DISASTER RESILIENT MESH NETWORK USING LORA AND NERVENET

LEAN CHEE HONG

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Electronics (Computer Networking) Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

April 2022

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	[illegible]
Name	:	Lean Chee Hong
ID No.	:	1702976
Date	:	13/05/2022

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"DISASTER RESILIENT MESH NETWORK USING LORA AND NERVENET"** was prepared by **LEAN CHEE HONG** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Electronics (Computer Networking) at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	Tham Mau Luen
Date	:	12 May 2022

Signature	:	
Co-Supervisor	:	
Date	:	

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Tham Mau Luen for his invaluable advice, guidance and his enormous patience throughout the development of the project

In addition, I would also like to express my appreciation to Dr. Yasunori Owada from NICT, Japan. He patiently provided assistance whenever there were bottlenecks or technical issues throughout the entire project.

ABSTRACT

When a natural disaster event happens, it could cause regional cellular network outages and hence disable network communication within the affected area. If a resilient network is implemented, alert messages with sufficient information can be sent over the Internet to provide a nationwide response. Japan National Institute of Information and Communication Technology has invented a resilient network framework called NerveNet, it supports mesh network where each node will approach other nodes in range if the current peer no longer responds. Using their technology, disaster nodes could be installed at disaster hotspots to send out disaster information or even provide light internet services. NerveNet does support data communication using Wi-Fi and LoRa. NerveNet Wi-Fi-Mesh links are used to provide wide bandwidth but low range data transmission, while NerveNet LoRa-Mesh supports narrow bandwidth data transmission in coverage of kilometers, which is suitable for crucial or emergency disaster data updates.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

ABP	Activation By Personalization
AP	Access Point
API	Application Programming Interface
BS	Base Station
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines Corporation
IP	Internet Protocol
IoT	Internet of Things
ISM	Industrial, Scientific, and Medical
JSON	JavaScript Object Notation
LAN	Local Area Network
LoRa	Long-Range
LoRaWAN	Long-Range Wide Area Network
LPWAN	Low-Power Wide Area Network
MAC	Media Access Control
MHDR	Media Access Control Header
MQTT	MQ Telemetry Transport
MType	Message Type
NDCC	Natural Disaster Command Centre
NICT	National Institute of Information and Communications Technology
P2P	Peer-to-Peer
OLSR	Optimized Link State Routing
OS	Operating System
OSI	Open Systems Interconnection
OTAA	Over-the-Air-Activation
PDU	Protocol Data Unit
REST	Representational State Transfer
SF	Spreading Factor
SIP	Session Initiation Protocol

SSL	Secure Sockets Layer
TCP	Transport Control Protocol
TLS	Transport Layer Security
VLAN	Virtual Local Area Network
WAN	Wide Area Network
WMN	Wireless Mesh Network

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

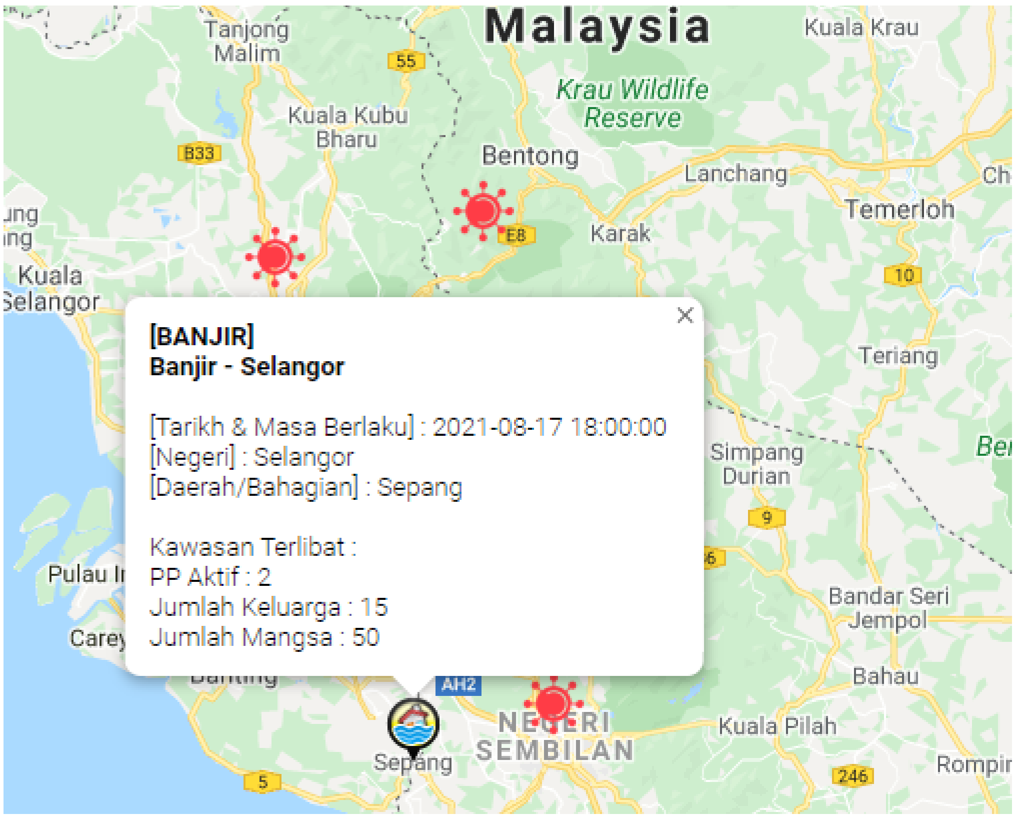
1.1 General Introduction

Malaysia is a Southeast Asia country bordered by Thailand, Indonesia, and Brunei, located outside of major typhoon paths and the Pacific Ring of Fire. Despite lies in a geographically stable region, Malaysia is still facing the risk of floods, landslides, and other human-made disasters. According to Lee and Noorazurah Mohamad of Universiti Teknologi MARA (2013), floods in Malaysia cause RM242 million of economic losses per average annually. Another study in the year 2019 by Center for Excellence in Disaster Management and Humanitarian Assistance (CFE-DMHA) stated that Malaysia had experienced 51 natural disaster events from the year 1998 to 2018, causing 281 people to die and more than 3 million people were affected, which caused around RM8 billion in damages.

The main warning system for disaster events in Malaysia is using SMS to alert residents of impending disaster risks. This method is highly dependent to the availability of a cellular network due to its tree topology property. If the disaster destroys the cellular network base station, communities are said to be isolated from internet services. Not only failed or delayed receiving the emergency alert, but it also increases the difficulty for rescue squad to obtain the lastest information using cellular communication service. Japan, a country with a high natural disaster rate, has been using a resilient mesh network named NerveNet to overcome this challenge. NerveNet has been developed by the National Institute of Information and Communications Technology (NICT) in Japan since the year 2006. According to Inoue and Owada (2017), NerveNet is conducted at a large scale of testbed with 30 base stations constructed within Tohoku University in Sendai at the year 2011. Then in the year 2014, NICT started operating NerveNet that deployed in real environments for disaster prevention purposes. NerveNet end devices do not rely on the availability of each other. One node goes inactive does not affect the overall service provided as other nodes will self-configure a new pathway to transfer data. Logically, any node can peer with any other nodes if they are under NerveNet network, which gives it fault-tolerance property during disaster events.

1.2 Importance of the Study

The risk of flooding in Malaysia is increasing year by year due to sea-level rise, and some studies point out that several seaside cities will be underwater by the year 2050. In addition, climate change also increased extreme weather events, such as heat waves that lead to wildfire, heavy rainfalls, and inland flooding. Severe floods could cause more drowning deaths and will lead to serious economic losses. Natural Disaster Command Centre (NDCC) is a center for disaster operation control in Malaysia. Figure 1.1 shows disaster information presented on their website.



[BANJIR]
Banjir - Selangor

[Tarikh & Masa Berlaku] : 2021-08-17 18:00:00
[Negeri] : Selangor
[Daerah/Bahagian] : Sepang

Kawasan Terlibat :
PP Aktif : 2
Jumlah Keluarga : 15
Jumlah Mangsa : 50

Figure 1.1: NDCC Disaster Map.

As shown in the figure, NDCC posts the information based on location, DateTime, victims, and disaster type. To ensure the collected information can be sent over Internet during disaster event, the data synchronization feature is another necessary component for a resilient mesh network. This feature allows all wirelessly connected base stations to share their database in such a way that any nodes could send information to the gateway if the detection node is destroyed. If the end stations could also send regional disaster data to the gateway from kilometers away wirelessly, the network is said to be perfect for a disaster management system.

1.3 Problem Statement

The resilient regional network is not populated in Malaysia, residents are vulnerable as disaster management and response are not guaranteed. With a resilient network implemented, alert messages with sufficient information can be sent over the Internet to provide a nationwide response. The database synchronization feature is also crucial for storing disaster sensors data over the network, and it also allows applications to pull data from nodes to the cloud. Cellular networks in Malaysia are usually vulnerable to disasters and using Wi-Fi transmission on disaster nodes could be one of the solutions. However, the range of Wi-Fi transmissions is short, and the cost of implementing a resilient disaster monitoring network system in a wide coverage range could be very high. Hence, LoRa is a better solution in this case, it supports wireless data transmission protocol and signal coverage in kilometers, which requires lesser cost to construct disaster nodes at a given range as compared to Wi-Fi.

1.4 Aim and Objectives

This study aims to develop a disaster response application using NerveNet LoRa mesh network. The objectives of this study are:

i. To establish a NerveNet Wi-Fi mesh network testbed with data synchronization.
ii. To establish a NerveNet LoRa mesh network testbed.
iii. To generate Python codes that handle the NerveNet LoRa and Wi-Fi mesh network data exchange.
iv. To evaluate the network performance of the NerveNet LoRa and Wi-Fi mesh.

1.5 Scope and Limitation of the Study

The scope of this project is to set up a testbed that could be used in natural disaster hotspots. The testbed should be resilient and able to transmit import disaster information so that the respective agency could take action immediately. NerveNet framework is designed and owned by Japan, the resources and references are regarding NerveNet structure are not disclosed to the public, this may increase the difficulty of investigation and research. Therefore, the project's progress was dependent on the sufficiency of documentation provided by NICT. Apart from that, some of the documentation may be written in Japanese, and language translation may not reveal the exact meaning of words.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

To design a disaster-resilient mesh network, one of the considerations is fault tolerance. Most of the communication service provided in Malaysia is tree topology based, low-level network nodes or devices are highly dependent on the availability of upper hierarchical level devices. When the service is down or congested, the end devices are not able to self-select or self-configure a new pathway because most of them are statically or point-to-point routed. In other words, the number of peers is relatively limited and fixed. Hence, the network connection between nodes should be as dynamic as possible to maximize the number of peer connections, which can be done using mesh protocols.

For disaster detection nodes, the power consumption and wireless data transmission range are one of the concerns. Usually, the power consumption of sensors and actuators itself is low, a power-saving wireless networking technology supporting wide range coverage is most suitable for disaster nodes. In this case, LoRa technology fits the requirement as it supports a range of transmission in kilometers, where LoRa sensors can operate for years with just an alkaline battery.

2.2 Resilient Network

Resilience in terms of computer networking refers to the flexibility and elasticity of faults or errors. A resilient service is said to be reliable as the system is adaptable to extreme circumstances such as natural disasters or malicious network attacks by adopting data synchronization, which is often achieved by the architecture of star, ring, or mesh network topology. Therefore, the need for resiliency certainly applies to critical services and infrastructure, especially if the functionality of the system is automated.

Generally, resilience cuts through several thematic areas, such as information and network security against attack, fault-tolerance, dependability, performability, and network survivability (Hutchison and Sterbenz, 2018).

When it comes to the requirement for disaster warning systems, secured data transmission from nodes to receivers is one of the crucial properties that need to be integrated into the system, which disaster-resilient communication network shall be considered.

As of today, the ability of a network system to provide acceptable and fault-tolerance service is more important than ever before. Information in terms of data is treated as an asset, where resilience is the key to design to maintain data persistence and connectivity when developing a reliable system. A resilient network was designed to aim to provide reliable network service to applications. These services shall include distributed processing, network storage, communication service, and access information.

2.2.1 Distributed Processing

According to IBM Documentation (2014), distributed processing or distributed computing is the use of multiple processors, computers, or software components but run as a single system, which is called distributed computer system. The components can be connected within LAN or WAN, which makes the entire network structure itself works as a single computer to offer benefits such as scalability and redundancy. In other words, the system can be expanded easily, while the same services can be provided by several components to ensure service continuity when one of the machines is unavailable.

2.2.2 Network Storage

Network storage is also known as Cloud storage. The service is said to be “Cloud” because it allows saving data in an off-site location that can be accessed through the Internet or a dedicated private network connection (IBM Cloud Education, 2019). Network storage is a server that stores the centralized data, it responds to the request from networked clients, which is usually in HTTP or WebSocket protocols. In a resilient network, the data pool is ready in network storage to be shared among other components or even accessed externally if needed. Not only update data, but network storage also provides backup for clients revived from death, this allows clients to attempt to recover their states by requesting real-time data from network storage.

