all the parent response messages, it calculates 2-way Link Quality and Link Cost for each response (refer Table 3.1). All the potential parent list is stored along with their respective Link Cost. A parent with least Link Cost is chosen and MLE Child ID Request unicast message is sent to that parent node.

If the chosen parent node is a Router (REED), it behaves as described in Router (REED) implementation and sends MLE Child ID Response unicast message if all conditions are satisfied.

Once the sender node receives Child ID from the parent, it assigns itself Child ID and Parent ID. The sender now assumes REED role. All communication to/from it happen via its parent (router) node.

REED TO ROUTER UPGRADE PROCESS

Figure 4.13 shows processes involved in REED to Router upgrade process. When ever there is a change in number of routers in the network, all the nodes in the network are notified. At REED, the following process happen:

- If number of routers in the network is < 16 (ROUTER_UPGRADE_THRESHOLD), then, it waits for a random time in the range [0, ROUTER_SELECTION_JITTER], where, ROUTER_SELECTION_JITTER = 120 seconds. It checks again for the same condition, before sending Router ID request to Leader. This process is called slow upgrade. A slow upgrade happens while network is building up Routers.

- If number of routers in the network lies within 16 < Router number < MAX_ROUTERS (32),

  then Router ID request is sent to Leader.
- As mentioned in Chapter 3, it is assumed in this implementation that it takes 90ms for the Router to receive Router ID from Leader (if Router IDs are available). The whole process is terminated if Router ID is not available.
- If the Router ID is received, REED stores the Router ID, and REED's details are added to global Router ID list. REED transmits Child ID release request unicast message to its parent router and starts functioning as a Router.
- Number of routers in the network increases and is notified to all nodes.

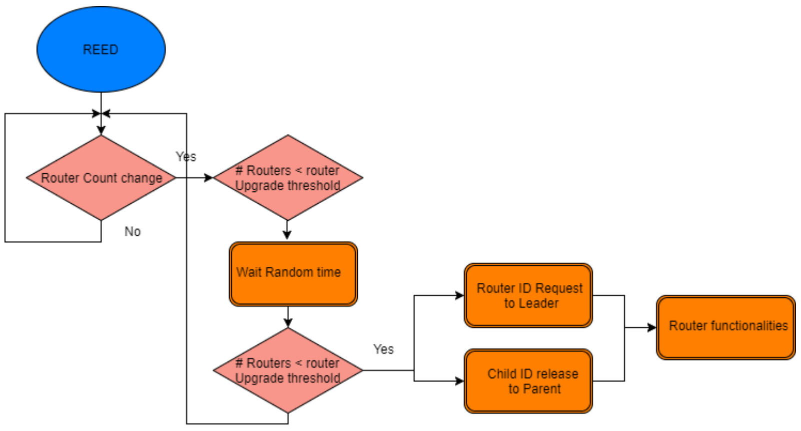

REED
Router Count change
Yes
No
# Routers < router Upgrade threshold
Wait Random time
# Routers < router Upgrade threshold
Yes
Router ID Request to Leader
Child ID release to Parent
Router functionalities

Figure 4.13: Processes associated with REED to Router upgrade

ROUTER TO REED DOWNGRADE PROCESS

Figure 4.13 shows processes involved in Router to REED downgrade process.

When a Router is notified with change in the number of routers in the network, it checks the following conditions, as described in Thread specification:

1. It is not the Border Router of the network
2. Number of Routers in the network > 23 (ROUTER_DOWNGRADE_THRESHOLD)
3. Number of neighbors having Link Quality >2 should be greater than 7 (MIN_DOWNGRADE_NEIGHBORS)

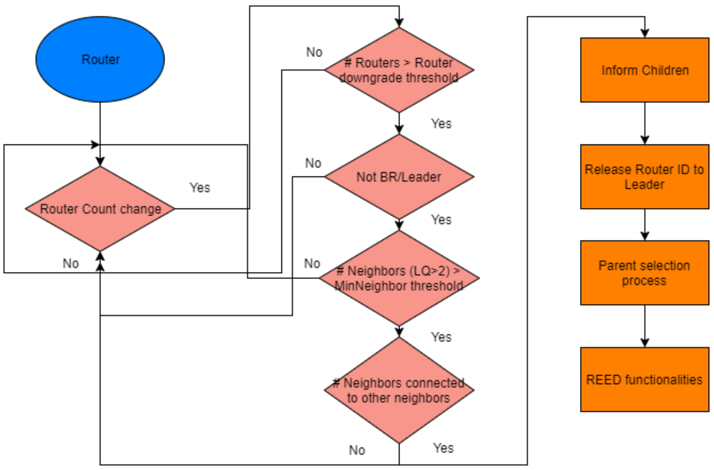

Router
Router Count change
Yes
No
# Routers > Router downgrade threshold
No
Yes
Not BR/Leader
No
Yes
# Neighbors (LQ>2) > MinNeighbor threshold
No
Yes
# Neighbors connected to other neighbors
No
Yes
Inform Children
Release Router ID to Leader
Parent selection process
REED functionalities

Figure 4.14: Processes associated with Router to REED downgrade

4. Additionally, Number of Children (REEDs attached to it) < 3× (Number of Routers in network - ROUTER_DOWNGRADE_THRESHOLD)

If all the above conditions are satisfied, the router transmits Router ID release message to Leader (assumed 90 ms). The router clears its neighbor list, entries in Child ID list is cleared, generating interrupt to all its Child REEDs to start parent selection process. Finally, the router temporarily becomes unconnected till it attaches to another router through parent selection process. Number of routers in the network decreases and is notified to all nodes.

NEIGHBOR DETECTION PROCESS

Neighbor detection process is triggered every 10 seconds in each router, once the router assumes all its functionalities. The call flow in the process is shown in Figure 4.15.

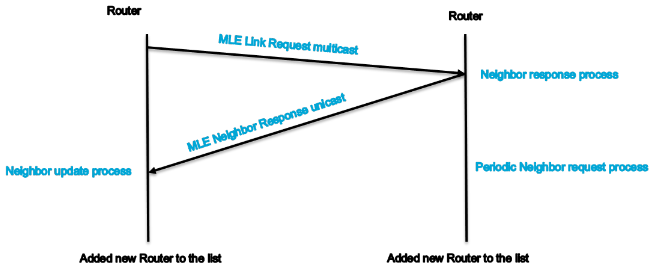

Router
Router
MLE Link Request multicast
Neighbor response process
MLE Neighbor Response unicast
Neighbor update process
Periodic Neighbor request process
Added new Router to the list
Added new Router to the list

Figure 4.15: Call flow for neighbor detection process

Following summarizes the process:

- MLE Link Request multicast (Link Local address) message is transmitted, which has its current neighbors' information. This is 1-hop message.

- Routers receiving this message process the incoming message Link Quality, stores the entry in their neighbor list, transmits MLE Link Response unicast message to the sender router along with the Link Quality information.

- At the sender router, MLE Link Response message is received along with the Link Quality of its own multicast message (Outgoing Link Quality). Additionally, response message Link Quality is also calculated (Incoming Link Quality). All the information about the neighbor router is stored in its neighbor list.

- This process is repeated periodically.

It is to be noted here that the Thread network partitioning process. Not implementing Thread network partitioning might affect network formation for large-scale networks.

The statistics collected in this layer are defined as follows:

1. MPL Data Traffic Received (bits or packets): MPL traffic received (in bits or packets) at a node.

2. MPL Data Traffic Sent (bits or packets): MPL traffic sent (in bits or in packets) from this node.

3. MPL Forwarding Delay (seconds): Time from the reception of an MPL data packet to the forwarding of it at a node.

4. Trickle Interval (seconds): Time interval of each trickle process.

5. Router count: Number of routers in the network.

6. Unconnected node count: Number of nodes unconnected in the network.

4.5. APPLICATION LAYER

Thread stack do not define any application layer. In this implementation, simple application layer is created for traffic creation and application statistics collection. They are called as application source and application sink.

Process model of application traffic source is as shown in Figure 4.16. It creates application traffic, by specifying the start time, end time, packet inter-arrival time and traffic destination. It is mainly implemented for MPL traffic generation. It is designed such that seed node in MPL scenarios will not be part of any multicast group. All the attributes can be configured for MPL traffic generation.

Process model of application traffic sink impleneted in OPNET Modeler 14.5 is shown in Figure 4.17. At each packet reception, the model records application statistics involving the receiver side of the communication. Then, it destroys the received packets. Statistics recorded at this layer are defined as follows:

1. Multicast end to end delay (seconds) : This is defined as time difference between the packet creation at application source of seed node and time of packet reception by application traffic sink at a node.

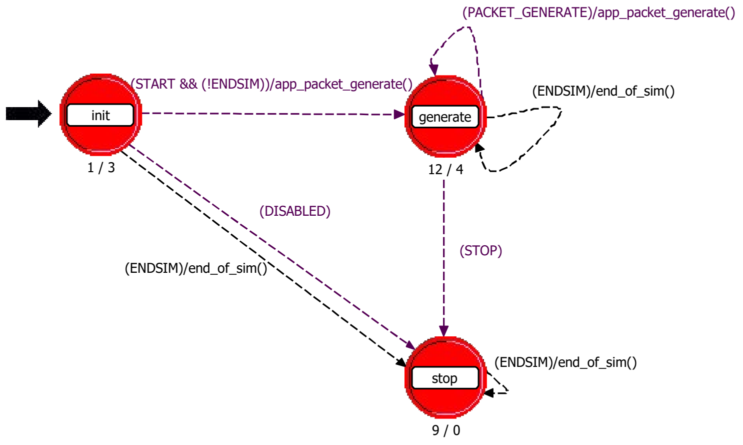


Figure 4.16: Application Traffic Source Model

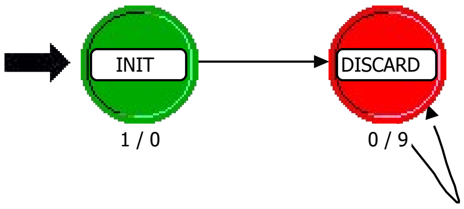


Figure 4.17: Application Traffic Sink Model

2. Multicast traffic received (packets and bits): Multicast traffic received in packets (in bits) by application traffic sink at a node.

3. Multicast traffic sent (packets and bits): Multicast traffic sent in packets (in bits) by application traffic source at seed node.

4. Multicast Packet Delivery Ratio: Ratio of packets successfully received through multicast at each node to the total number of multicast traffic generated by seed node.

5

Results and Performance Analysis

5.1. Introduction

This chapter summarizes the results of various scenarios that are obtained through simulation. The results are primarily divided into 2 separate sections. The scenarios under consideration contain 100 nodes, 256 nodes with single and multiple seeds for multicast forwarding.

5.2. Simulation setup

Each node in the scenario is placed 2.5m apart within a grid of equal rows and columns.