2.2.3 Communication Service

Communication shall include computer-supported collaboration, video conferencing, and instant messaging. For distributed disaster warning system, the communication between nodes and gateway is significant to seek for latest data, then establish a data transfer or exchange process. This is the prerequisite for the data synchronization process in both local database and network storage.

2.2.4 Access Information

To make use of synchronized information in database effectively, the resilient network should provide access to the database as needed. Usually, the access feature is built into an application that directly communicates with network storage. This model separates endpoints by dedicating network storage as the middleman between the user end and sensor nodes or data collectors.

2.3 Mesh Network

Mesh in a network system refers to the network topology in which the nodes are fully or partially interconnected as many as possible to form a redundancy connection. In mesh networks, the infrastructure nodes are equipped with self-configure and self-routing protocols, such as BATMAN-adv, Zigbee, Wireless Ad-Hoc, OLSR, and NerveNet.

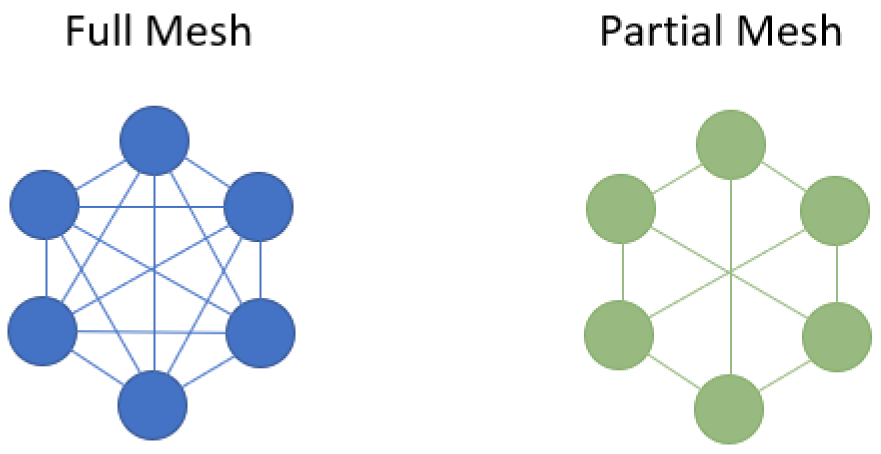


Figure 2.1: Full and Partial Mesh Network Topology.

Mesh networks are getting popular recently due to their efficiency and reliability in data transfer. Since the nodes are interconnected, there is no central point to rely on, which means data packets transfer is lesser dependent on each node. Redundancy connections are the key to achieving fault-tolerance because each node can self-define alternative paths and distribute the workloads based on availability. Moreover, the speed of message transfer also generally increased due to the shortest hop count or route cost of routing path is defined by nodes itself.

2.3.1 Mesh and P2P Network

A P2P network is where each node act as both client and server, they request or respond services to from each other for resource sharing purpose. This concept is quite similar to mesh networks. However, they are different. Usually, P2P nodes store their own data, each node exclusively communicates with another node, which makes the network topology still tree-alike. In most conditions, the purpose of implementing a P2P network is to optimize the usage of bandwidth and also reduce the route traffic between clients and servers. In contrast, adding new mesh nodes does increase the bandwidth usage, but it provides resiliency to the network, which P2P could not.

2.3.2 Wireless Mesh Network

Wireless Mesh Network (WMN) is a mesh network where all the nodes communicate using radio waves instead of Ethernet, the nodes are also known as radio nodes (Rong et al., 2013). In the WMN network, each radio node works as both router and host, they calculate the shortest path distance to forward the packets to the destination when necessary. Usually, a WMN network will consist of a mesh client, mesh router, and mesh gateway, where clients will simply forward packets to the mesh router without computing the path, while mesh router will further send packets mesh gateway so that they can be forwarded to the external network. Thanks to its resiliency and frequent intercommunications, WMN is one of the considerations for developing IoT applications using Wi-Fi or LoRa protocols. However, it is yet to support high mobility nodes such as smart cars because the mesh network connection breaks frequently will cause performance to be reduced.

2.4 LoRa

LoRa (Long Range) is an LPWAN modulation technique patented by LoRa Alliance, which is derived from Chirp Spread Spectrum (CSS) technology (The Things Network, 2021). LoRa is ideal for long-range transmission with relatively low bit rates, data can be transmitted at a wider range as compared to Wi-Fi and Bluetooth, which makes it suites for low-power remote applications such as sensors and actuators (The Things Network, 2021).

Referring to the international agreement, LoRa bandwidth is restricted to 125 kHz, 250 kHz, and 500 kHz, while only 125 kHz and 250 kHz bandwidth are used in Europe countries. The data bit rate is not only dependent on restricted bandwidth but also manipulated by another factor called Spreading Factor (SF), which controls the chirp rate (The Things Network, 2021). The N^{th} number of SF is inversely proportional to the data transmission rate. For example, reducing the SF by one will double the chirp sweep rate, hence double up the data transmission speed too. Higher SF reduces the chirp sweep rate, which is easier to accurately decode the signal, thus increasing the range of LoRa transmission. As a trade-off, a slower sweep carries lesser information. LoRa modulation provides six SF in total, which is from SF6 to SF12. Apart from LoRa transmission rate, end devices that send or receive signals modulated from different SF will not interfere with each other even though they operate at the same frequency channel.

2.4.1 LoRa Standards

LoRa operates on license-free sub-GHz ISM bands, such as 433 MHz, 868 MHz, 915 MHz, or even 2.4 GHz, for higher data transmission rate with the cost of range (The Things Network, 2021). Since the LoRa could transmit data at a high range, regional parameters were set over countries according to their restrictions. As in Malaysia, the frequency plan used is called AS1 (used in Malaysia, Singapore, and Japan), while the range of frequencies is named AS920-923 (Asia 920-923 MHz).

2.5 LoRaWAN

LoRaWAN (Long Range Wide Area Network) is a media access control (MAC) protocol based on LoRa modulation, which is categorized as OSI model layer 2 (data link layer) protocol, or network interface layer in terms of TCP/IP model. LoRaWAN network uses ALOHA based protocol, the messages are transmitted to the transmission channel without acknowledging its availability, thus LoRa end devices do not peer with specific gateways, every gateway within the valid range of transmission channel could receive the message (The Things Network, 2021). The architecture of LoRaWAN network is shown in Figure 2.2 below.

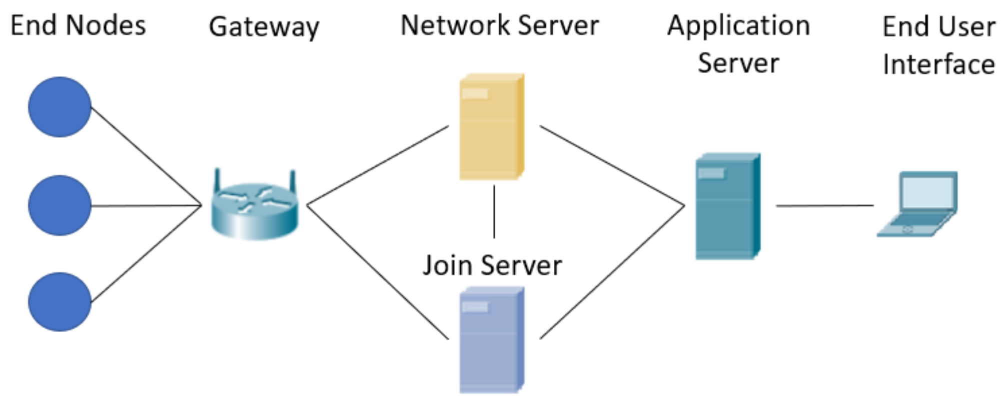


Figure 2.2: LoRaWAN Architecture.

LoRaWAN end nodes are usually small appliances such as sensors or actuators, they are often wireless because just a battery could support their operation for years. LoRaWAN gateway is simply just a relay to forward LoRa messages from end nodes to Network Server via Wi-Fi, Ethernet, fiber optic, cellular, or 2.4 GHz radio links. There is no fixed connection between end devices and LoRaWAN gateway, any gateway can receive LoRa message from any node, as long as within the range. LoRaWAN network server controls the entire network as it receives packets from the gateway, the data communication between gateway and network server is using network layer protocols. Not only routing the messages, but network server also responsible to also eliminates duplicated LoRa messages forwarded by multiple gateways, then eventually forwarding data to the application server. Join server does not participate in data communication if the end node is connected to the LoRaWAN network, it only involves at the over-the-air-activation process. When application server receives a processed message from network server, it provides ready data to the client interface, such as presenting feedback from nodes. Moreover, it also can generate a downlink payload that will be sent back to end nodes, such as shutdown or adjusting parameters of nodes.

2.5.1 LoRaWAN Activation Process

In LoRaWAN activation process, join server is important to manage the over-the-air activation (OTAA) process. It is an authentication process for end nodes to participate in the network, which is always initiated by end nodes only (The Things Network, 2021). Generally, the message from end node to network server is called as uplink message, while the message from network server to end node is called as downlink message.

For LoRaWAN 1.0.x version, OTAA is done between network server and end nodes. Secret key (AppKey or known as root key) and public keys (AppEUI, DevEUI) are stored in end devices before activation process begins. However, AppKey is not included in the Join-request message, it is just used to ensure data integrity. End device is always the one initiate activation process, it sends unencrypted Join-request message to network server. If the Join-request is permitted by network server, it will response an AppKey-encrypted Join-accept message including a DevAddr to identify the end device address within current LoRaWAN network, and then sends a AppSKey to application server. The end device will use the AppKey to derive two more secret key, namely NwkSKey and AppSKey. After activation process, NwkSKey is used between the end device and network server to verify data integrity and payload encryption/decryption if contains MAC command. While AppSKey is used encrypt application payload so that the end-to-end communication between end device and application server is secured.

In lastest LoRaWAN 1.1 version, the OTAA process is done between end nodes and join server. Secret keys (AppKey, NwkKey, known as root keys) and public keys (JoinEUI, DevEUI) are stored in end devices before activation process begins. Same as LoRaWAN 1.0.x , secret keys are not included in the Join-request message. The Join-request message is sent unencryted to network server, the frame contains Join-request message is known as PHYPayload. Then, network server will use the JoinEUI key in received request to DNS lookup the ip address of join server, further sends JoinReq message which contains Join-request to join server if successful of DNS lookup. After join server received the JoinReq message, it will response network server with JoinAns message, which contains Join-accept message, network session keys, SNwkSIntKey, FNwkSIntKey, NwkSEncKey, and AppSKey. Upon received the JoinAns message, network server will generate a Join-accept message using the keys received, then encrypt it using NwkKey and send to the end device. The end device will use its NwkKey to derive SNwkSIntKey, FNwkSIntKey, and NwkSEncKey. On the other side, network server will send an encrypted AppSKey and application payload to the application server. Application server will first decrypt AppSKey using another secret key shared by join server, then decrypt the application payload using AppSKey. If the AppSKey is not available from network server, application server will directly request AppSKey from join server using AppSKeyReq message, then join server will response an AppSKeyAns message with encrypted AppSKey. After activation process, NwkSEnKey is used between the end device and network server to encrypt or decrypt payload on port 0 or FOpt field which contains MAC command. SNwkSIntKey and FNwkSIntKey are used to ensure data integrity of uplink and downlink data message respectively.

There is one alternative of activation process called Activation By Personalization (ABP). ABP will bypass the authentication of join procedure and join server is not used, connection between network server and end node is binded and pre-selected. Hence, ABP is not secure as compared with OTTA. End nodes activated by ABP can only connect with one static network and keeps the same security session for its entire lifetime, while switching of network requires manual change of keys stored in end node.

2.5.2 LoRaWAN Message Types

Generally, there are four data message types used by LoRaWAN as shown in Figure 2.3. These data message types are used to transport MAC commands and application data which can be combined in a single message. Data messages can be classified as confirmed or unconfirmed, where confirmed data message require acknowledgement by the receiver, while unconfirmed data message does not require.

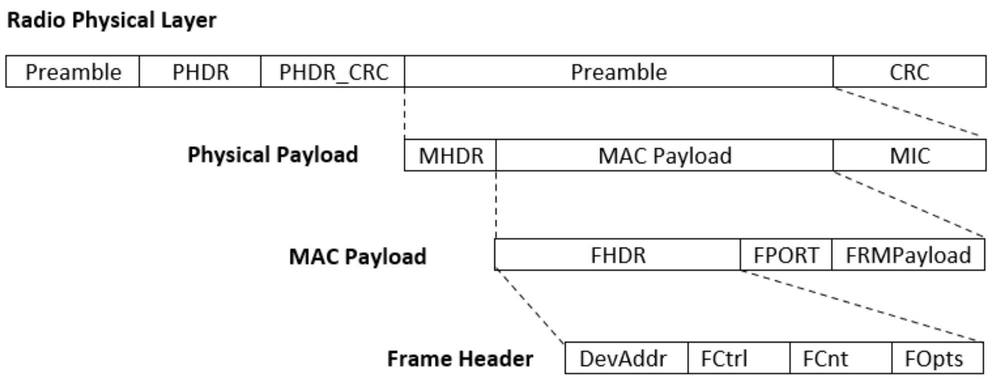


Figure 2.3: LoRaWAN Data Message in Fields.