The physical layer parameter presented in Figure 5.1, have been chosen to satisfy the path loss equation given by Equation 4.2.

Table 5.1: Physical layer parameters used during simulation

Parameter	**Value**
Packet	-85dBm
Transmit power	0 dBm
Path loss exponent(γ)	3

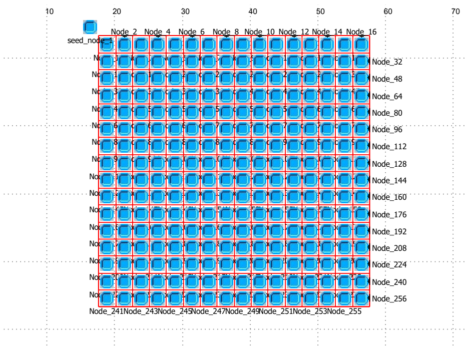
5.3. Data processing and results collection

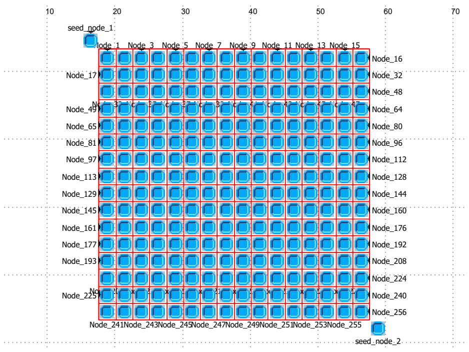
In Chapter 4, layer-wise statistics collected for simulation were defined. Thread model for different MPL and network formation scenarios was simulated. All the defined statistics were collected in OPNET Modeler 14.5 and exported to a text file. The data was processed in Spyder 3.1.4 from the Anaconda tool chain and was programmed in Python 3.6.0.

5.4. Thread MPL Results

5.4.1. Scenario Setup

Figure 5.1 depicts geometric grid configuration of MPL 256 node scenario for 1 seed, 2 seed and 4 seed node scenarios. Figure 5.2 depicts geometric grid configuration of MPL 100 node scenario for 1 seed, 2 seed and 4 seed node scenarios. Simplified Thread model in Matlab was used to determine routers in 100 node and 256 node scenarios. The model was simulated to determine the routers in the network, as router number and routers differed for each run. This was repeated for 10 runs for both 100 node and 256 node scenario. 10 router configurations for each (100 and 256) was stored to configure the OPNET model.


(a) With one seed node


(b) With two seed nodes

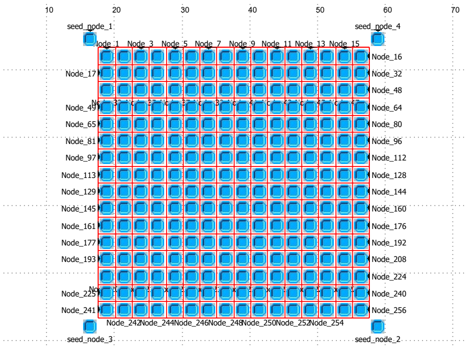
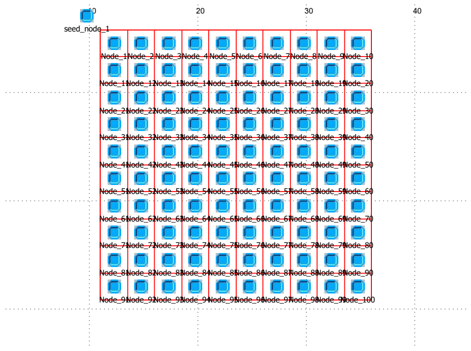

(c) With four seed nodes

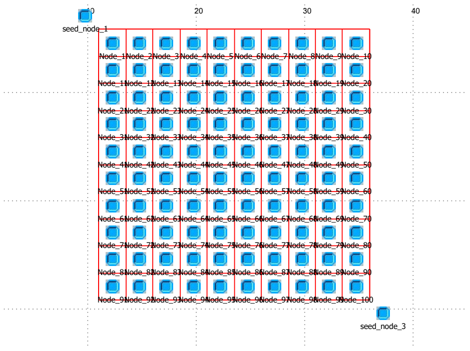
Figure 5.1: Geometric configuration of 256 nodes in a grid with 1, 2 and 4 node scenarios

10 different scenarios for 100 node was created, each with different router configuration. Each scenario was simulated for for 5005 simulation seconds.This was done for 1,2 and 4 seed scenarios, thereby having 30 different scenarios. Results were collected for all the runs.The same process was repeated for 256 node scenario as well.

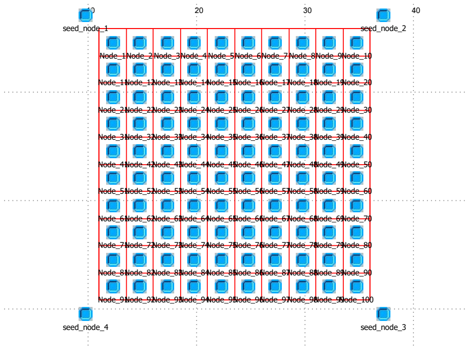
For result analysis of each parameter, values from all the router configurations was considered for each of 100 node 1 seed, 100 node 2 seed and 100 node 4 seed cases. The same procedure was repeated for 256 node single and multi-seed scenarios as well.



(a) With one seed node



(b) With two seed nodes



(c) With four seed nodes

Figure 5.2: Geometric configuration of 100 nodes in a grid with 1, 2 and 4 node scenarios

Application layer attributes configured for seed node are as shown in Figure 5.2

Table 5.2: Application layer parameters used during MPL simulation

Attributes	**Value**
Destination	Multicast_1
Packet inter-arrival time	constant(10)
Packet Size (bits)	constant(512)
Start time	constant(10)
Stop Time	Infinity

In case of a 100 node scenario, a grid of 10 × 10 nodes is used and in case of a 256 node scenario, a grid of 16 × 16 nodes is used.

5.4.2. ANALYSIS OF MULTICAST END-TO-END DELAY

One of the Multicast latency measurement parameter in the network is Multicast End-to-End delay. This section analyses Multicast End-to-End Delay that is collected at the Application layer sink of the Thread implementation model for both 100 node and 256 node single and multi- seed scenarios.

100 NODE SCENARIO

Figure 5.3 shows the variation of Multicast End-to-End Delay simulated for 100 nodes with a single seed node. The graphs shows an increase in Multicast End-to-End Delay

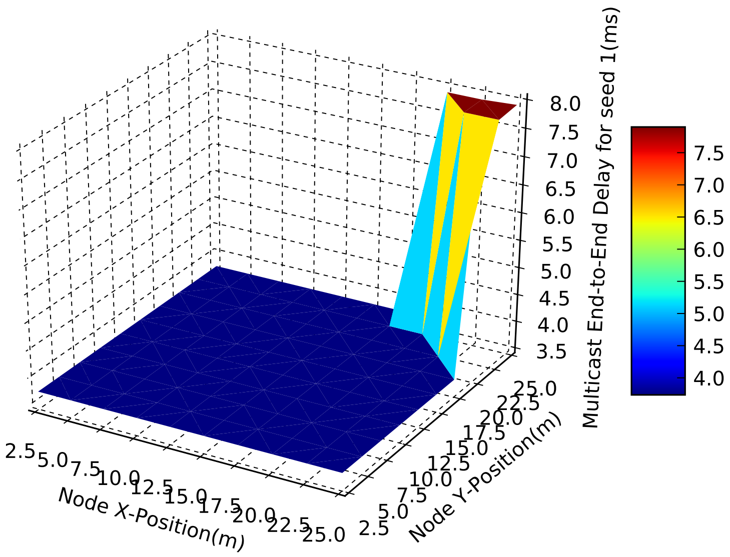


Figure 5.3: Multicast End-to-End Delay simulated with a single seed for 100 nodes

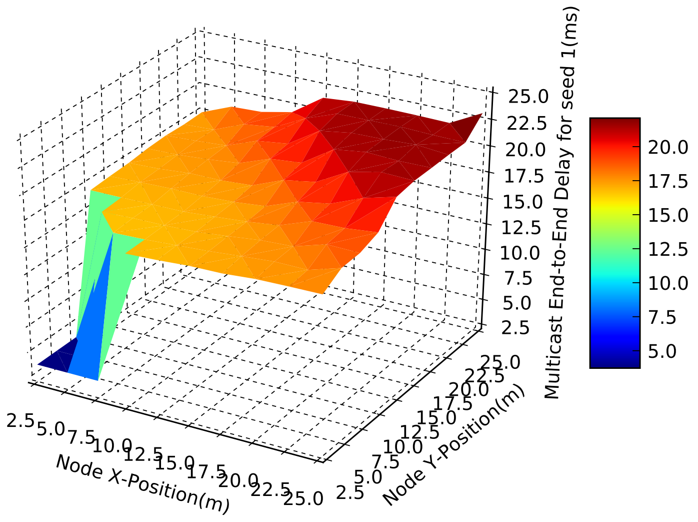
with an increase of relative distance from the seed node, placed at the origin of the $X-Y$ grid. The node placement in the grid is similar to the one show in Figure 5.2a. With the seed node placed at the origin, it is seen that the average Multicast End-to-End Delay remains $\approx 4ms$ to $4.5ms$. Whereas, close to the diagonal end from the seed node (refer Figure 5.2a), the average Multicast End-to-End Delay increases drastically. This is attributed to:

1. Distances between the seed node and nodes placed are relatively high.
2. Multiple hops(2) needed for packet to reach the destination, Multicast Forwarding Delay is added by each forwarding device.

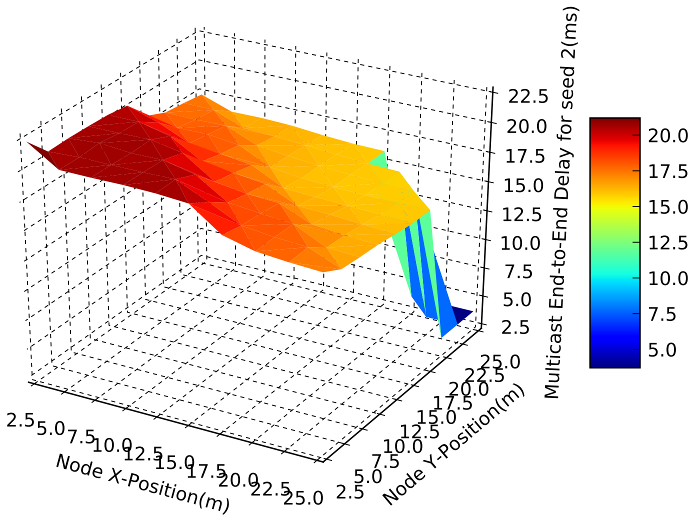
.

The increase in average Multicast End-to-End Delay almost doubles to $\approx 8ms$ when compared to nodes placed close by.