The first three most significant bits of MHDR field is the identifier of message types, which is also known as MType field (Prajzler, 2019). LoRaWAN 1.0.x version uses 7 kinds of MType, while LoRaWAN 1.1 version have one additional type called Rejoin-request. Similar with Join-request, Rejoin-request is also initiated by end nodes to initialize a new session context. Once network server permitted this request, it will response with a Join-accept message. The MType variations and its description is tabulated as Table 2.2 below.

Table 2.1: MType Variations and Descriptions.

MType Binary	**LoRaWAN 1.0.x**	**LoRaWAN 1.1**	**Description**
000	Join-request	Join-request	Uplink OTAA Join-request.
001	Join-accept	Join-accept	Downlink OTAA Join-accept.
010	Unconfirmed Data up	Unconfirmed Data up	Uplink data frame (confirmation not required).
011	Unconfirmed Data down	Unconfirmed Data down	Downlink data frame (confirmation not required).

100	Confirmed Data up	Confirmed Data up	Uplink data frame (confirmation requested).
101	Confirmed Data down	Confirmed Data down	Downlink data frame (confirmation requested).
110	(Reserved for future use)	Rejoin-request	Uplink OTAA Rejoin-request.
111	Propietary	Propietary	Implementation of non-standard message formats.

2.6 NerveNet

NerveNet is a resilient network developed by National Institute of Information and Communications Technology (NICT) in Japan. NerveNet is a regional-area network to provide reliable Internet access and sharing resilient information platform in emergency situations such that when network from internet service provider no longer persists. NerveNet base stations are interconnected by Ethernet-based wired or wireless transmission systems such as optical/metal Ethernet, Wi-Fi, FWA, satellite, and Unmanned Aerial Vehicle (UAV), which sums up a mesh-topological network (Inoue, et al., 2014). As compared with most common network infrastructures nowadays, such as fixed-line networks and cellular networks, they are both tree topology based, NerveNet is much more fault-tolerant to disconnections and node failures because of its mesh topology. Not only that, NerveNet can continuously provide connectivity services to applications such as mobile phones without the Internet, because each base station supports basic services like DHCP, DNS, SIP proxy, and mobility management (Chanakitkarnchok, et al., 2019).

In NerveNet architecture, databases are distributed among all the base stations over a mesh network. NerveNet databases handle data synchronization automatically, which stores routing informations and application data respectively. The synchronization process is performed by the NerveNet OS, external application servers can also join NerveNet mesh network by using a custom API to update data in the nearest database of a base station.

NerveNet database uses hearsay daemon to synchronize tables in different databases within the network, each node compares the hash of other databases with its own database. Hearsay daemon synchronizes MySQL database with insert and update queries only, delete actions will not be synchronized. When a NerveNet end node is disconnected or shut down, it will drop the data in synchronization tables. Once it is back in operation, it initiates the synchronization process by seeking the difference of table with other nodes, then updates the database with the latest data. If all nodes in the network go down at the same time, they are unable to relieve data using hearsay daemon synchronization after alive because existing databases are all empty.

2.7 MQTT

MQTT (MQ Telemetry Transport) is an OASIS standard messaging protocol that is designed to serve IoT applications (MQTT.org, 2022). It suits the best when encountering extremely lightweight messaging transport that is ideal for connecting remote devices with a minimal code footprint and network bandwidth. The devices that use MQTT protocol to send or receive data are called MQTT clients, they connect to a common MQTT broker to perform data exchange between each other. An example of MQTT bi-directional communication is shown in Figure 2.4.

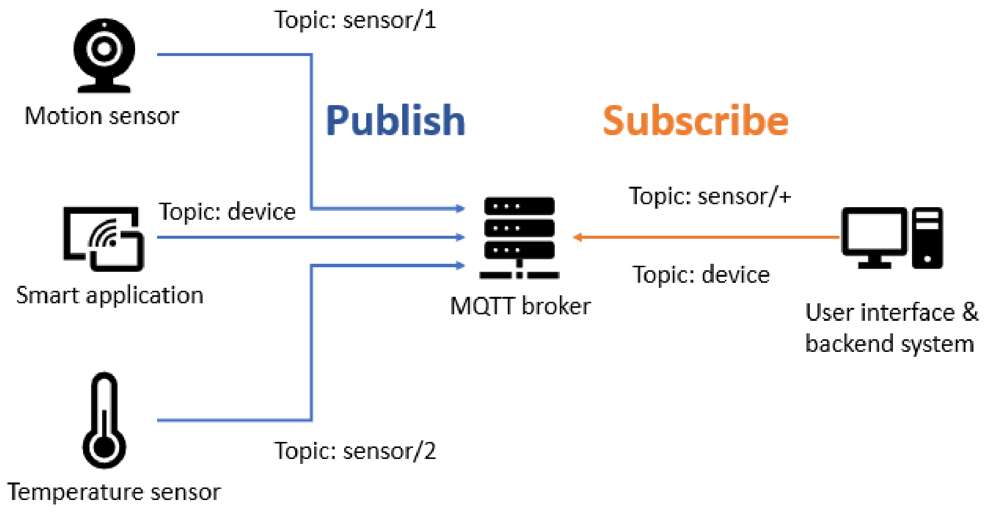


Figure 2.4: MQTT Data Transmission Fow.

Whenever the MQTT client wants to subscribe or publish data, it will need to connect to an MQTT broker using IP address. By default, MQTT listens to port 1883, message that is sent in this port is in plain text. If secure messaging is preferred, MQTT also uses port 8883 to transmit data over SSL/TLS. The broker is a framework that acts as the middleman between subscriber and publisher, it can reside in a dedicated device or any MQTT client. To filter out messaging sessions, the broker uses a topic, which is a UTF-8 encoded string, to identify messages for each client. The topic is defined by clients, subscribe/publish to the same topic allowing them to exchange data. MQTT topic uses "/" symbol as level delimiter, "+" as single-level wildcard, and "#" as multi-level wildcard. For example, a client subscribed to "house/#" topic could receive messages published from both "house/lamp" and "house/door/3" topics as well. The message published over MQTT topic is called payload, which is prepared as plain text, then transmitted as binary bytes. According to the MQTT specifications, the length at most for MQTT topic and payload is 65536 bytes and 268,435,456 bytes respectively. By default, the maximum size of each MQTT packet size is 256 MB, while the maximum buffer size in each subscription from either client or broker is 5000 messages. The MQTT packet's protocol data units are shown in Figure 2.5.

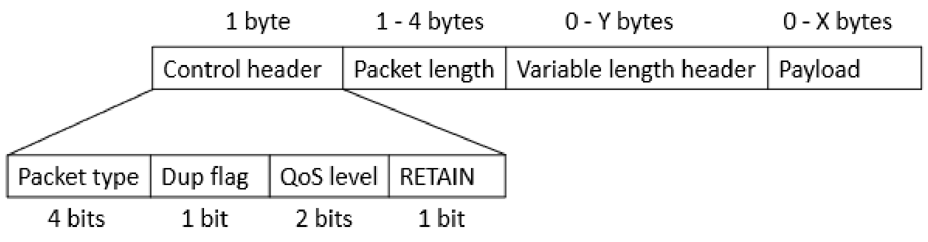


Figure 2.5: MQTT Packet PDU.

The control header and packet length are fixed headers, which are always present in MQTT packet. In contrast, variable length header and payload are not always present. The 4 bits packet type is the value of connection operation, such as 0011 represents PUBLISH packet, and 1000 represents PUBLISH packet. The DUP flag is "1" when the QoS level bit is greater than 0, it indicates that this is a duplicated message due to resent. QoS is the agreement between publisher and subscriber in terms of guarantee for delivery of a message. QoS level holds 2 bits of binary data, which indicates three different qualities of service. When QoS = 0, the message is sent at most once, and the subscriber will not acknowledge the publisher whether the message is received. If QoS = 1, the publisher will send the message at least once, it keeps the message and retransmits after a certain time until the subscriber reply a PUBACK acknowledgment packet. If QoS = 2, the message is sent exactly once, they perform a 4-way-handshake process to guarantee it is received by the subscriber. Due to the request and response flows, this QoS is the safest and also the slowest. The last PDU in the control header is called RETAIN. When the publisher sets RETAIN = 1 in MQTT packet, the broker will store the last message sent by the publisher, then forward it to each subscriber immediately when clients initiate subscribing to that topic. With this feature, the receiver client could pick up the retained message if it reconnects to the broker.

2.8 Docker

In this project, most of the applications are executed based on Docker container images, including the NerveNet applications and NerveDash monitoring system. Docker is software that allows developer to build and deploy applications easily. In terms of working environment, Docker act as an operating system for container images, this is similar to virtual machines that allow different guest OS images to run in one host (AWS Docker, 2022). Container image is a package that wraps up all necessary programming code modules, source files, and dependencies for an application to run (Docker Container, 2022). The container image can be run at any host that uses the same OS and installed with Docker, therefore it is very convenient for deployment. When deploying applications on-site, there are bundles of issues needed for troubleshooting, such as driver incompatibility, outdated software, configuration or setting issue, and many other possibilities. Therefore, the Docker container image is getting important nowadays because it allows applications to run without excessive installation, developer could also include every single dependency into the container image so that the application could be run without any problem related to the hosting machine’s software availability. Not only that, since the Docker handles container images the same way the virtual machine does, each container is isolated from the host machine’s environment and also between containers, only configured communication allows them to interact, such as using

TCP/UDP port numbers and UNIX socket. These characteristic makes Docker perfect to be adopted by cloud computing service provider, such as Amazon Web Services and Microsoft Azure.

2.9 Deployed NerveNet Monitoring System

This project is currently in the second stage, the main objective is to deploy a NerveNet testbed with LoRa and Wi-Fi connectivity with MQTT response. The first stage has been done by senior Lim Wei Sean, his service architecture is shown in Figure 2.4 below.

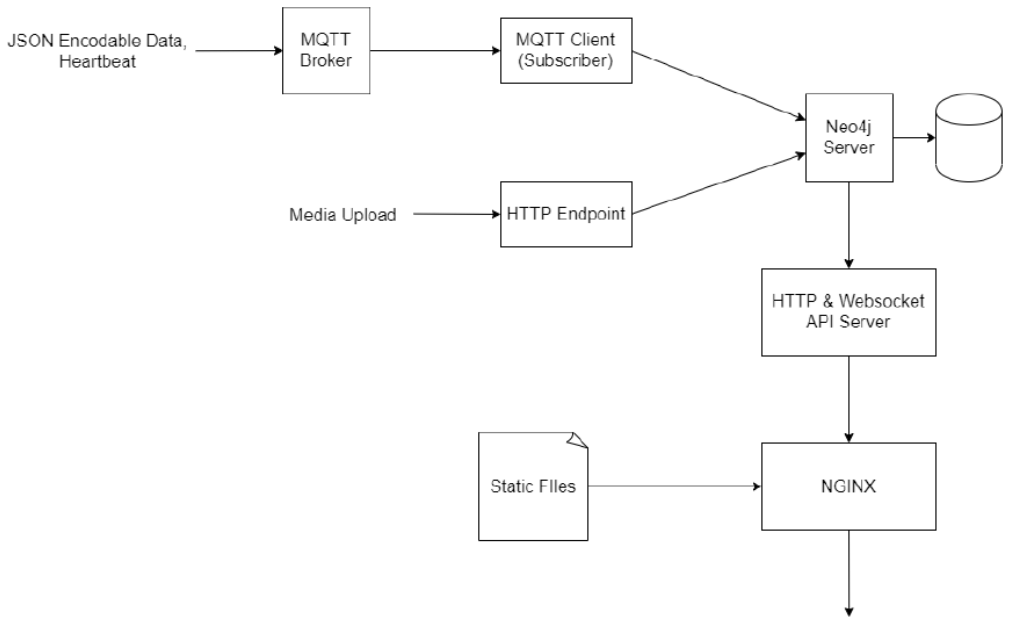


Figure 2.6: NerveNet System Deployment (Lim, 2021).

2.9.1 NerveNet Architecture

A simple three base station NerveNet testbed using Raspberry Pi is constructed, and its network connectivity is shown in Figure 2.5 below.

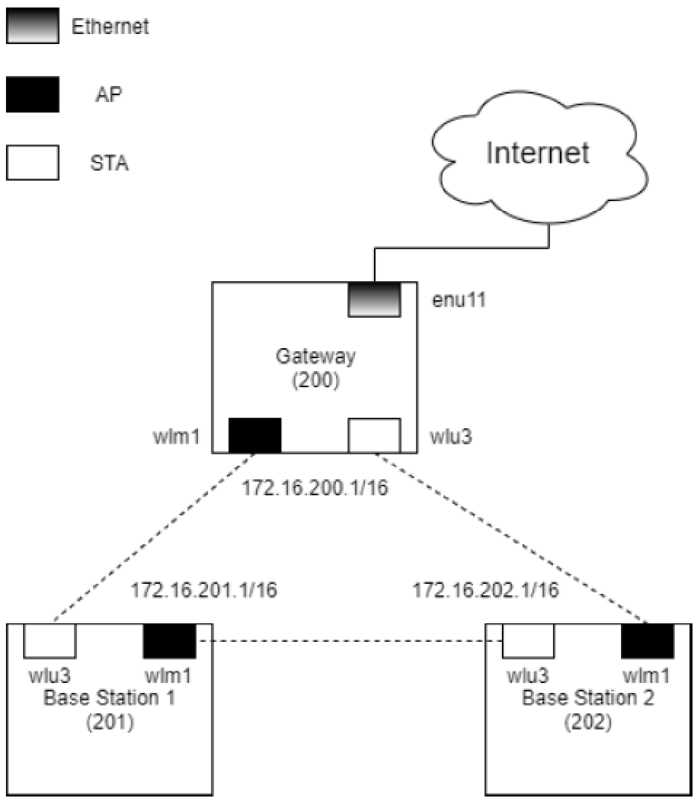


Figure 2.7: NerveNet Base Stations (Lim, 2021).

This network is designed to have one gateway node and two base station nodes. Using the ERB configuration of NerveNet, ERB link is established (wireless link) between two nodes. Two endpoints of each link are access point and client wireless interface respectively. In ERB wireless access points can only accept one connection from a specific client. For each node, wlm1 is the access point interface while wlu3 is the client interface, enu11 interface in the gateway node is the Ethernet port. By default, NerveNet is configured within the 172.16.n.0/16 network, where n is the node id defined during the installation of the network.

2.9.2 NerveNet Web Application: NerveDash

NerveDash is the name of a web application designed by Sean. The main components to support NerveDash are Neo4j (cloud server), MQTT service, HTTP server, REST API, Websocket API, and Nginx server.

When NerveNet gateway sends a message, it will first be handled by MQTT client for JSON encodable data. If the data is media (image, video), it will be sent to HTTP server instead of MQTT broker. When the media storage has exceeded its limit, the oldest stored media will be replaced by the latest media data. Both MQTT and HTTP service point towards Neo4j server, a graphical-based NoSQL database server. By default, MQTT broker will set a count-down timer to receive a “heartbeat” message from NerveNet gateway (publisher). If the message is not received in 20 seconds interval, an inactive message will be sent to Neo4j (subscriber) to record that the respective node is down until a new “heartbeat” message is received.

The data in Neo4j can then be retrieved via a RESTful API or a WebSocket API. The RESTful API is used for simple text retrieval, while WebSocket API is used for file streaming. Finally, the Nginx static file hosting feature is used to serve the frontend static files of the web application. The Nginx server can also be configured to provide load balancing for all HTTP endpoints if needed.

2.10 Summary

To design a disaster resilient network, mesh network is the best topology to achieve fault-tolerance during emergency situations. The basic requirement of a resilient network, such as distributed processing, network storage, and communication service, can be met using NerveNet framework, it supports wireless mesh connection and data synchronization to fulfill those requirements. Lastly, LoRaWAN device is suitable for NerveNet nodes as it supports wide range of connectivity coverage using mesh topology. LoRa-based end devices also have the advantage of low power consumption to support long-lasting such as years with just a battery.

CHAPTER 3

METHODOLOGY AND WORK PLAN

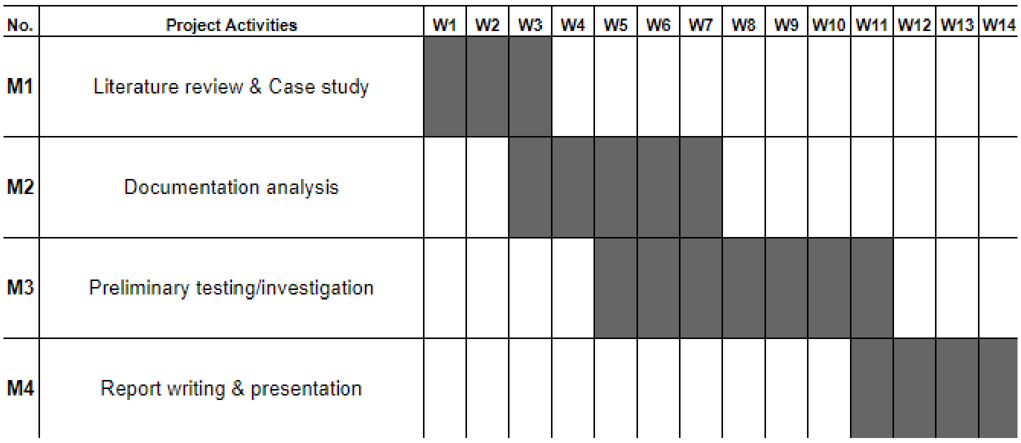
3.1 Introduction

This project aims to develop a disaster-resilient mesh network with NerveNet LoRa connection NerveNet Wi-Fi connection with data synchronization in both x86 and armhf machines. To build the planned prototype, case studies and literature reviews on NerveNet, LoRa, Raspberry Pi, MQTT, HTTP, and database are necessary.

3.2 Work Plan

In part one of Final Year Project, the main task is to study and acknowledge technics required to build the entire system. For the literature review and case study, the targeted topics are the concept of resilient network, mesh network, NerveNet, LoRa communication, and basic Linux commands. Since this project's first stage has been carried out by Lim, knowledge transfer and system replication are also needed, and important notes and procedures regarding system deployment are made into documentation.

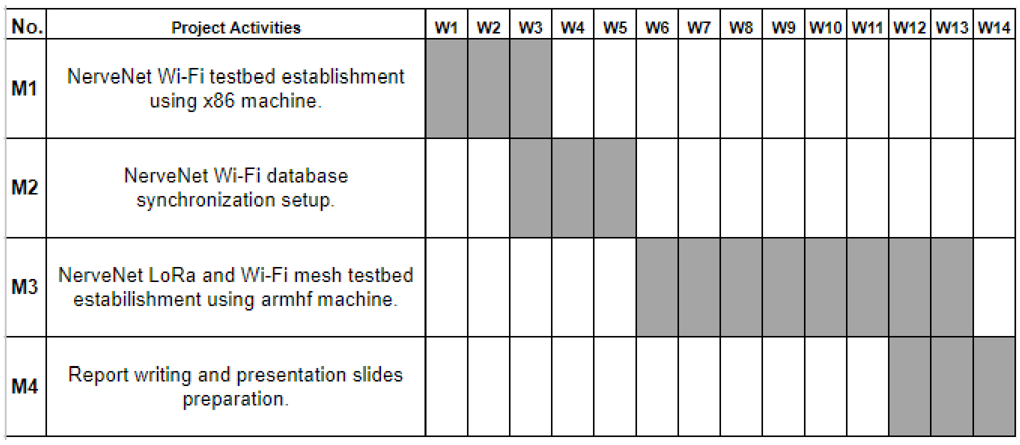
Next, the preparation and testing of replicates NerveNet testbed design by Lim also has been done, such as NerveNet installation, mesh links establishment, MQTT & HTTP responses, and the server host application. At the end, Final Year Project progress report and presentation are also made. The project activities are shown in the Gantt chart in Figure 3.1.



No.	Project Activities	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
M1	Literature review & Case study	▨	▨	▨											
M2	Documentation analysis			▨	▨	▨	▨	▨							
M3	Preliminary testing/investigation					▨	▨	▨	▨	▨	▨	▨			
M4	Report writing & presentation											▨	▨	▨	▨

Figure 3.1: Gantt Chart of FYP Part 1.

In FYP part 2, the scope is to set up the NerveNet Wi-Fi and LoRa mesh testbed using x86 and armhf machines. The feature of NerveNet database synchronization via Wi-Fi mesh is also implemented. After all testbed setup is done, the performance tests are carried to identify the reliability of data transmission among NerveNet nodes. Lastly, several Python programs are written to ease of LoRa MQTT data exchange and read/write database. The Gantt chart of FYP part 2 is shown in Figure 3.2.



No.	Project Activities	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
M1	NerveNet Wi-Fi testbed establishment using x86 machine.	▨	▨	▨											
M2	NerveNet Wi-Fi database synchronization setup.			▨	▨	▨									
M3	NerveNet LoRa and Wi-Fi mesh testbed estabilishment using armhf machine.						▨	▨	▨	▨	▨	▨	▨	▨	
M4	Report writing and presentation slides preparation.												▨	▨	▨

Figure 3.2: Gantt-chart of FYP Part 2.

3.3 Eclipse Paho – MQTT and Python

MQTT is a lightweight IoT messaging protocol based on the publisher and subscriber model, it provides reliable messaging service and real-time data transfer using TCP/IP protocols. It is suitable for NerveNet devices to forward disaster application data over destination.

Paho is a MQTT implementation developed by Eclipse. In the first stage of the project done by Lim, the MQTT subscriber is implemented at the hosting server. MQTT publisher/subscriber program at each NerveNet LoRa node is constructed using Paho Python Client to send/receive LoRa packets. Moreover, a Python program to read/write NerveNet database is also constructed. At the end, the program also sends all collected disaster information to NerveDash via MQTT and HTTP POST request.

3.4 Architecture of Planned NerveNet LoRa Network

The hardware chosen for deploying NerveNet LoRa network is listed as below.

i. 2 units of Raspberry Pi 3.

ii. 1 unit of Raspberry Pi 4.

iii. 3 units of Intel NUC Mini PC

iv. 6 units of TP-Link AC1300 Archer T4U High Gain Wireless MU-MIMO USB Adapter.

v. 4 units of RFlink LoRa RM-92A.

vi. 2 units of Clealink LoRa Pi Hat.

vii. 2 units of Pi Hat GPS.

viii. 1 unit of Global SAT BU-353S4 USB GPS

ix. 1 unit of DFRobot TEL0138 USB GPS

x. 2 units of RFlink USB-LoRa

xi. 4 units of USB Lora Antenna.

The planned network structure is designed as Figure 3.3 below.

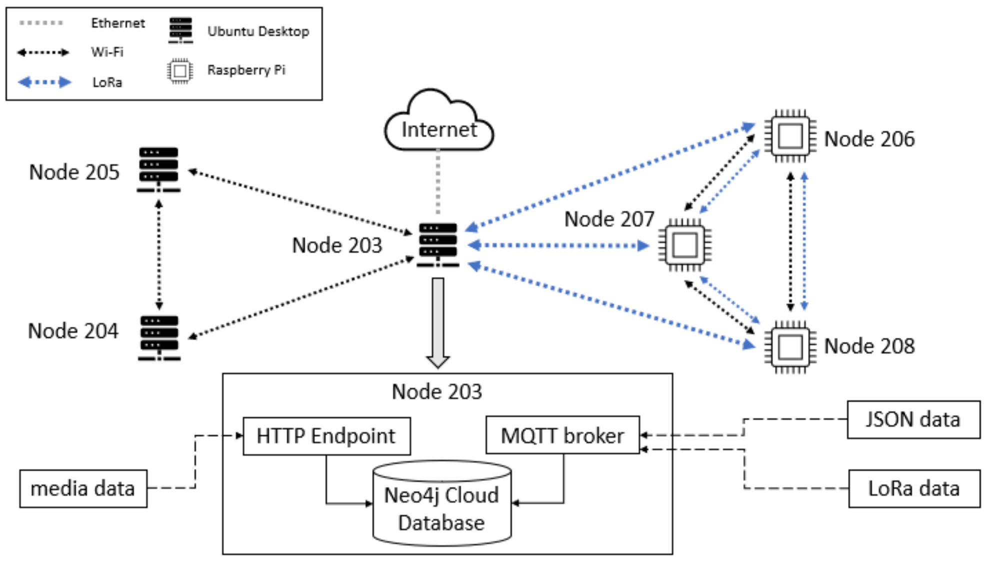


Figure 3.3: NerveNet Testbed using Wi-Fi and LoRa.

3.5 NerveNet Wireless Mesh Network

Usually, a mesh network is simply adding a redundant connection for each device within the network topology, then the device will look up for an alternative pathway to reach its destination if its primary peer is down. NerveNet

Wi-Fi mesh network framework not only provides the function to look up every single mesh node in the network, it also add-in database synchronization to share common data within the mesh network. The lookup feature is built by using a service daemon called PTMGR (Path Tree Management Generation), it needs to be installed in the essential node within the mesh network. PTMGR continuously seeks for peers' network status to identify if any node is down or new node has joined the network. If the connection between nodes is steadily maintained by PTMGR, the nodes could directly connect or access to each other and perform NerveNet SQL database synchronization. The nodes will compare the data rows within each other to update with the latest data.

Not only Wi-Fi mesh, NerveNet also supports LoRa mesh network with the use of specific LoRa equipment. NerveNet LoRa uses a frequency band of 920 MHz for all LoRa node, if each node transmits data at the same time, the signal radio waves will collide, and the listener will receive a corrupted signal. Hence, NerveNet LoRa uses a time slot to overcome this issue. Each node synchronizes the time from the GPS receiver to other nodes, then the node will transmit LoRa data within the period of the preset time slot. For example, 10 seconds are divided into five slots to form a cycle, the time slots are allocated to five nodes, thus each node will transmit LoRa signal during its time slot only. With properly configured time slots and channels to reduce disturbance, NerveNet LoRa nodes are possible to communicate at the speed of 100 Bytes per second over several kilometers distance with a power consumption of only tens of milliWatt.