It should be noted that the average Multicast End-to-End Delay remains fairly constant over significant part of the node grid. This is justified by the fact that significant number of nodes are at a distance $< 30m$ relative to the seed node. From Figure 5.1, it is expected that these nodes communicate with a single hop.



(a) Relative to Seed 1

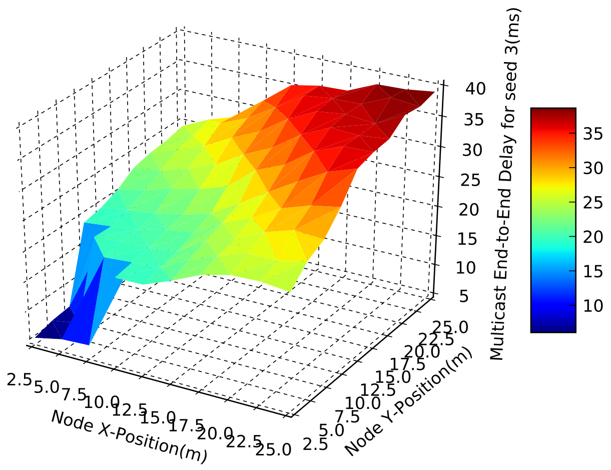


(b) Relative to Seed 2

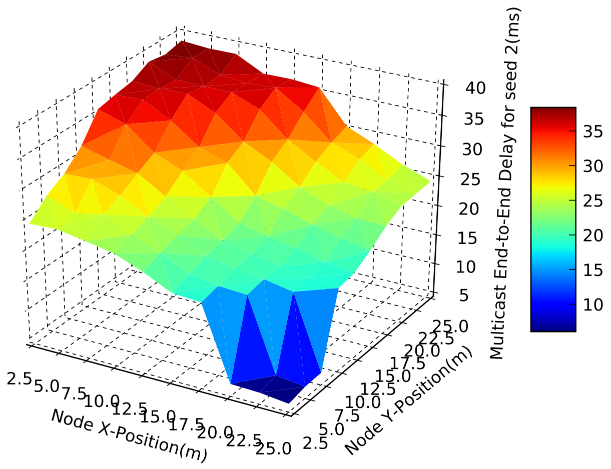
Figure 5.4: Multicast End-to-End Delay for a 100 node scenario with 2 seed nodes

Furthermore, Figure 5.4 shows the variation of average Multicast End-to-End Delay for a 100 node scenario with 2 seed nodes.

The corresponding grid, in Figure 5.2b, consists of 2 seed nodes place diagonally opposite each other to maintain maximum distance. As seen from Figure 5.4, the Multicast End-to-End Delay with respect to a single node increases drastically for significant part of the grid. The substantial increase in the parameter is expected due to collisions arising from two transmitting seeds. It is further observed that collisions reduce closer to a single seed node. The peak value of average Multicast End-to-End Delay is $< 23ms$.

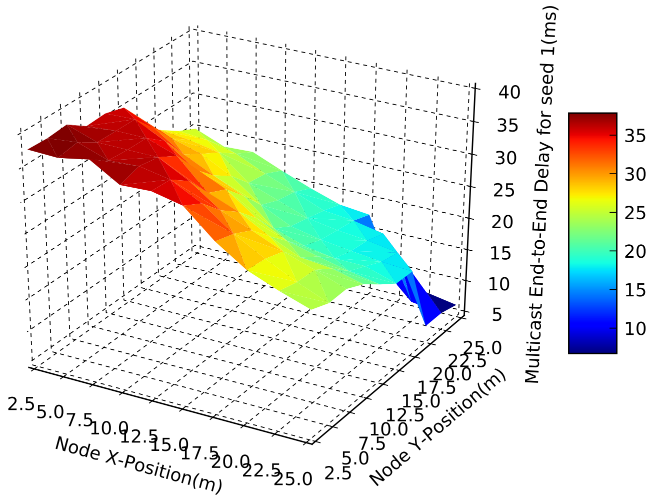


(a) Relative to Seed 1

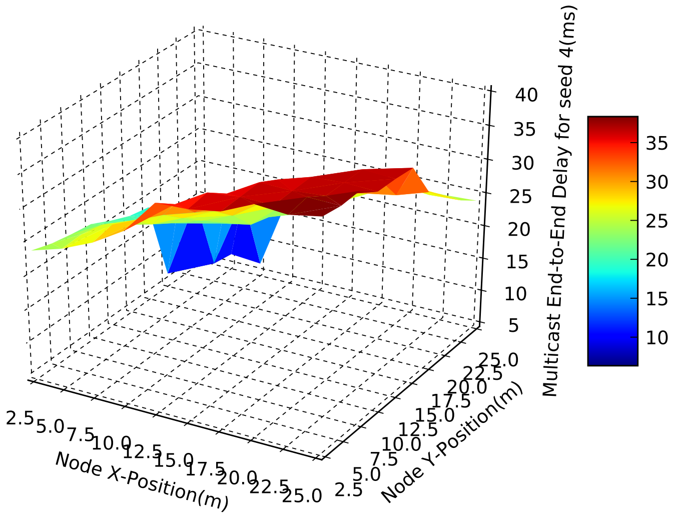


(b) Relative to Seed 2

A similar analysis is extended to a scenario with 4 seed nodes. Figure 5.5 shows the variation of average Multicast End-to-End Delay with 4 seed nodes in the grid, depicted by Figure 5.2c.



(c) Relative to Seed 3



(d) Relative to Seed 4

Figure 5.5: Multicast End-to-End Delay for a 100 node scenario with 4 seed nodes

The increase in average Multicast End-to-End Delay, relative to a particular seed node is proportional to the relative distance between the seed node and the node placed in the grid. The peak value of this parameter is $> 40ms$ owing to the number of collisions from 4 transmitting seed nodes. Figure 5.6 gives an overview of Multicast End-to-End Delay values for various seed configuration, which is tabulated as shown in Table 5.3.

Table 5.3: Overview of simulated values for Multicast End-to-End Delay for 100 nodes

Seeds	**Maximum** (ms)	**Minimum** (ms)	**Average** (ms)
1	5.12	2.92	3.98
2	116.66	3.25	21.89
4	196.48	24.56	48.04

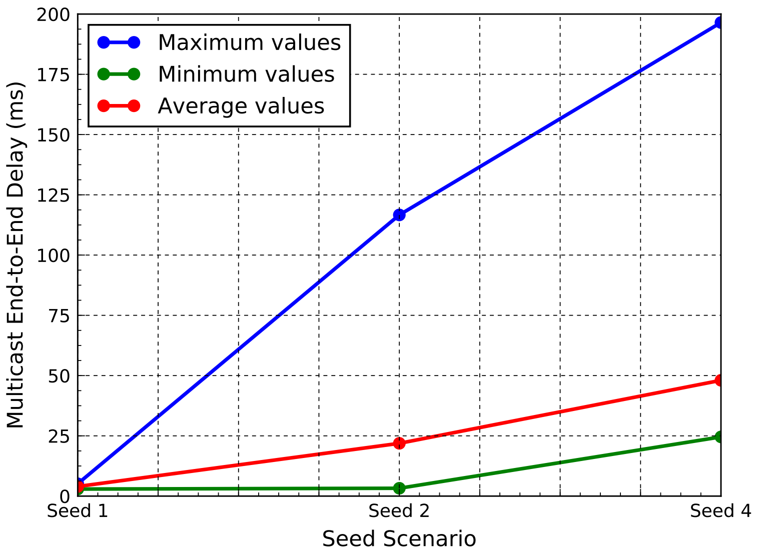


Figure 5.6: Summarized Multicast End-to-End delay for all the seed scenarios

256 NODE SCENARIO

Figure 5.7 shows the variation of average Multicast End-to-End Delay simulated for 256 nodes with a single seed node. The 256 node grid placement of nodes corresponds to Figure 5.1a, wherein the seed node is place close to *Node_1*. Owing to its proximity with the seed node, *Node_1* experiences the least Multicast End-to-End Delay on average as seen from Figure 5.7.

Similar to the 100 node analysis, the Multicast End-to-End Delay increase when the relative distance from the seed node increases beyond a euclidean distance of $\approx 30m$. Beyond this range, multi-hop transmission is adopted that increases Multicast End-to-End Delay. The peak delay with a single seed is $\approx 14ms$, and is observed to at a distance farthest away from the seed node.

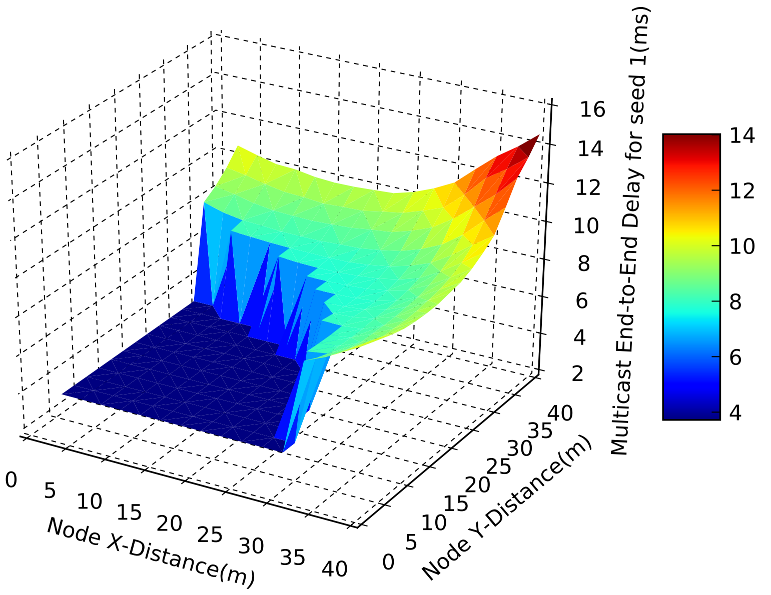
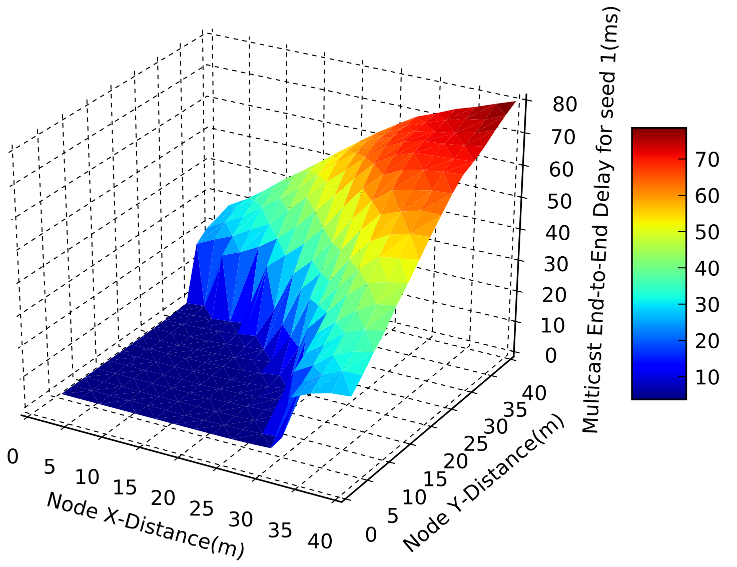
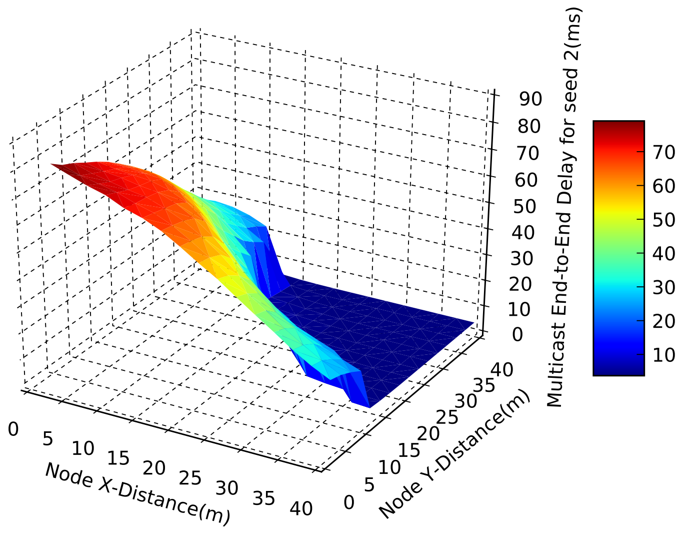