3.5.1 Wi-Fi Mesh Network

To establish NerveNet Wi-Fi mesh connection as network testbed architecture which is shown in Figure 3.2, each device is the first set up with NerveNet IP address, AP interface and WPA client interface, where x86 machines and Raspberry Pi OS uses the same way configure. These interfaces are then linked to NerveNet VLAN interface to use the ERB tunnel link, which is used by the PTMGR to enable each node to communicate. Then, the Wi-Fi connection is configured using the Hostapd and WPA supplicant services provided by NerveNet Docker container instead of those original packages with Linux

Ubuntu, this is to avoid incompatibility with the packages' environment. The Network Manager service is also masked to avoid interference, because it tends to bring up USB Wi-Fi interface as client mode from AP mode. After these settings are configured, all nodes in x86 NerveNet Wi-Fi domain are now able to ping each other and synchronize their NerveNet database. This is also the same as in armhf NerveNet Wi-Fi domain.

3.5.2 NerveNet LoRa Mesh Network

NerveNet LoRa mesh configuration is set based on using RFlink-RM92A as the LoRa module, the parameters that could be altered are radio frequency channel, bandwidth, and spreading factor. In this testbed, all LoRa node RF-channel of 41, RF-bandwidth of 500 kHz, and spreading factor of SF12. While the time slot cycle in this testbed is set to be four slots per minute, which means each LoRa node (BS203, BS206, BS207, and BS208) is allocated with a 12 seconds duration to transmit LoRa signal at each minute. To communicate with plain text data, NerveNet LoRa node uses MQTT service to buffer published data, then sends out the LoRa data within the allocated time slot with its best effort. The LoRa data will remain in the MQTT buffer and waits for the next cycle if the attempt to transmit fails. However, NerveNet LoRa MQTT communication uses QoS level zero, LoRa packet loss is still possible.

3.5.2.1 LoRa IoT Pi HAT

For Raspberry Pi 3, LoRa IoT Pi HAT is installed to facilitate RFlink-RM92A LoRa module and GPS receiver. By default, Raspberry OS identify Pi HAT as serial port, therefore it is enabled in raspi-config. After the serial port is enabled, the OS may occupy the serial device as serial login connection interface, therefore serial login is disabled in raspi-config. Then, the serial device name and GPS baud rate (9600 bits per second) are added in NerveNet LoRa configuration.

3.5.2.2 USB-LoRa and USB-GPS Receiver

Since Raspberry Pi 4 and Intel NUC mini PC are incompatible with LoRa HAT, USB-GPS receiver and USB-LoRa adapter are used. The Global SAT GPS with baud rate of 4800 bps is equipped in Raspberry Pi 4, while DFRobot TEL0138 GPS with baud rate of 9600 bps is equipped in Intel NUC mini PC. Since both equipments are USB serial devices, the name of device in terms of "/dev/tty" might differ every time the machine boot or equipment is replugged in. Thus, symbolic links in terms of product ID and vendor ID are created in Udev rules, so that the name of "/dev/ttyUSB-gps" is always referring to the GPS receiver, while "/dev/ttyUSB-RF" is always referring to the USB-LoRa adapter. As the same in discussed above, the device name and GPS baud rate are then added in NerveNet LoRa module.

3.6 Image Transfer using NerveNet LoRa Mesh

Since NerveNet LoRa mesh MQTT uses QoS level zero, the delivery of LoRa messages is not guaranteed. The higher the payload size or the number of messages, the higher the rate of packet loss. Since the maximum payload size for NerveNet LoRa MQTT packet is 152 Bytes, it is impossible to send an image in a single packet. To overcome this issue, the image file is first encoded to base64 string, then it is fragmented to 80 Bytes per row, with a row index and comma symbol at the leading position of each row. Then, a Python program is made to handle the MQTT LoRa message with acknowledgment. The algorithm is illustrated in Figure 3.4.

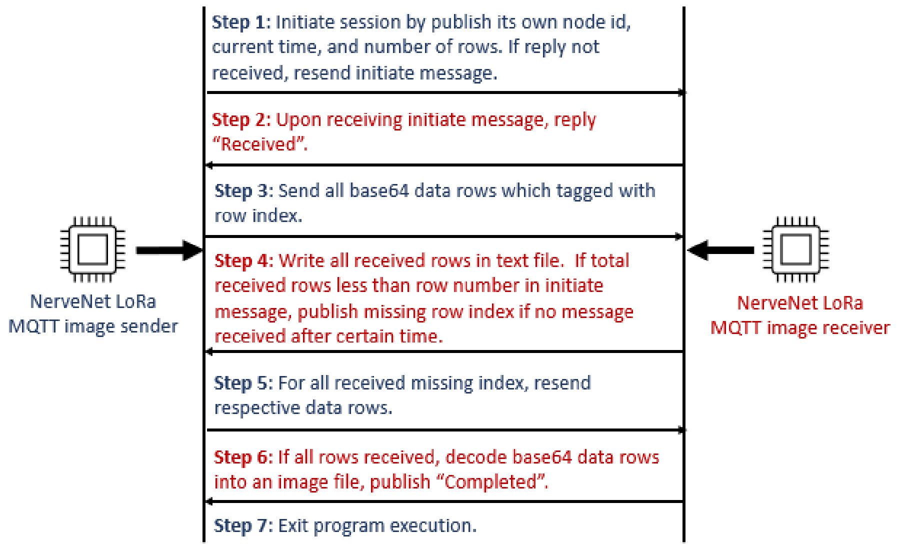

Step 1: Initiate session by publish its own node id, current time, and number of rows. If reply not received, resend initiate message.
Step 2: Upon receiving initiate message, reply "Received".
Step 3: Send all base64 data rows which tagged with row index.
Step 4: Write all received rows in text file. If total received rows less than row number in initiate message, publish missing row index if no message received after certain time.
Step 5: For all received missing index, resend respective data rows.
Step 6: If all rows received, decode base64 data rows into an image file, publish "Completed".
Step 7: Exit program execution.
NerveNet LoRa MQTT image sender
NerveNet LoRa MQTT image receiver

Figure 3.4: Algorithm to Send and Received Fragmented Image using NerveNet LoRa MQTT.

3.7 Performance Evaluation

After the NerveNet testbed is constructed, the network performance within x86 and armhf Wi-Fi domain is benchmarked in terms of latency, throughput, jitter, and image synchronization. Moreover, the average time taken to receive a LoRa message and image is also evaluated. In this project, the performance is evaluated where each NerveNet node is separated 10 cm apart.

3.7.1 Latency

The network latency within NerveNet Wi-Fi domain is benchmarked using the ping command in the terminal. Ping uses the Internet Control Message protocol, by pinging peer's NerveNet IP address, the minimum, maximum, and average latency in milliseconds is returned. There are two conditions to evaluate the latency, the first condition is when one out of three base stations is down, then the network latency in P2P connection is recorded. The second condition is when all three base stations are available, the network latency is evaluated based on the mesh link.

3.7.2 TCP/UDP Throughput

Iperf3 is used to evaluate the TCP and UDP throughput in NerveNet Wi-Fi domain. For TCP throughput, 100 packets are sent to the receiver, retransmission of lost packets will increase the time taken to receive all packets. The total data size received at the receiver side divided by the time taken is so-called throughput, which is in milliseconds. For UDP throughput, the sender bandwidth is set at 50 MBps, duration to send dummy data is 10 seconds by default, which means 500 MB of data to send in total. Since UDP packet receiver will not acknowledge the sender if the packet is lost, the received data size divided by 10 seconds is the UDP throughput. The throughput in NerveNet x86 and armhf Wi-Fi domain based on P2P and mesh link are both evaluated.

3.7.3 Jitter

In real-life applications, there will be a certain delay between continuous packets caused by slow network connections, congestions, and queuing. The delay between packets is called jitter. High jitter will cause reduced application performance and user experience. When measuring UDP throughput using Iperf3 tool, the jitter in milliseconds is also returned. The jitter in NerveNet x86 and armhf Wi-Fi domain based on P2P and mesh link are both plotted.

3.7.4 NerveNet Database Image Synchronization

NerveNet database supports updating images as data rows, the image file can also be synchronized to other nodes within NerveNet domain. The average time taken to synchronize different-sized images is recorded. Since different node as updater may lead to a different result, the average time taken to synchronize image file from all nodes are measured. The image size is tabulated in Table 3.1.

Table 3.1: Image Resolution and Size for Database Synchronization Test.

	Image resolution	Image size
1	960 x 540	264.8 kB
2	1920 x 1080	909.7 kB
3	3554 x 1999	2.5 MB
4	9600 x 6800	10.9 MB

3.7.5 LoRa Message

To measure the efficiency of NerveNet LoRa data transmission, the LoRa message with 30 Bytes and 90 Bytes payload size is sent 10, 20, 40, 60 times at once respectively, then the number LoRa packets received, and packets lost in each case is recorded. Finally, the time taken to receive the different-sized image via NerveNet LoRa transmission is also recorded.

3.8 Summary

Using NerveNet Wi-Fi mesh network, mesh links between Linux devices enable them to communicate with each other and share common database. Essential data could be shared among each node within NerveNet Wi-Fi domain in just seconds. However, data transmission over Wi-Fi is limited to hundreds of meters, this could be expensive to deploy sufficient nodes that collect data in a wide area. NerveNet LoRa mesh is therefore applied in IoT device, which is Raspberry Pi in this project, to send or receive small-sized data over kilometers.

The data transmission performance of NerveNet Wi-Fi and LoRa mesh are discussed in the next chapter.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

The NerveNet Wi-Fi mesh network performance within x86 (BS203, BS204, BS205) and armhf (BS206, BS207, BS208) are evaluated. Figure 4.1 and Figure 4.2 show the Wi-Fi links between each device.

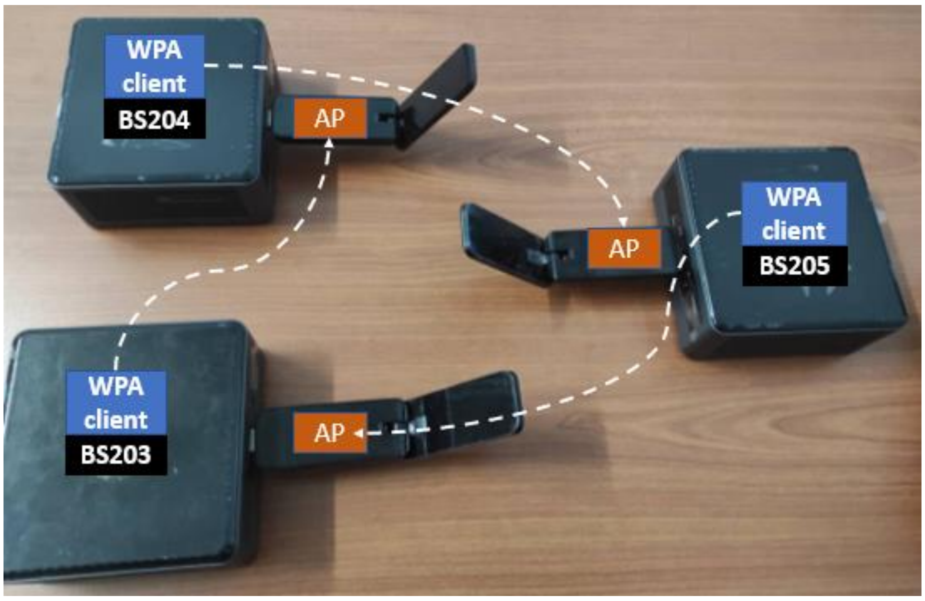


Figure 4.1: NerveNet Triangular Wi-Fi Mesh using x86 Intel NUC Mini PC.

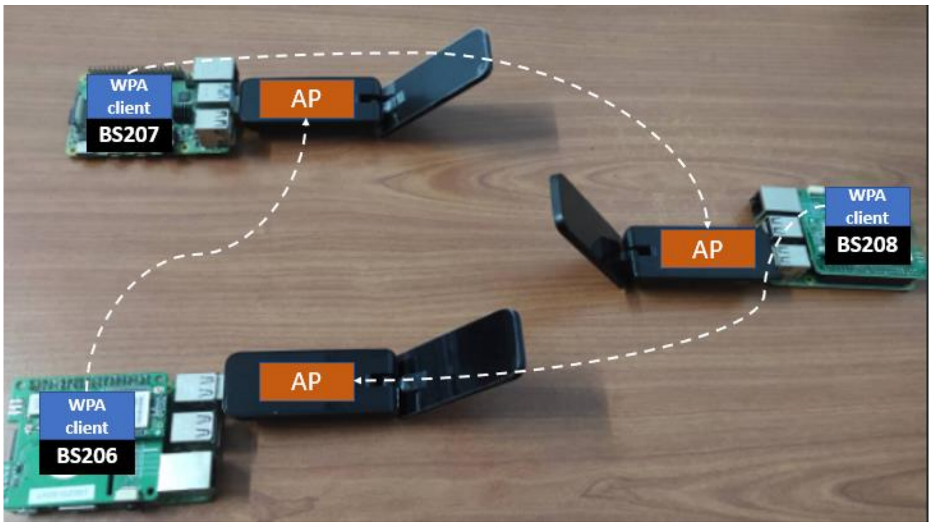


Figure 4.2: NerveNet Triangular Wi-Fi Mesh using Armhf Raspberry Pi.