Figure 5.7: Multicast End-to-End Delay simulated with a single seed for 256 nodes



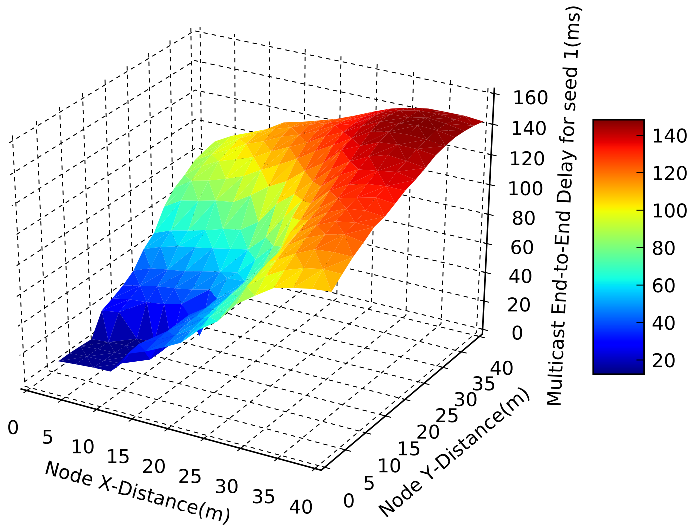
(a) Relative to Seed 1



(b) Relative to Seed 2

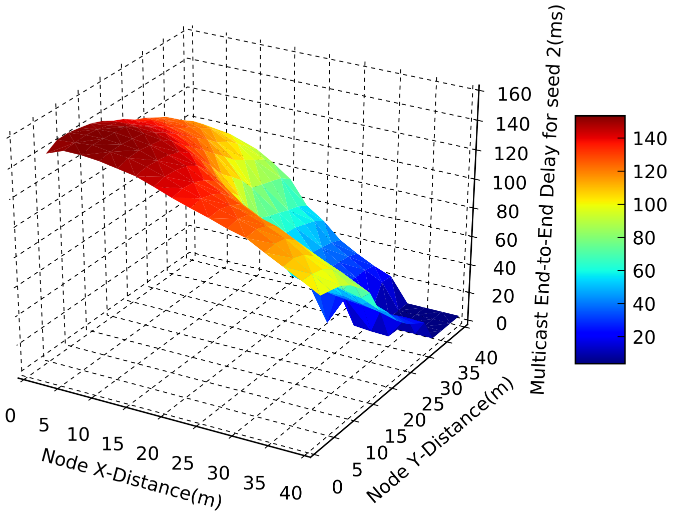
Figure 5.8: Average Multicast End-to-End Delay simulated for 256 nodes with 2 seed nodes

Figure 5.8 shows the variation of average Multicast End-to-End Delay in a scenario with 2 seed nodes, which corresponds to the grid arrangement in Figure 5.1b.

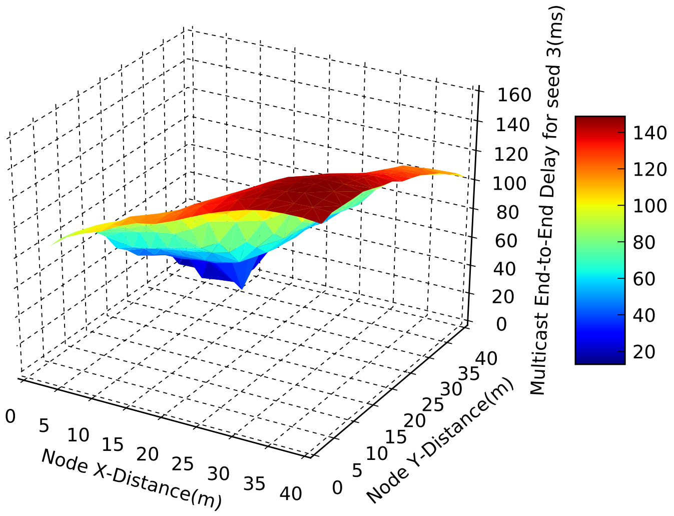


(a) Relative to Seed 1

Similar to the 100 node scenario, there is an increase in average Multicast End-to-End Delay owing to collisions. However, in contrast to the 100 nodes case, the increase only begins when the euclidean distance $> 30m$. The peak average of Multicast End-to-End Delay in this case $\approx 80ms$, and is observed at as distance farthest from the seed node.

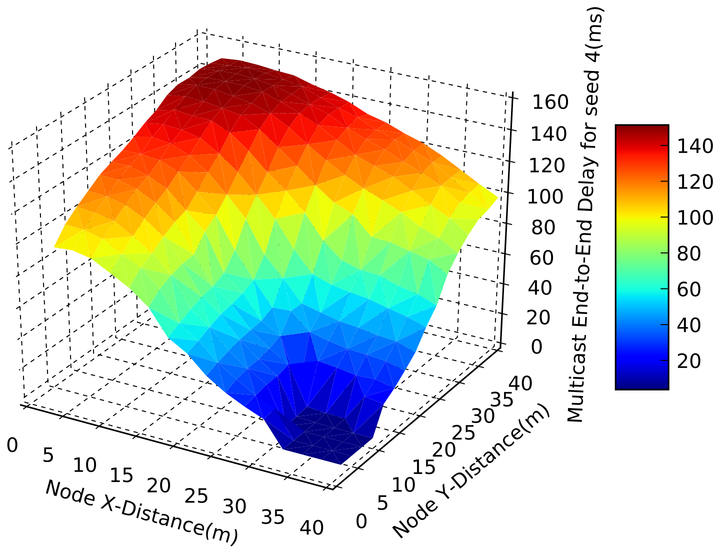


(b) Relative to Seed 2



(c) Relative to Seed 3

The variation of average Multicast End-to-End Delay for 256 nodes with 4 seed nodes is shown in Figure 5.9.



(d) Relative to Seed 4

Figure 5.9: Average Multicast End-to-End Delay for 256 nodes with 4 seeds

The variation is quite expected owing to increase in collision due to 4 transmitting seed nodes. Following the 100 node scenario, the Multicast End-to-End Delay with respect a particular seed node and a node in the grid have the least relative euclidean distance. The peak average Multicast End-to-End Delay in this case is $\approx 128ms$.

Table 5.4 and Figure 5.10 summaries the values of Multicast End-to-End Delay for various seed configurations.

Table 5.4: Overview of simulated values for Multicast End-to-End Delay for 256 nodes

Seeds	**Maximum** (ms)	**Minimum** (ms)	**Average** (ms)
1	31.37	4.41	6.87
2	120.65	4.47	37.12
4	307.62	87.81	127.83

5.4.3. Analysis of Packet Delivery Ratio

Packet loss in a network is analyzed through Packet Delivery Ratio (PDR) statistic. This section analyses the PDR collected at the Application layer of Thread model, for both 100 and 256 node single and multi-seed scenarios.

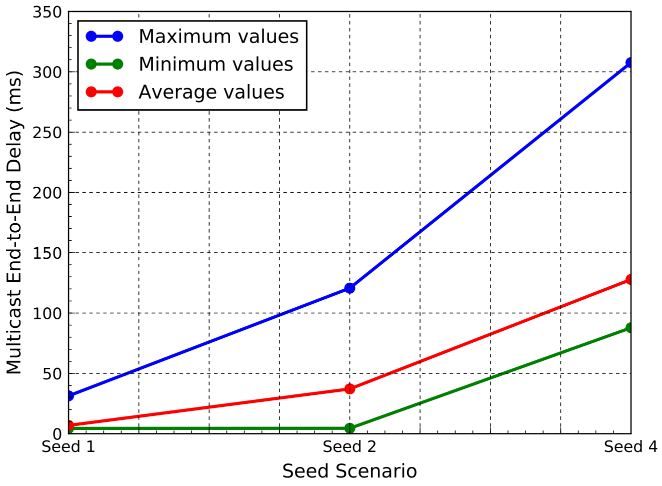


Figure 5.10: Summarized Multicast End-to-End delay for all the seed scenarios

100 NODE SCENARIO

Figure 5.11 shows the average PDR for 100 node configuration with a single seed. It is observed that PDR is 100%, i.e., no packet loss is observed in this scenario. This is expected since, a MPL packet is transmitted every 10 seconds, thereby, allowing an ample amount of time for the packet to reach all the nodes with least collisions. Furthermore, the average number of routers in the network is < 15. Therefore, only 15 nodes are capable of forwarding MPL packets, reducing the data traffic in the network.

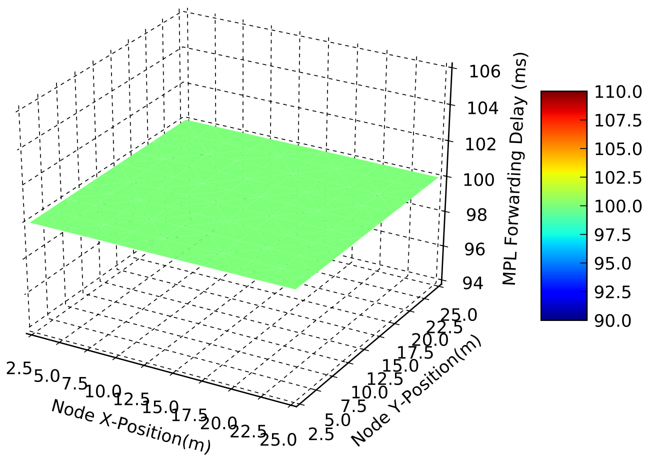


Figure 5.11: Average Packet Delivery Ratio for 100 nodes with 1 seed

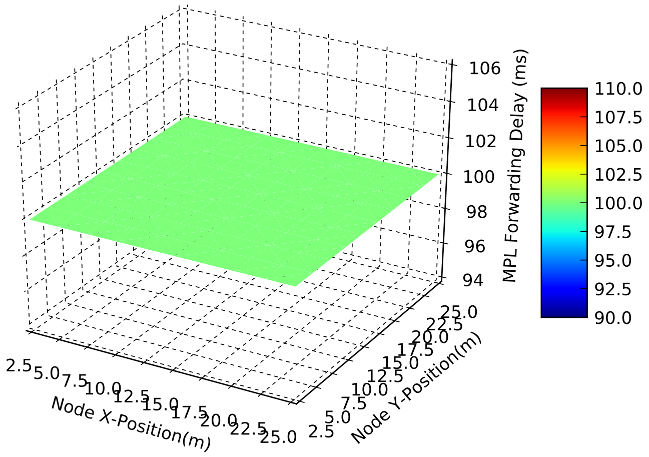


Figure 5.12: Average Packet Delivery Ratio for 100 nodes with 2 seed

Figure 5.12 shows the average PDR for 100 node configuration with a 2 seeds. It is observed that PDR is 100%, i.e., no packet loss is observed in this scenario. The reasons again can be attributed to that stated to single seed scenario. Additionally, even if a slight drop in PDR is expected at diagonal corners of each seed node, PDR remains 100% due relatively lower distance between the seed node and farthest diagonal node.

256 NODE SCENARIO

Figure 5.13 shows the average PDR for 256 node configuration with a single seed. It is observed that PDR is 256%, i.e., no packet loss is observed in this scenario.

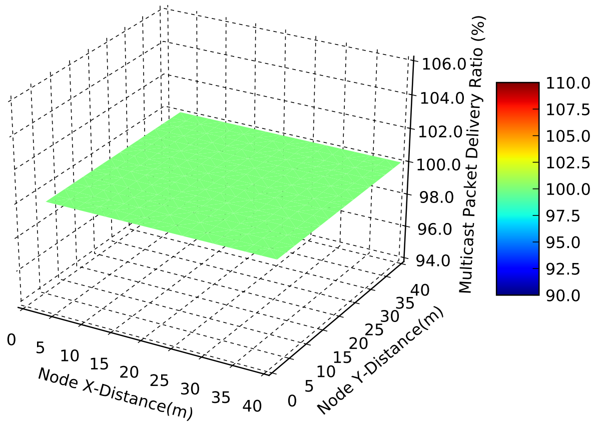


Figure 5.13: Average Packet Delivery Ratio for 256 nodes with 1 seed

This is expected since, a MPL packet is transmitted every 10 seconds, thereby, allowing an ample amount of time for the packet to reach all the nodes with least collisions. Furthermore, the average number of routers in the network is < 23. Therefore, only 23 nodes are capable of forwarding MPL packets, reducing the data traffic in the network.

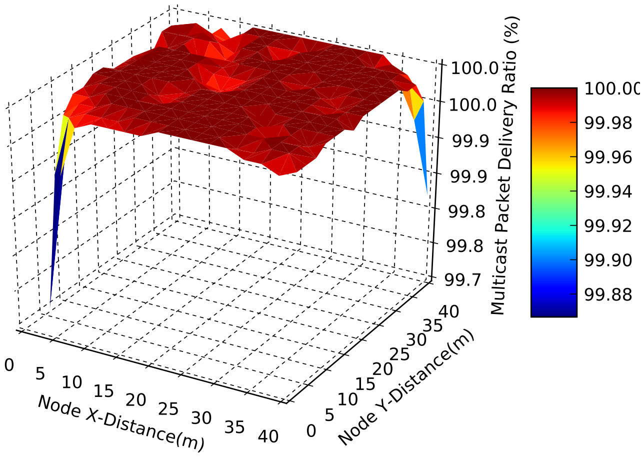


Figure 5.14: Average Packet Delivery Ratio for 256 nodes with 2 seed

However, this is not the case for multi-seed scenarios. As shown in Figure 5.14, PDR is 100% for most of the nodes and drops at the diagonal corners to ≈ 99.7%. This is due to increase in Mulicast End-to-End delay which results in late reception of packet at farthest end. In this case, Node 1 is farthest to Seed 2. By the time Seed 2 multicast packet reaches Node 1, Seed 1 (which is very close to Node 1) transmits next multicast packet (of higher sequence number), thereby resulting in packet dropping. But again, the packet loss itself is low compared to the number of packet received ($approx$2 every 500 packets).

Moving on to PDR in 4 seed scenario, the average PDR is 100% for nodes in the center, it drops to 98.5% at average distance ≈ 20m from seed nodes as shown in Figure 5.15. PDR drops further to 98.2% at diagonal corners. This again is expected as number of multicast messages have increased with increase in packet drops at each node. The nodes concentrated in the center are almost equidistant from all seed nodes. Multicast End-to-End delay is same from the seed nodes, Multicast Forwarding Delay too is similar.

5.4.4. Analysis of Multicast Forwarding Delay

Multicast Forwarding Delay (seconds) is collected in network layer of Thread implementation model. This section analyses Multicast Forwarding Delay for 100 and 256 node single and multi-seed scenarios.It is expected that only routers in the network display MPL forwarding delay.Additionally, the delay is expected not to exceed 128msec. This

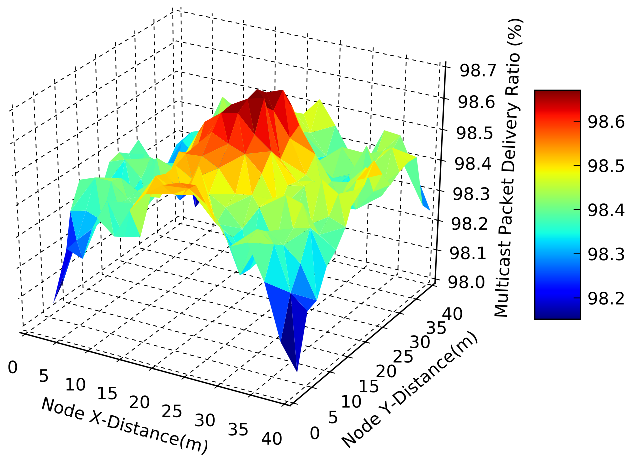


Figure 5.15: Average Packet Delivery Ratio for 256 nodes with 4 seed

is due to the way trickle algorithm is implemented in Thread. Trickle time interval is set to 64 msec. The trickle interval remains same for multiple forwarding of same MPL message. Number of times a thread router will forward an MPL packet is always 2.

100 NODE SCENARIO

Figure 5.16 shows maximum Multicast Forwarding Delay (sec) for 100 node 1 seed scenario. It is observed that the maximum delay is 127.989ms. Therefore, the delay lies within the expected value of 128ms.

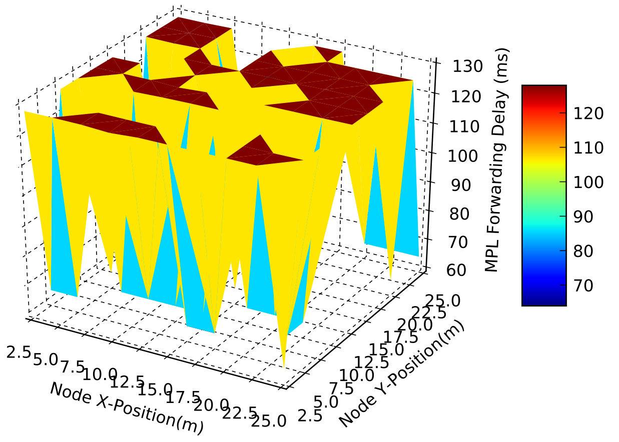


Figure 5.16: Multicast Forwarding Delay for 100 nodes with 1 seed node

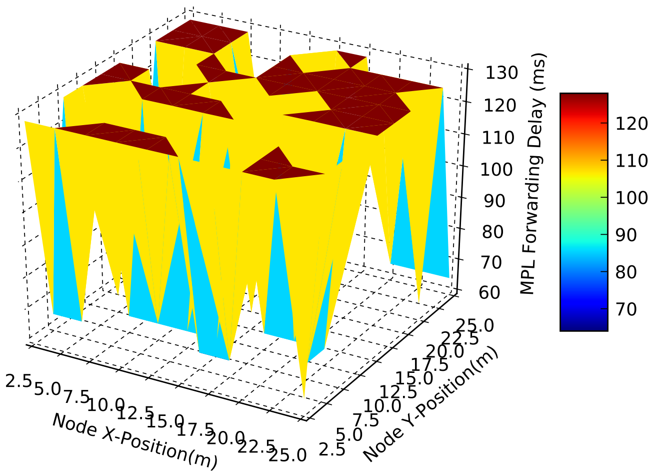


Figure 5.17: Multicast Forwarding Delay for 100 nodes with 2 seed nodes

Furthermore, it can be observed that the graph gives an idea on concentration of routers in the network. Regions with very high MPL forwarding delay in the graph, show the presence of routers. Similarly, Multicast Forwarding Delay in 100 nodes 2 seed scenario, the values lie within 128ms, with maximum value being ≈ $128ms$ as expected. Figure 5.17 depicts maximum Multicast Forwarding Delay possible in 2 seed 100 node scenario, along with the router distribution in the network.

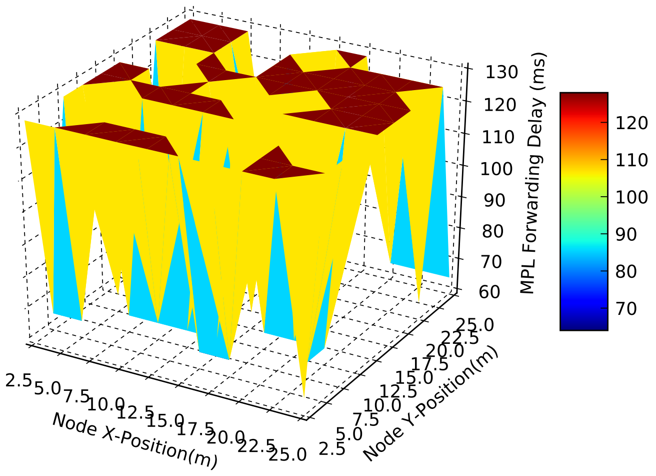


Figure 5.18: Multicast Forwarding Delay for 100 nodes with 4 seed nodes

However, in 4 seed 100 node scenario, it is observed that the maximum Multicast Forwarding delay is 129.674ms. The extra delay can be attributed to MAC delay (of about 3.5ms). Figure 5.18 depicts maximum Multicast Forwarding Delay possible in 4 seed 100 node scenario.