In this NerveNet LoRa mesh testbed, any device within the LoRa network could perform LoRa MQTT data exchange with each other. The NerveNet LoRa mesh MQTT messaging performance is carried out between BS203 and BS207. Additionally, image file transfer with fragmented base64 string using NerveNet LoRa is also tested to identify if it is suitable for image file transfer. Figure 4.3 shows the LoRa link between the devices.

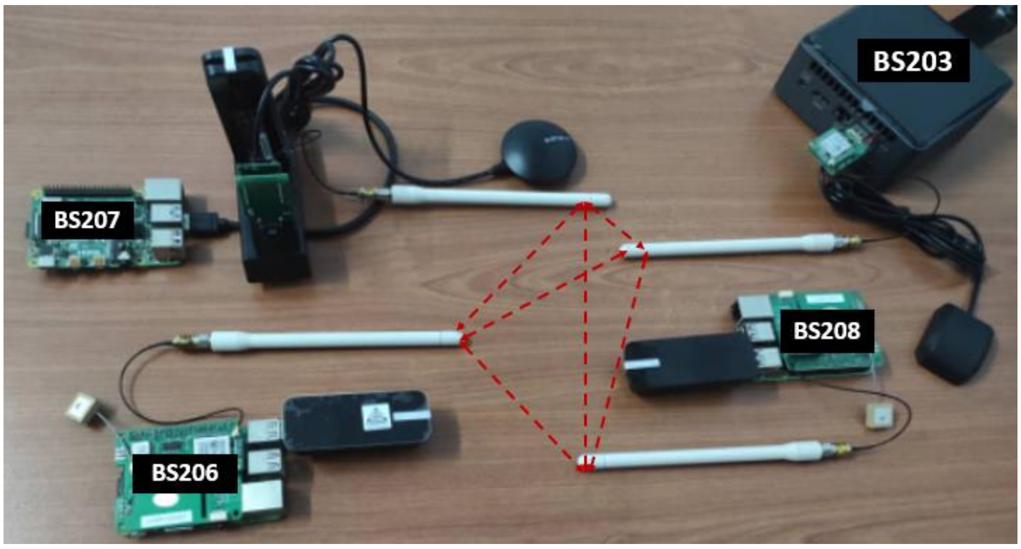


Figure 4.3: NerveNet LoRa Mesh Network using one x86 Intel NUC Mini PC and three Armhf Raspberry Pi.

4.2 NerveNet x86 Wi-Fi Mesh Benchmark

There are two methods to evaluate TCP/UDP throughput, latency, and jitter. The first method is shutdown one out of three NerveNet nodes within the Wi-Fi domain to record network performance in P2P link. The second method is turn on all three NerveNet nodes within the Wi-Fi domain to record network performance in mesh-link. The result of TCP and UDP throughput using Iperf3 tool are recorded and shown in Figure 4.4 and Figure 4.5.

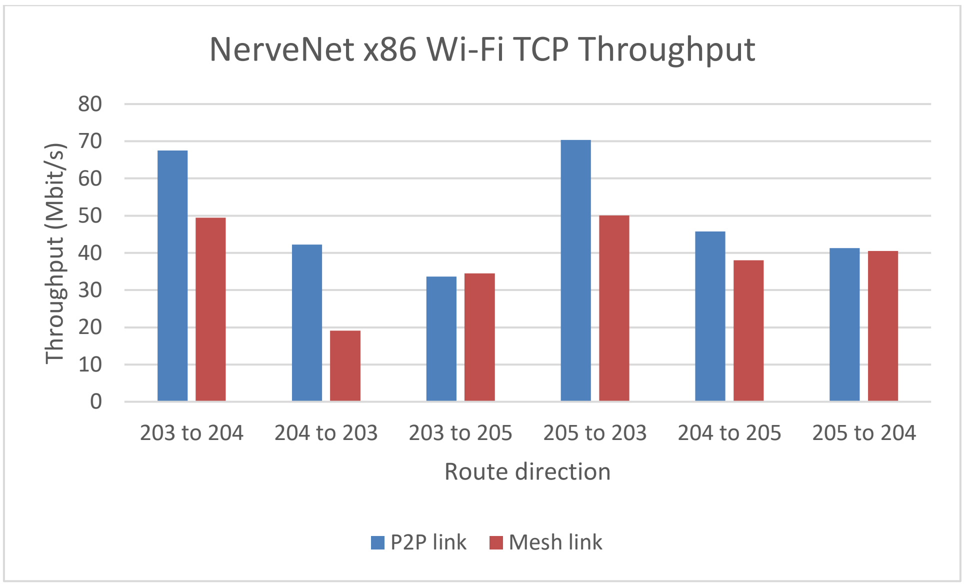


Figure 4.4: NerveNet x86 Wi-Fi TCP Throughput.

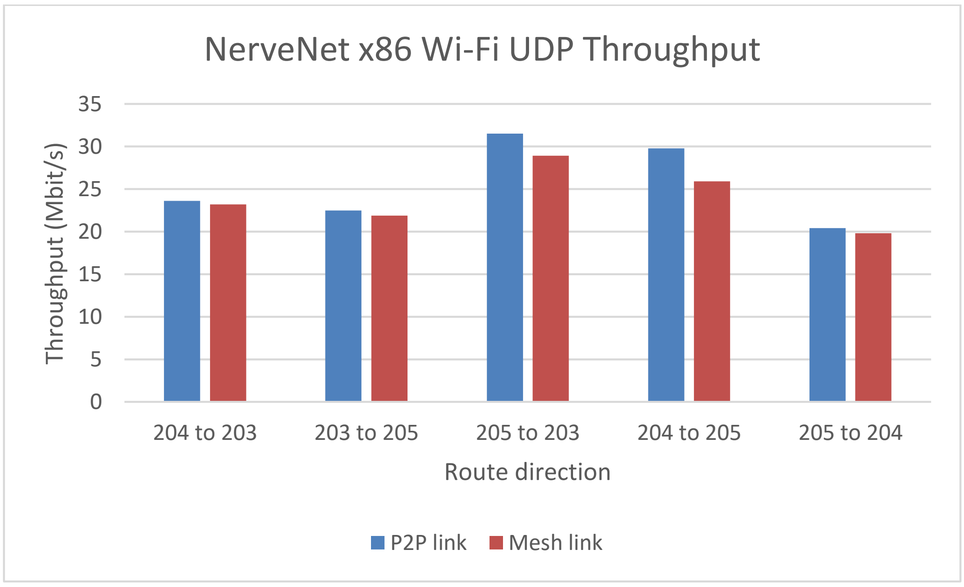


Figure 4.5: NerveNet x86 Wi-Fi UDP Throughput.

Based on Figure 4.4 and Figure 4.5, P2P link has generally higher TCP and UDP throughput as compared with mesh-link, this is because the sender does not need to calculate a pathway to transfer the data. Also, when the route direction is from Wi-Fi client interface to Wi-Fi AP interface of peer device, the TCP throughput is also higher as compared with the reverse ordered route direction.

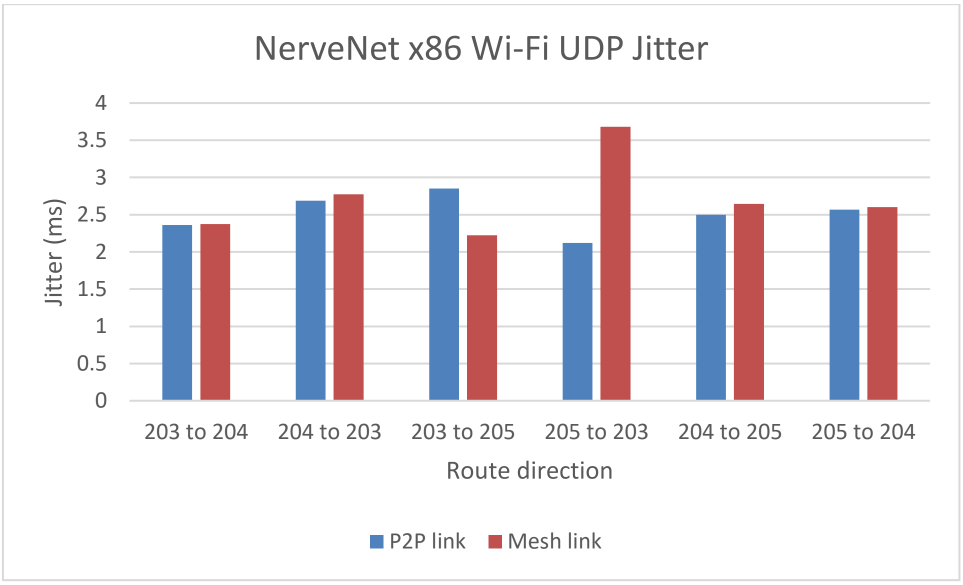


Figure 4.6: NerveNet x86 Wi-Fi UDP Jitter.

As referring to Figure 4.6, the jitter of NerveNet Wi-Fi within x86 domain in the cases of P2P and mesh links are more or less similar. The difference between the highest and lowest jitter is less than two milliseconds. According to Khalifeh, Gholamhosseinian, and Hajibagher, the QoS requirements of jitter for video conferencing is less than 30 ms. Therefore, NerveNet Wi-Fi within triangular x86 nodes has good fundamentals to handle applications that require low jitter, such as providing VoIP services.

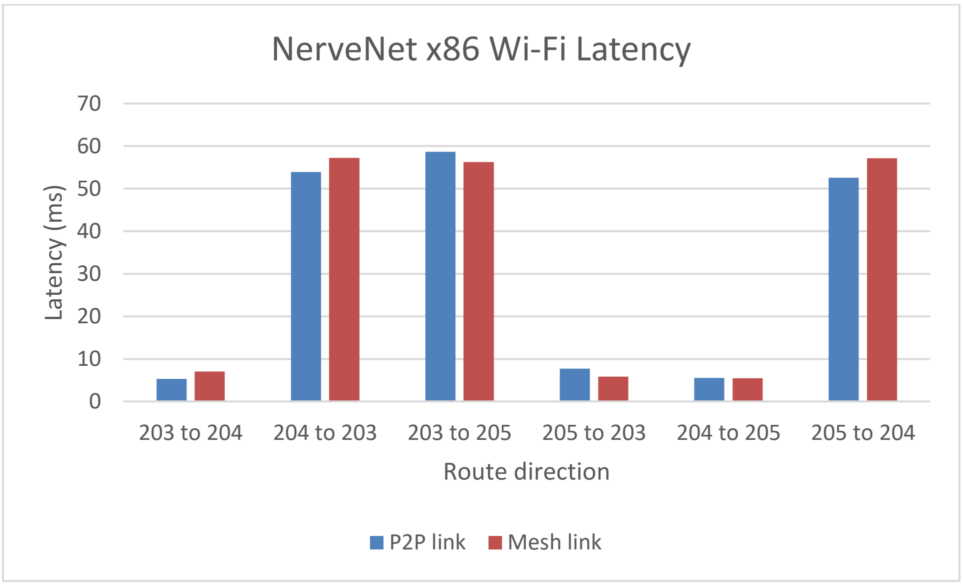


Figure 4.7: NerveNet x86 Wi-Fi Latency.

According to the Figure 4.7, there is no big difference in terms of P2P and mesh links within NerveNet x86 Wi-Fi domain. However, the latency is less than 10 ms when the route direction is from Wi-Fi client interface to Wi-Fi AP interface, while the latency is five times greater if the route direction is reversed. This can be explained by NerveNet Wi-Fi in x86 machines having a lower route cost when the target's AP interface is the next hop of its own client interface. Therefore, even if the target is just located at the next hop of its AP interface, the sender would still seek for target from its client interface's next hop, causing the packet return time to increase.

4.3 NerveNet Armhf Wi-Fi Mesh Benchmark

As the same with NerveNet x86 Wi-Fi domain, P2P and mesh links are also used to evaluate the TCP/UDP throughput, jitter, and latency in NerveNet armhf Wi-Fi domain. The result of TCP and UDP throughput using Iperf3 tool are recorded and shown in Figure 4.8 and Figure 4.9.

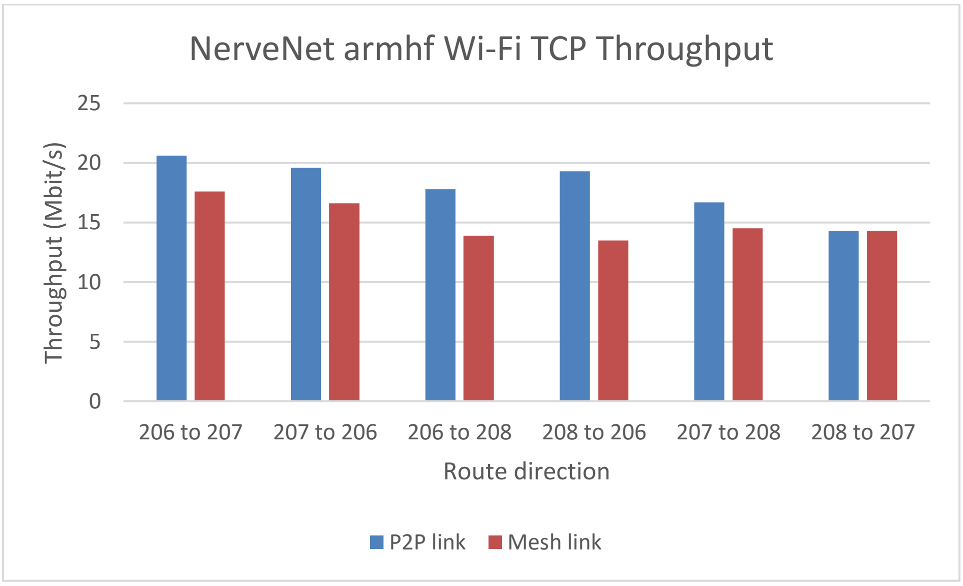


Figure 4.8: NerveNet Armhf Wi-Fi TCP Throughput.

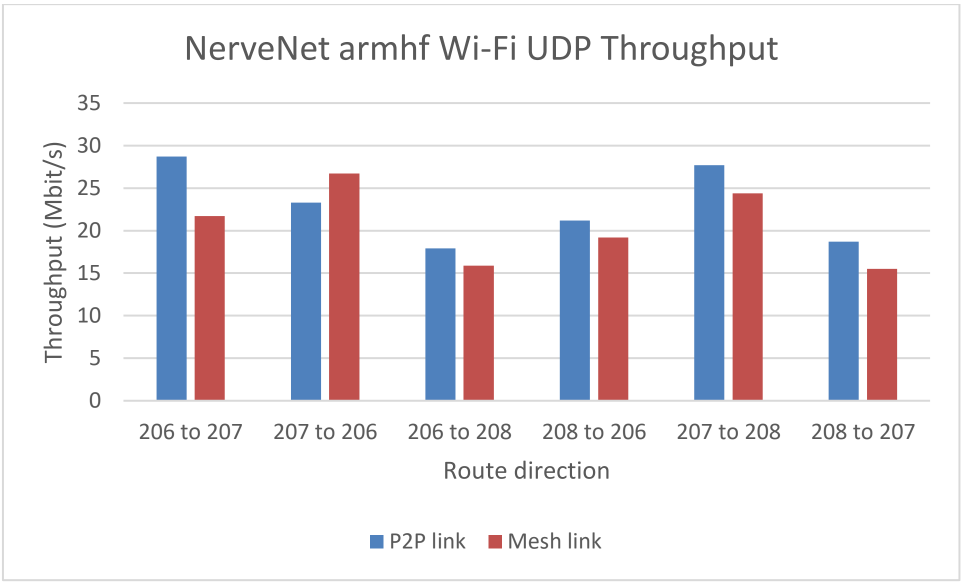


Figure 4.9: NerveNet Armhf Wi-Fi UDP Throughput.

According to Figure 4.8 and Figure 4.9, P2P link generally has a higher throughput as compared with mesh link. Unlike x86 machines, the relationship between throughput and route direction in NerveNet armhf Wi-Fi domain is not obvious. The highest throughput (BS207 to BS206) and lowest throughput (BS206 to BS208) via mesh link are both obtained when the target is located at the next hop of the sender's AP interface.

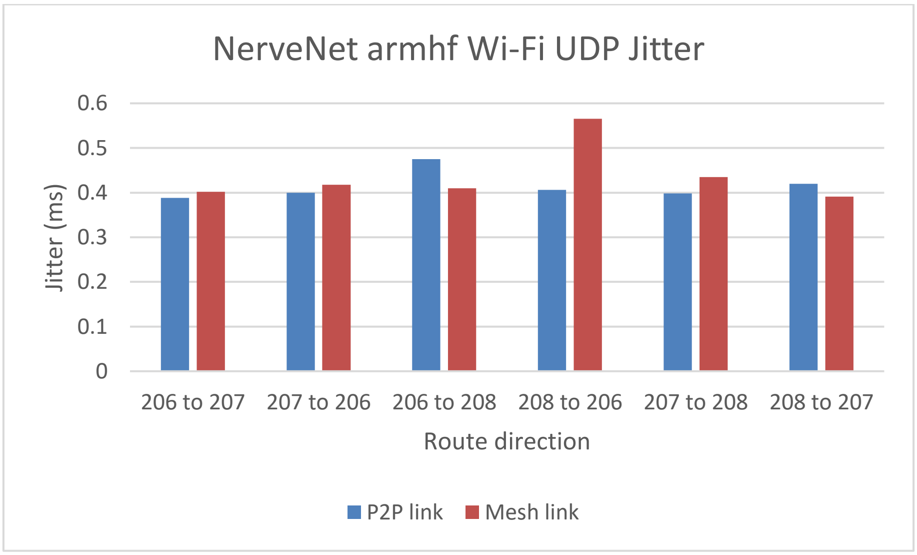


Figure 4.10: NerveNet Armhf Wi-Fi UDP Jitter.

Figure 4.10 shows the average jitter of P2P and mesh links within NerveNet armhf Wi-Fi domain. The variance of jitter at each link and route directions is as tiny as ignorable. However, even the highest jitter is just between 0.5 to 0.6 ms, which is at least three times lesser than the jitter in NerveNet x86 Wi-Fi domain.

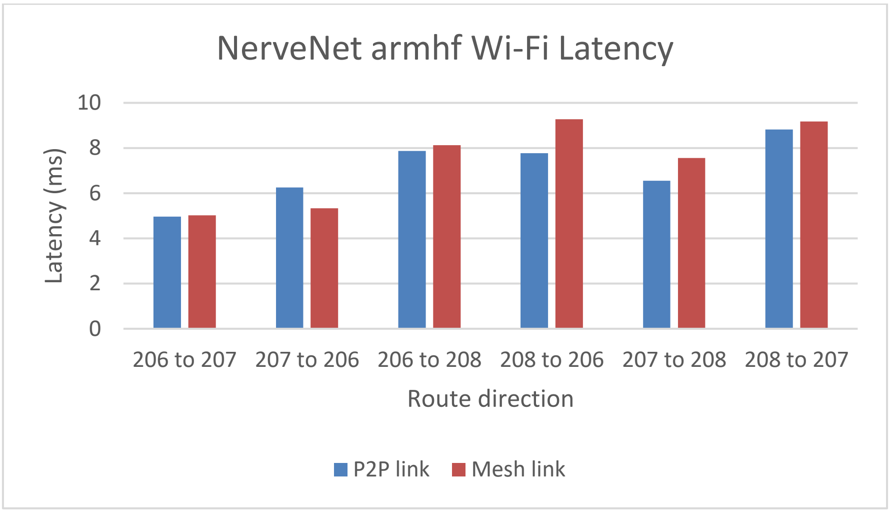


Figure 4.11: NerveNet Armhf Wi-Fi Latency.

From Figure 4.11, it can be concluded that P2P link generally has lower latency as compared with mesh link in NerveNet armhf Wi-Fi domain. The route direction from the sender's client interface to AP also has lower latency as compared with reversed route direction. However, the overall latency is still low, the difference is also not as big as of NerveNet x86 Wi-Fi domain.

4.4 Image Synchronization using NerveNet x86 Wi-Fi Mesh

To evaluate the time taken from an image file to be synchronized in all NerveNet x86 base station databases via Wi-Fi mesh, the cases of all nodes as image senders are tested. The corresponding time taken for the other two peers to receive the synchronized image file is recorded in Figure 4.12.

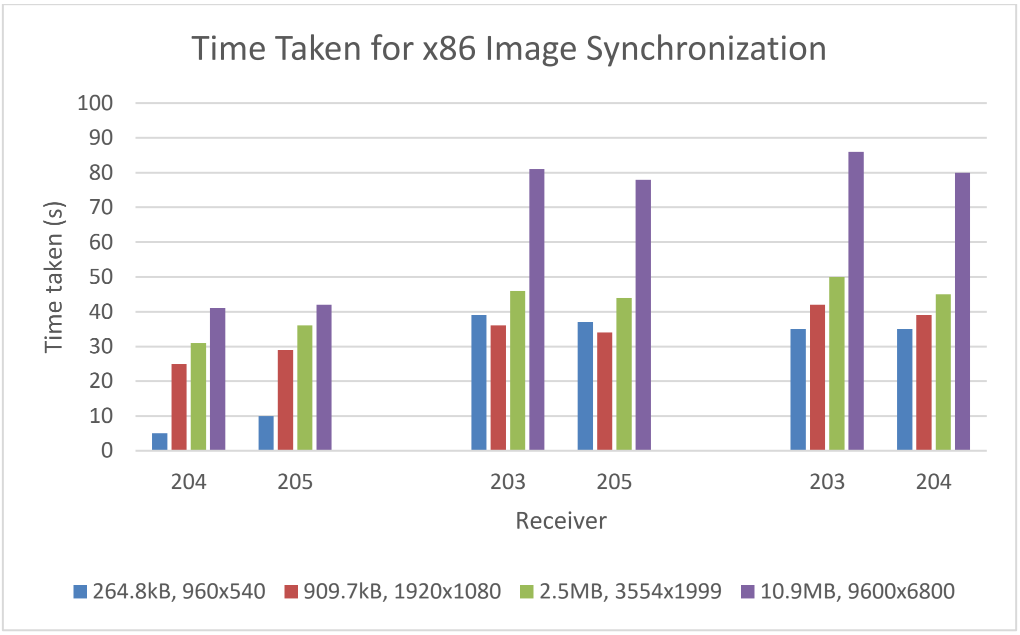


Figure 4.12: Time Taken for x86 Image Synchronization.

From Figure 4.12, it is clearly stated that as the image file size increased, the time taken for the peers to receive the synchronized file also increased. However, the time taken is not linear. For example, Node 203 takes 50 seconds to receive a 2.5 MB image file from BS205. However, it only takes 86 seconds to receive a 10.9 MB image file, which is at least four times greater than the 2.5 MB image file. The figure also shows that the image sent by BS203 takes least time to be synchronized in the peers' database, this could be due to the PTMGR daemon running at BS203, therefore it takes the least time to calculate Wi-Fi pathways.

4.5 Image Synchronization using NerveNet Armhf Wi-Fi Mesh

To evaluate the time taken from an image file to be synchronized in all NerveNet armhf nodes' database via Wi-Fi mesh, the test cases are similar to cases in NerveNet x86 Wi-Fi mesh. The result is shown in Figure 4.13.

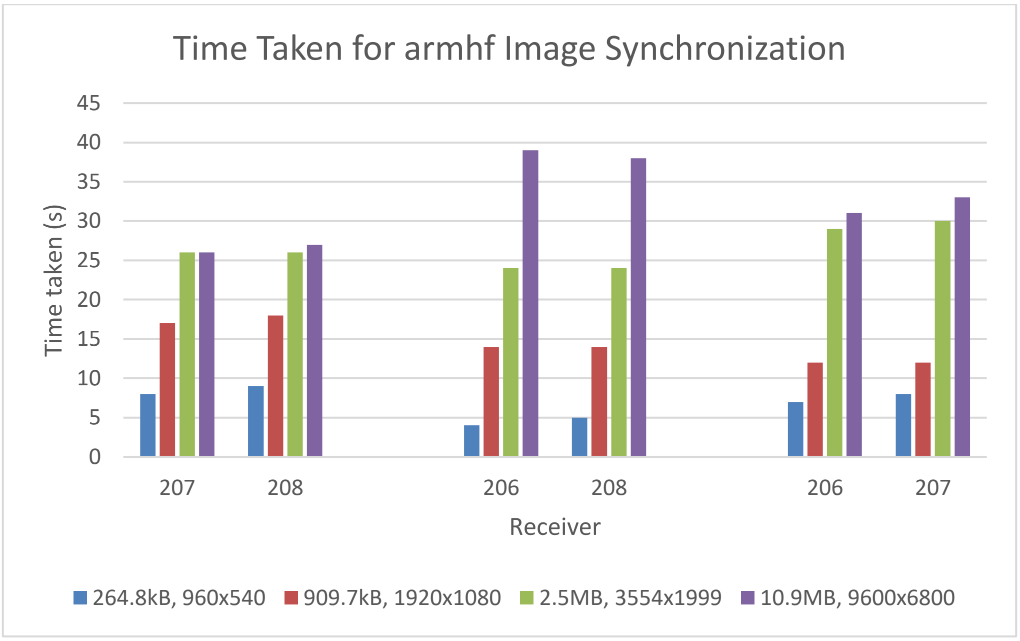


Figure 4.13: Time Taken for Armhf Image Synchronization.

From Figure 4.13, the characteristics of time taken to receive a synchronized image are also similar to the result in NerveNet x86 Wi-Fi mesh. Time taken to synchronize an image file is not linear to file size, while the image file sent from the node running PTMGR daemon, which is BS206, generally takes the least time to synchronize an image file.

4.6 NerveNet LoRa Mesh Messaging Performance

Since NerveNet LoRa mesh MQTT uses QoS level zero, the percentage of lost LoRa packets is interested. To test the NerveNet LoRa mesh MQTT messaging performance, the number of packets lost with MQTT payload size of 30 Bytes and 90 Bytes are recorded accordingly. Not only that, the number of LoRa packets sent at once could affect the ratio of lost packets, hence the number of LoRa messages published at once is varied at 10, 20, 40, and 60 messages. After the LoRa MQTT subscriber has not received any message for 20 minutes, the remaining LoRa packets are considered lost. The test is carried out using BS203 as LoRa MQTT subscriber while BS206 as the LoRa MQTT publisher.