256 Node Scenario

Figure 5.19 shows maximum Multicast Forwarding Delay (sec) for 256 node 1 seed scenario. It is observed that the maximum delay is 130.41ms. The extra delay can be attributed to MAC delay (of about 3.5ms). In 2 seed 256 node scenario, it is observed that the maximum Multicast Forwarding delay is 134.22ms.

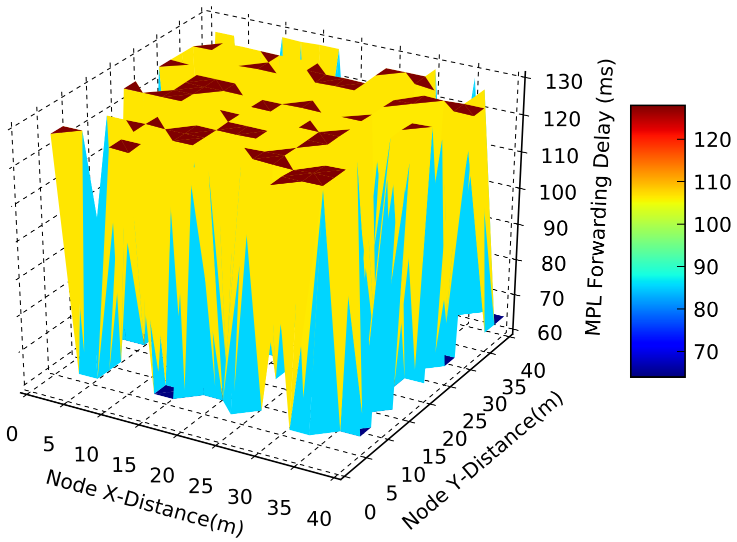


Figure 5.19: Multicast Forwarding Delay for 256 nodes with 1 seed node

Figure 5.20 depicts maximum Multicast Forwarding Delay possible in 2 seed 256 node scenario. The extra delay can be attributed to MAC delay (of about 3.5ms). Additionally, another reason for the extra delay is that each router starts its trickle algorithm after it receives new MPL packet. Whereas, Multicast Forwarding Delay is calculated from the absolute time. It does not consider the relative start time of Trickle at each node. Hence, the extra delay in addition to MAC delay can be accounted to the time taken for the MPL packet to reach that router.

Figure 5.21 depicts maximum Multicast Forwarding Delay possible in 4 seed 256 node scenario. Here too, the maximum value is 136.817ms, which exceeds 128ms. The reasons can be accounted to the MAC delay and the delay

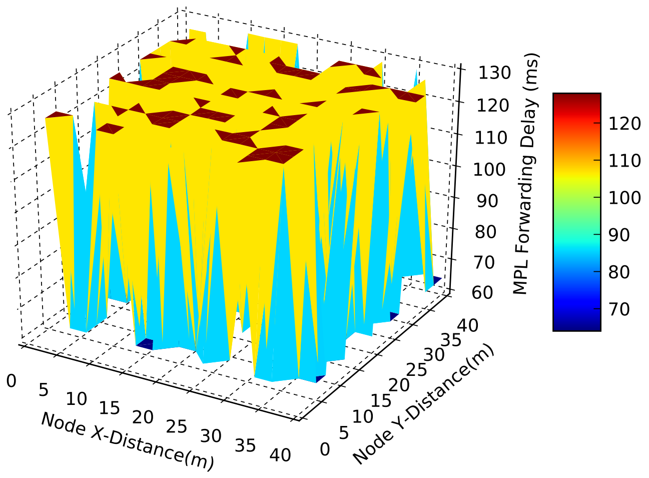


Figure 5.20: Multicast Forwarding Delay for 256 nodes with 2 seed nodes

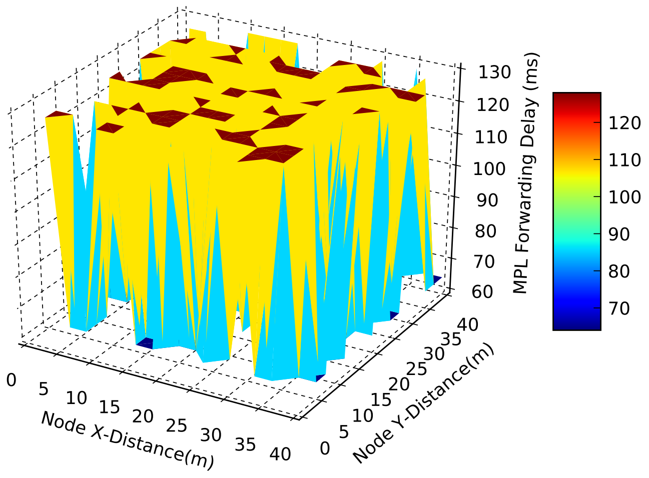


Figure 5.21: Multicast Forwarding Delay for 256 nodes with 4 seed nodes

5.5. SCALABILITY OF THREAD

This test was conducted mainly to determine the behavior of implemented Thread model as the network size increases. It was also conducted to determine upper limit for the model to work accurately. It was observed that tool becomes unreliable and cease to work for conducting test on node size higher than 1000.

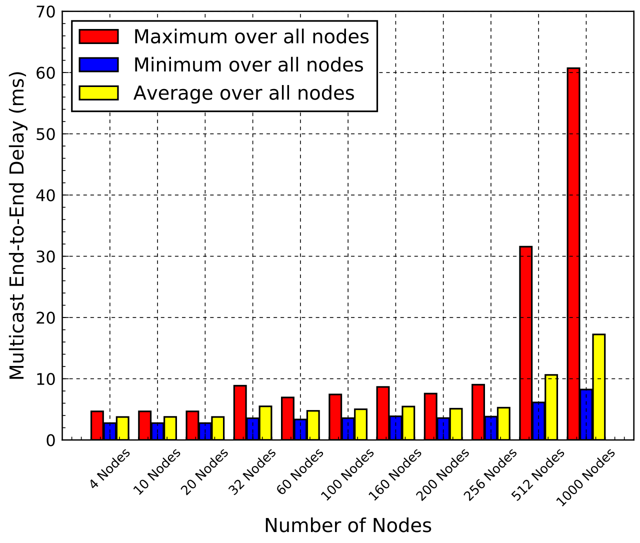


Figure 5.22: Multicast End-End delay for different network topologies

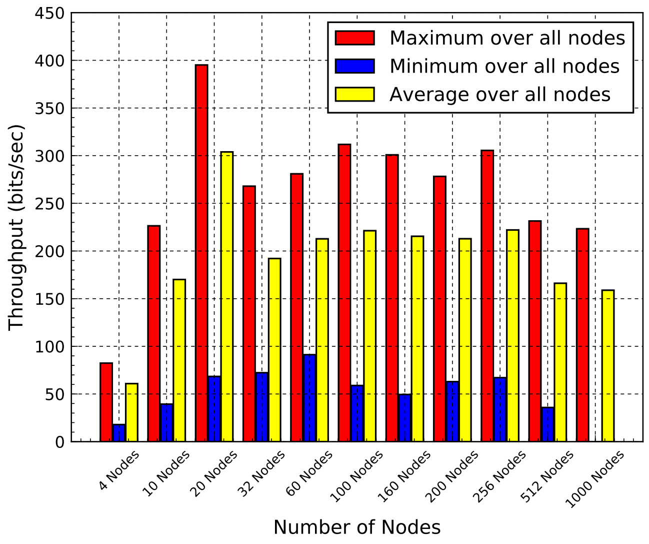


Figure 5.23: Throughput for different network topologies

Figure 5.23 depicts Throughput(sec) for different network topologies. It is observed that Throughput increases as the network size increases till 25 node and then starts to decrease. Hence, Thread model performs it's best for network size in the range 25-30 nodes. Even if decrease in Throughput are observed for increasing network size after 30 nodes, the decrease itself is not very drastic in higher network size as observed in the Figure referred.

Figure 5.22 depicts Multicast End-End delay(sec) . Average Multicast End to End delay increases with increase in network size. This gives an insight about variability of performance parameters with network size. It can be stated that Thread performance parameters varies slowly with increasing the network size.

These results are similar to the experimental results on multicast for scalability conducted by Silicon Labs. The results are confidential, hence, it cannot be referred here. However, it conforms that the implemented Thread model is reliable to conduct large-scale performance simulations. Additionally, the results can be useful and be a basis for real time hardware experiments of Thread large-scale scenarios.

5.6. Network formation

A 35 node line topology test was conducted to evaluate the correctness of Thread network formation process in model implementation. It is assumed that all nodes are Router capable. "Trickle expirations limit" is set to 0.

Test scenario: 35 node line topology

Each node was places 10m apart as shown in Figure 5.24. Path loss exponent (γ) was set to 4.5, making the radio range of each node ≈ 9.8m. All other values of the attributes remain same, as defined in the start of this Chapter. It is expected that the number of

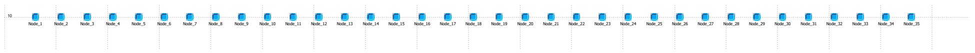


Figure 5.24: 35 node line topology

routers in the network reach maximum number of routers possible in network (32) and remain 32. The number of routers cannot reduce below 32 in this test as, all the network downgrade conditions are not satisfied. Additionally, a single REED should be present with 2 un-connected nodes.

Figure 5.25 depicts variation of router count, REED count and un-connected nodes count in the network, across complete simulation time.

It is observed that a node takes 6-8 seconds to assign itself as a Border Router. Number of routers in the network increase drastically as they do not find any suitable parent (as implemented in parent selection and REED to router upgrade process, Chapter 4) in their radio range. Network convergence time was observed to be 144.22 seconds. The number of routers in the network reach 32 and remain 32. Additionally, there is 1 REED and 2 un-connected nodes in the network.

This test proves that all the network formation processes are working as expected.

Scenario: 100 nodes

For 100 node scenario, average number of routers range from 15-18, as observed in Thread Matlab model. Figure 5.26 shows Thread model implementation of 100 nodes

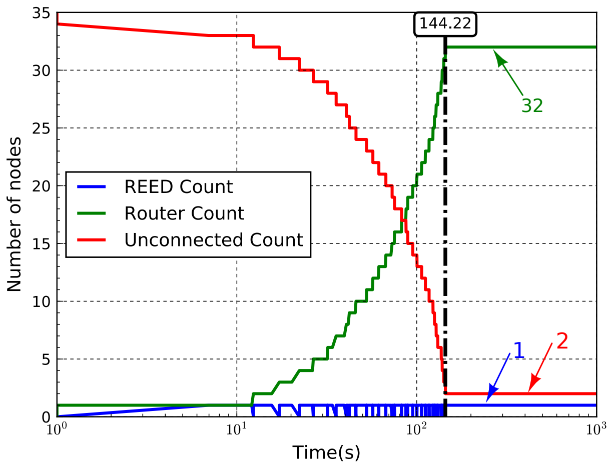


Figure 5.25: Number of REEDs, routers and un-connected nodes during network formation for 35 nodes

scenario for network formation. Here, each node is placed 2.5m apart and path loss exponent is considered as 3, with radio range of ≈ 31m.

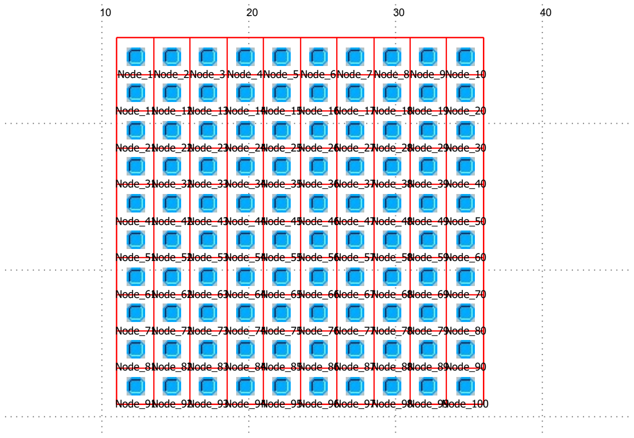

10
20
30
40
Node_1 Node_2 Node_3 Node_4 Node_5 Node_6 Node_7 Node_8 Node_9 Node_10
Node_100

Figure 5.26: Geometric layout of 100 nodes grid for network formation

The scenario was simulated for 5005 (simulation)seconds. Figure 5.27 depicts variation of router count, REED count and un-connected nodes count in the network across complete simulation time.

Following can be observed from the figure: After the assignment of Border Router at around 9.45 seconds, un-connected nodes become part of the network rapidly and become REEDs. This is due to dense network setting, with Border Router in the range of almost half of the total number of nodes. Here, we can also observe very slow increase in routers in the network. From 10s to 30 seconds in the graph, REED to router upgrade

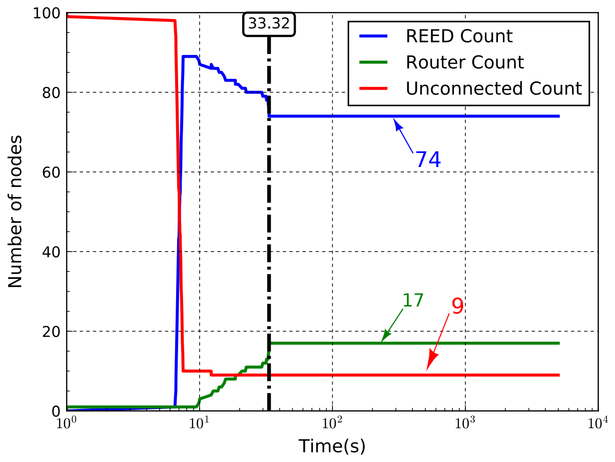


Figure 5.27: Number of REEDs, routers and un-connected nodes during network formation for 100 nodes

process can be observed. Router to REED downgrading process is also observed at certain times. Network converging time is 33.32 sec for this scenario.

It is observed that 9 nodes remain unconnected in the network after the converging time. This is due to non- uniform distribution of routers in the network. Concentrating and confining routers to some parts of the network only. Hence, the network is not available to all the nodes for this scenario. Therefore, this network topology and configuration is not good for 100 nodes

SCENARIO: 256 NODES

Figure 5.28 depicts arrangement of nodes in 16×16 grid, placed 2.5m apart, with a path loss exponent$(\gamma) = 3$ and radio range of $\approx$ 31m. For 256 node scenario, the Matlab mathematical model predicts the average number of routers in the network to be in the range of 23-25.

The scenario is simulated for 5005 seconds. Figure 5.29 shows variation of router count, REED count and un-connected nodes during full simulation time. Once Border Router is assigned to the network, there is rapid increase in number of REEDs in the network (decrease in un-connected nodes in the network) as observed from 8 sec to 10 sec in the graph. There is gradual increase in number of routers in the network from 11.188 sec and till 20 sec. Here, we can observe REED to router upgrade till the number of router in the network reach maximum of 32. From 20 sec, we observe router downgrading to REED to reach an optimal value of 23 at 32.44 sec. The number of routers remain 23 thereafter, throughout the simulation.

Though it is expected for the number of un-connected nodes to become 0 gradually, for this particular scenario, there are 78 number of unconnected nodes in the network.

Hence, this network topology and configuration is not good for 256 nodes.

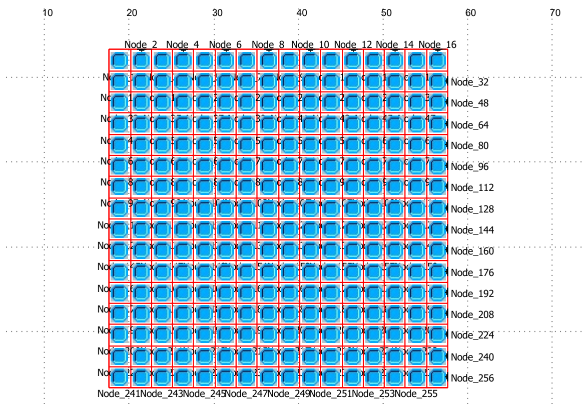


Figure 5.28: Geometric layout of 256 nodes grid for network formation

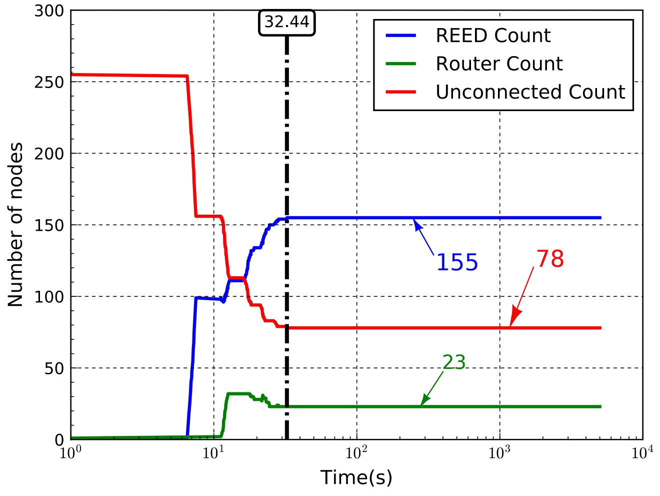


Figure 5.29: Number of REEDs, routers and un-connected nodes during network formation for 256 nodes

5.7. ZIGBEE MODEL FOR MULTICAST SCENARIO

A Zigbee network model is simulated in OPNET 14.5, for comparison. The model is completely developed, maintained and provided by *Philips Lighting* as a modified version of the Zigbee network protocol, whose complete details may not be disclosed in this work.

Figure 5.30 shows the geometric layout of 100 nodes in the form a 10×10 grid. Furthermore, the seed node, in this case called the *Coordinator* node, is also shown to be placed close to *Node_1*, with a relative distance of $X = 2.5m$ and $Y = 2.5m$. Multicast packet is transmitted every 10sec by coordinator node.

Table 5.5 shows important performance parameters obtained by the simulation of the Zigbee network. All the multicast performance statistics described here was pre-

Table 5.5: Table showing various Zigbee global performance parameters

Parameter	Maximum	Minimum	Average
Throughput(kbps)	125.42	2.099	114.23
Delay(ms)	6.33	6.05	6.19
Load(bits/s)	316	0	150.52
Number of neighbours per node	54.34	0.237	27.24

defined as global parameters to the network. Hence, node-wise value of them are unavailable. Delay(ms) mentioned in the table refers to Multicast End-to-End delay observed in the network.

It is difficult to compare Thread multicast performance parameters with the results of Zigbee model for this scenario as attributes configuration and model implementation is not completely known.

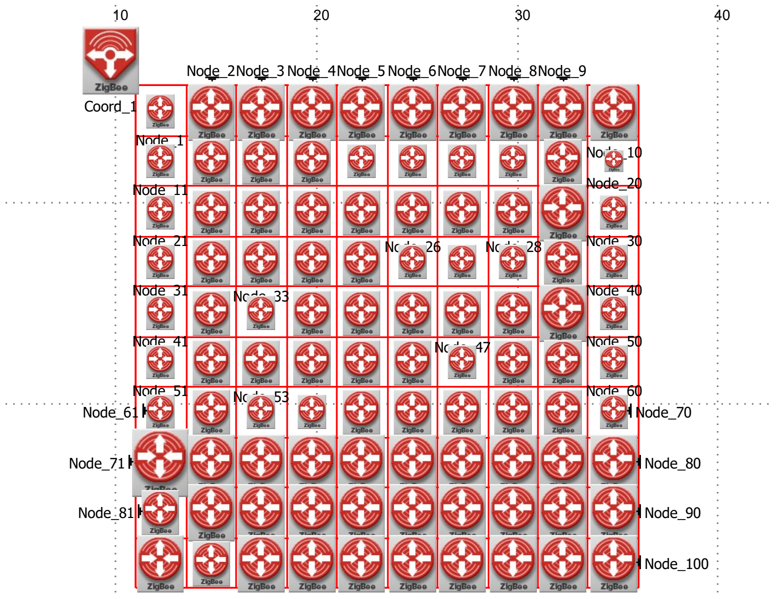


Figure 5.30: Geometry grid layout of ZigBee network provided by *Philips*

In summary, overall latency is high for 256 node scenario when compared to 100 node scenario. 127.83 ms is the average Latency possible in the scenarios. 200ms latency is the average value as mentioned in public document [28] and in the thesis [29] for the network communication part alone. Worst case latency stated by OpenAIS [30] is 400ms for total packet processing and network communication.The observed worst case latency in this study is 307 ms.

Thread is a developing standard. Hence, there were few errors, missing functional description in the specification document. This was identified during thesis research and notified to Thread Group.

6 CONCLUSIONS

6.1. CONCLUSION

The thesis study summarizes Thread network protocol implementation and performance analysis for large-scale network scenarios.

Chapter 1 introduces WSN, Thread and Thread networking features. The Thread network architecture is studied extensively through a simplified Thread network Matlab model that complies with Thread 1.1.1 specification.

Chapter 2 provides a study on related technologies to understand current industry standard for IoT applications. It is observed that the current technologies and Thread share a common 802.15.4 specification. However, Thread provides greater inter-operability in comparison to the Zigbee standard, which currently is widely used, industrial standard for professional lighting applications. Additionally, simplified Thread network model discussed in this chapter provided a basis for Thread model implementation and analysis in OPNET 14.5.

Chapter 3 discusses the Thread architecture and its functionalities as described in Thread specification 1.1.1. The chapter places greater focus on Thread network functionalities, such as, MLE, parent selection process, router/REED upgrade and downgrade process and neighbor detection process.

Important parameters, required to evaluate Thread's network performance were identified in Chapter 4. The OPNET Modeler 14.5 model implementation specifically concentrates on Multicast, MLE and Neighbor discovery functionalites, without fully implementing other features irrelevant for this study. Furthermore, Chapter 4 details the implementation of a large-scale mesh network scenario with 100 nodes and 256 nodes. The implemented models aim to additionally study Thread's scalability features in terms of number of network nodes.

Chapter 5 depicts performance of the implemented model. In Multicast performance study, for each of the scenarios, Latency and Packet Delivery Ratio (PDR) for single and multi seed are studied. It is observed that Latency increases with increase in the seed nodes in the network for each scenario. This is attributed to increase in number of collisions in the network, MAC delay, hop-count in addition to forwarding delay at routers. Worst and best case values of all the scenarios under consideration are reported.

Packet Delivery Ratio decreases with increase in number of seeds for a given scenario. This too can be attributed to increase in collisions in the network. It is to be noted that Packet Delivery Ratio remains similar across different scenarios for same number of seeds.

Hence, latency and packet loss in Thread lies within defined values in this study. The network formation process is fairly fast and reliable. The values seem to be in accordance with the requirements of Lighting application. Though network convergence time was fairly low, there were a lot of unconnected nodes in the scenarios under consideration. The model should be analyzed for other topologies and network configurations to determine optimal topology for this model. Additionally, Thread network partition is not implemented in this Thread model, which can also be the reason for large un-connected nodes in the analyzed scenarios.

Though Thread seem quite suitable for Lighting applications, it cannot be fully determined till real hardware experiments are conducted for large-scale scenarios.

Some of the improvements which Thread specification can provide is flexibility in maximum number of routers in the network, based on the application. Additionally, the Thread documentation flow is not sequential. All the functionalities and assumptions are not defined fully or clearly. For example, the call flow for MLE Parent selection and neighbor detection was not complete and clear. Hence, some assumptions had to made during implementation. This can also be due to lack of complete understanding of all the functionalities in Thread protocol.

6.2. FUTURE WORK

This study involved complex network modeling, implementation and analysis of the Thread Protocol. All functionalities defined in Thread 1.1.1 specification have not been fully implemented. Additionally, the 1.1.1 version of the specification is itself recent and was released during the second half of the Thesis study. Further work in this study involves implementation of new and related functions according to the 1.1.1 revision will continue at Philips Lighting Research. Furthermore, the implementation will be extended to large-scale real hardware implementation, which was not possible within the scope of this thesis.

Implementation of Thread protocol was challenging since there was lack of support and documentation. Additionally, challenges posed by OPNET 14.5 tool made implementation and results collection complex. Looking into OpenThread can be a good option to use for hardware implementation as it provides certified full stack software implementation. Hence, implementation time can be reduced drastically. The challenge, however, is to porting OpenThread on top of a simulator framework, which has to be looked into.

ACKNOWLEDGEMENTS

First and foremost, I would like to extend my gratitude to my teacher, my supervisor at TU Delft, Dr.ir. Fernando Kuipers, for all his support and guidance throughout the study. Without him, this thesis would not be possible. He has always been very patient, understanding, motivating and have provided valuable thought provoking insights into any problem that came up during my study period. I really thank you very much Professor! I hope to learn from you more in future.

I would like to thank my supervisors, Dr. Xiangyu Wang and Dr.ir.Esko Dijk for providing an opportunity to conduct my thesis at Philips Lighting Research. Mr. Luca Zappaterra along with my supervisors have constantly taught, guided, motivated and supported me throughout the thesis. They have provided me with flexibility to work independently and have always patiently helped me throughout the study. It was a privilege to work with you all. I am very grateful and thank you once again.

I would also like to thank Mr. Francisco Estevez for his constant support and cooperation while working with OPNET. I would like to thank Mr. John Mills, Ms. Winkie Visser and Ms. Marije Lambers, along with all the members of IoT Systems group, Philips Lighting for all the support and providing me a great learning experience.

I would like to thank my Academic Couselors - Ms. Agaby Masih and Ms. Susanne van Aadrenne, who have provided me all the assistance and support when I need the most. They have motivated me constantly to work and complete my study here.They have helped me face difficult times with courage and confidence. I am very grateful to you both. Additionally, I would like to extend my thanks to student psychologists at TU Delft, who have helped me through my difficult times.

I would like to thank my parents, Mrs. Lakshmi VS and Mr. Shamsundar BR. They are my first teachers and I thank them for their constant support, unconditional love, teachings and sacrifices. I would also like to thank my sister Disha Shamsundar, for her understanding, unconditional love, support and sacrifice. Without you all, I would have not completed this study.

I would like to thank my extended family in Europe - Anand, Vijay, Neelima, Kruthika and Ganesh for taking care of me, helping me and supporting me. They made me feel at home in Europe. I would also like to thank my cousins- Girisha, Kamala and Karuna, who have helped me throughout my studies. Finally, I thank all my friends and my fellow students, Professors and staff at TU Delft for all the support, help and great memories.

REFERENCES

[1] F. Osterlind, E. Pramsten, D. Roberthson, J. Eriksson, N. Finne, and T. Voigt, "Integrating building automation systems and wireless sensor networks," in *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on.* IEEE, 2007, pp. 1376–1379.

[2] A. Pandharipande, D. Caicedo, and X. Wang, "Sensor-driven wireless lighting control: System solutions and services for intelligent buildings," *IEEE Sensors Journal*, vol. 14, no. 12, pp. 4207–4215, 2014.

[3] V. C. Gungor and G. P. Hancke, "Industrial wireless sensor networks: Challenges, design principles, and technical approaches," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 4258–4265, 2009.

[4] E. Aljarrah, M. B. Yassein, and S. Aljawarneh, "Routing protocol of low-power and lossy network: Survey and open issues," in *2016 International Conference on Engineering MIS (ICEMIS)*, 2016, pp. 1–6.

[5] C. Reinisch, W. Kastner, G. Neugschwandtner, and W. Granzer, "Wireless technologies in home and building automation," in *2007 5th IEEE International Conference on Industrial Informatics*, vol. 1, 2007, pp. 93–98.

[6] 802.15 - Wireless Personal Area Network (WPAN) Working Group. (2006) IEEE 802.15.4-2006 - IEEE standard for information technology– local and metropolitan area networks– specific requirements– part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low rate wireless personal area networks (WPANs). [Online]. Available: https://standards.ieee.org/findstds/standard/802.15.4-2006.html

[7] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. (2007) IETF RFC4944: Transmission of IPv6 packets over IEEE 802.15.4 networks. [Online]. Available: https://www.ietf.org/rfc/rfc4944

[8] P. Thubert and J. W. Hui. (2011) Compression format for IPv6 datagrams over IEEE 802.15.4-based networks. [Online]. Available: https://tools.ietf.org/html/rfc6282

[9] J. Hui and R. Kelsey. (2016) Multicast protocol for low-power and lossy networks (MPL). [Online]. Available: https://tools.ietf.org/html/rfc7731

[10] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. (2012) RPL: IPv6 routing protocol for low-power and lossy networks. [Online]. Available: https://tools.ietf.org/html/rfc6550

[11] Thread Group. (2017) Thread 1.1 specifications. [Online]. Available: https://threadgroup.org/ThreadSpec

[12] Riverbed SD-WAN, WAN optimization, app performance, SD-edge solutions. [Online]. Available: https://www.riverbed.com/nl/index.html

[13] Z. Lu and H. Yang, *Unlocking the power of OPNET modeler.* Cambridge University Press, 2012.

[14] T. D. P. Mendes, R. Godina, E. M. G. Rodrigues, J. C. O. Matias, and J. P. S. Catalão, “Smart home communication technologies and applications: Wireless protocol assessment for home area network resources,” *Energies*, vol. 8, pp. 7279–7311, 2015.

[15] L. Mainetti, L. Patrono, and A. Vilei, “Evolution of wireless sensor networks towards the internet of things: A survey,” in *SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks*, 2011, pp. 1–6.

[16] Zigbee Alliance. [Online]. Available: http://www.zigbee.org/

[17] TCLA. (2016) The connected lighting alliance endorses ZigBee 3.0 for residential lighting | zigbee alliance. [Online]. Available: http://www.zigbee.org/the-connected-lighting-alliance-endorses-zigbee-3-0-for-residential-lighting/

[18] Z-Wave Alliance. [Online]. Available: https://z-wavealliance.org/

[19] D. Lan, “Experimental study of thread mesh network for wireless building automation systems,” MSc. Thesis, KTH, School of Electrical Engineering (EES), 2016. [Online]. Available: http://kth.diva-portal.org/smash/record.jsf?pid=diva2:1040491

[20] OpenThread. [Online]. Available: https://openthread.io/

[21] N. Labs. (2016) Nest announces open source implementation of thread. [Online]. Available: https://www.nest.com/at/press/nest-announces-open-source-implementation-of-thread/

[22] H. Gonzalez, “Study of the protocol for home automation thread,” MSc. Thesis, Universitat Politecnica de Catalunya, 2017. [Online]. Available: https://upcommons.upc.edu/bitstream/handle/2117/101928/memoria.pdf

[23] J. Hui, D. Culler, and S. Chakrabarti. (2009) 6lowpan: Incorporating IEEE 802.15.4 into the IP architecture. [Online]. Available: http://www.ipso-alliance.org/wp-content/media/6lowpan.pdf

[24] P. Levis and T. H. Clausen. (2011) The Trickle Algorithm. [Online]. Available: https://tools.ietf.org/html/rfc6206

[25] J. Postel. (1980) User datagram protocol. [Online]. Available: https://tools.ietf.org/html/rfc768

[26] R. Braden. (1989) RFC 1122. [Online]. Available: https://www.ietf.org/rfc/rfc1122

[27] J. Postel. (1981) Transmission control protocol. [Online]. Available: https://tools.ietf.org/html/rfc793

[28] A. Somaraju, S. Kumar, H. Tschofenig, and W. Werner, “Security for low-latency group communication,” 2016. [Online]. Available: https://tools.ietf.org/id/draft-somaraju-ace-multicast-01.txt

[29] C. Nie, “Researching the BLE based intelligent lighting control system,” MSc. Thesis, TU Delft, 2014. [Online]. Available: http://resolver.tudelft.nl/uuid:e6141160-b3a0-405f-ba6d-85e305c138fb

[30] O. Consortium, “Final reference architecture of OpenAIS system,” *OpenAIS: Open Architecture for Intelligent Solid State Lighting Systems*, 2014. [Online]. Available: http://openais.eu/user/file/openais_final_reference_architecture_(d2.7)_v1.0-pub.pdf