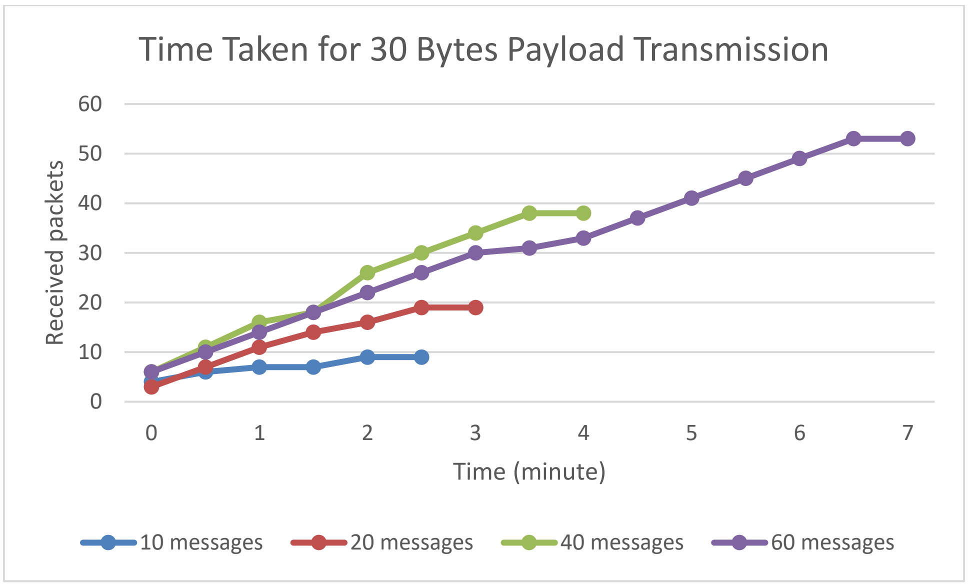


Figure 4.14: Time Taken for 30 Bytes Payload Transmission.

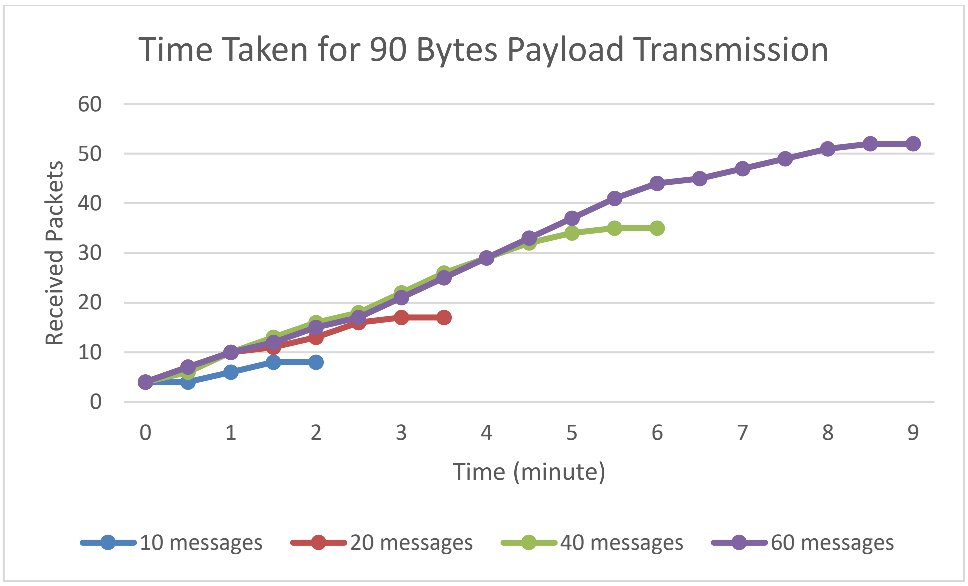


Figure 4.15: Time Taken for 90 Bytes Payload Transmission.

Based on Figure 4.14 and Figure 4.15, the number of received LoRa MQTT messages is almost linear with time. Which means the rate of LoRa MQTT message to be received is almost constant. However, the time taken to receive LoRa MQTT message is not a manipulated variable on the subscriber side. Instead, the published LoRa message is queued and buffered in MQTT broker to wait for transmission, once the published message is transmitted over NerveNet LoRa mesh, the over-the-air duration is short, thus it is almost immediately received by the subscriber side. The manipulating variable is said to be LoRa MQTT payload size because the bandwidth is fixed, higher payload size means a higher bit rate, therefore increasing the risk of LoRa signals being interfered or corrupted.

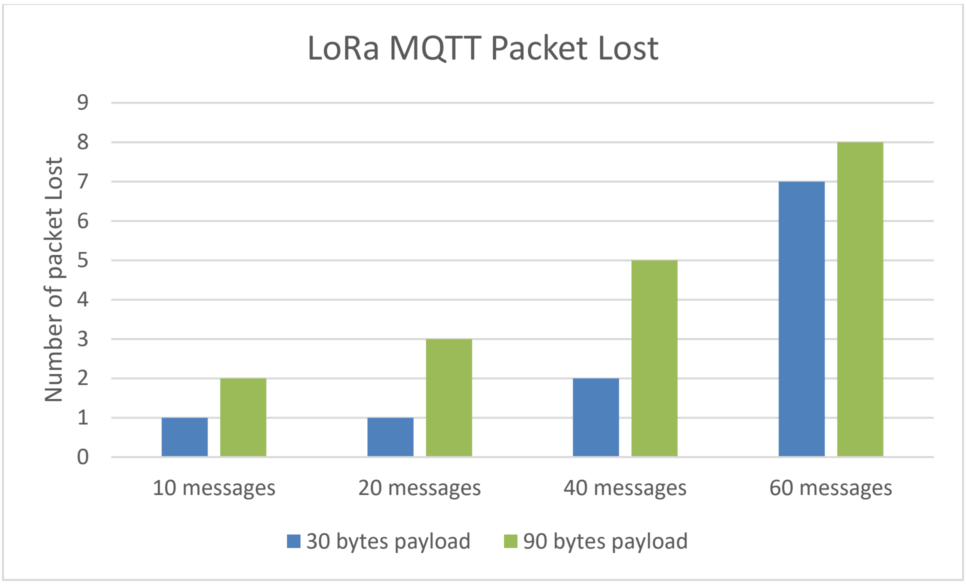


Figure 4.16: Number of LoRa MQTT Packet Lost.

The number of NerveNet MQTT LoRa messages lost is shown in Figure 4.16. By dividing the total message sent by the number of packets lost, the percentages of lost packets with 30 Bytes payload size in 10, 20, 40, and 60 messages are 10%, 10%, 5%, and 11.67% respectively. On the other hand, with the payload size of 90 Bytes, the percentage of lost packets in 10, 20, 40, and 60 messages are 20%, 15%, 12.5%, and 13.33% respectively. Hence, it is concluded that the larger the LoRa MQTT payload size, the slower the LoRa packet transmission, and the higher the risk of LoRa packet being lost.

4.7 Image File Transfer using NerveNet LoRa Mesh

LoRa is designed to send small data using IoT applications. In this project, the ability of NerveNet LoRa mesh to send and receive the whole image file is interested. To test with the image file transfer, the image file size of 2.8 kB, 5.0 kB, 9.2 kB, and 19.0 kB are used, with cases of fragmenting base64 encoded string into 70 Bytes, 80 Byes, 90 Bytes, and 100 Bytes per LoRa MQTT message, while each contains preceding row index number and a comma symbol as the delimiter. The time taken to receive all base64 encoded strings from a particular image file is tabulated in Table 4.1.

Table 4.1: Time Taken to Receive Different Sized Image File.

	2.8 kB	5.0 kB	9.2 kB
70 Bytes	7 minutes 58 seconds	30 minutes 9 seconds	1 hour 30 minutes 13 seconds
80 Bytes	8 minutes 57 seconds	25 minutes 32 seconds	1 hour 4 minutes 50 seconds
90 Bytes	11 minutes 39 seconds	21 minutes 50 seconds	1 hour 1 minute 15 seconds

Form Table 4.1 shows that the best MQTT payload size to transfer an image file is 90 Bytes of base64 encoded strings per LoRa message. When the number of total LoRa messages increase, the risk of packet being lost also increase. Thus, even though 70 Bytes and 80 Bytes of MQTT payload size could be received faster than the 90 Bytes payload size if the image size is 2.8 kB, but their performance is reduced if image size increases. However, the image files used in this experiment are considered low-resolution, they are blur images, and the useful information that could be extracted may be limited. In this experiment, sending base64 encoded strings with 100 Bytes payload size is also tested, but none of the LoRa message is received. A 19.0 kB image file is also tested in all cases of payload size, but none of the cases could receive all LoRa messages within 4 hours duration. There are also many variables that are out of control and may affect the time taken to transfer the whole image file, such as weather conditions, air humidity, and surrounding signal interference. Hence, there are cases where both NerveNet LoRa mesh MQTT subscriber does not receive any message for hours of duration.

4.8 Summary

Based on the results obtained, NerveNet Wi-Fi mesh network using both x86 and armhf machines is reliable. The TCP/UDP throughput, jitter, and latency within triangular topology are able to fulfill most of the requirements of various

IoT applications. If properly planned and configured, the NerveNet x86 Wi-Fi mesh devices are even able to handle simple Internet services during natural disaster event. For NerveNet LoRa mesh, the LoRa message could be received in a few seconds if the MQTT payload size is small. It is suitable to be implemented in IoT devices where the information to be transmitted is in short plain text. Since LoRa is not designed to transfer large-sized data, the time taken to transfer base64 encoded image is long as it is said to be not efficient or reliable to doing so. However, the results of sending fragmented base64 encoded image show that NerveNet LoRa mesh MQTT is able to transmit long messages if properly planned.

CHAPTER 5

PROBLEMS AND RECOMMENDED SOLUTIONS

5.1 Problems encountered

While using NerveNet LoRa mesh MQTT to send base64 encoded image file, there are many factors that could affect the sensitivity of subscribers to receive the original LoRa signal, such as weather conditions, air humidity, and surrounding signal interference. Sometimes, if the message amount is huge, the trailing messages may not be received for hours, even if the existing LoRa MQTT connection has no issue. This could be due to a certain reason that the LoRa messages are kept dropped by MQTT broker or not handled by NerveNet LoRa mesh framework.

5.2 Recommended Solutions

To ensure that all LoRa packets published could be received at subscriber side, more testing cases are required using different testbed environments, the distance between sender and receiver, and payload size per packet. Keep follow up with NICT to obtain the lastest NerveNet software is also recommended to deal with existing bugs.

REFERENCES

AWS Docker, 2022. *What is Docker?* Available through: Amazon Web Services website: < https://aws.amazon.com/docker/ > [Accessed 5 April 2022]

CFE-DMHA, 2019. *Malaysia Disaster Management Reference Handbooks*. Available through: United States, Center For Excellence in Disaster Management & Humanitarian Assistance website: <https://www.cfe-dmha.org/LinkClick.aspx?fileticket=he2xmI8xZFQ%3d&portalid=0> [Accessed 30 July 2021].

Chanakitkarnchok, A., Kawila, K., Sato, G., Owada, Y. and Rojviboonchai, K., 2019. Disaster-Resilient Communication Framework for Heterogeneous Vehicular Networks, *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2019, pp. 1-6, doi: 10.1109/PIMRC.2019.8904211.

Docker Container, 2022. *What is a Container?* Available through: Docker website: < https://www.docker.com/resources/what-container/ > [Accessed 6 April 2022]

Hutchison, D. and Sterbenz, J., 2018. Architecture and design for resilient networked systems, *Computer Communications*, Volume 131, Pages 13-21, ISSN 0140-3664, doi: 10.1016/j.comcom.2018.07.028.

IBM Cloud Education, 2019. *What is Cloud Storage?* Available at: <https://www.ibm.com/my-en/cloud/learn/cloud-storage> [Accessed 1 August 2021].

IBM Documentations, 2014. *What is distributed computing*. Available at: <https://www.ibm.com/docs/en/txseries/8.1.0?topic=overview-what-is-distributed-computing> [Accessed 1 August 2021].

Inoue, M. and Owada, Y., 2017. NerveNet Architecture and Its Pilot Test in Shirahama for Resilient Social Infrastructure. *IEICE Transactions on Communications.* E100.B. 10.1587/transcom.2016PFI0006.

Inoue, M., Ohnishi, M., Peng, C., Li, R. and Owada, Y., 2011. NerveNet: A Regional Platform Network for Context-Aware Services with Sensors and Actuators. *IEICE Transactions*. 94-B. 618-629. 10.1587/transcom.E94.B.618.

Khalifeh, A., Gholamhosseinian, A., and Hajibagher, N. Z., 2011. *QOS For Multimedia Applications with Emphasize on Video Conferencing*. Available at: < http://www.diva-portal.org/smash/get/diva2:504299/FULLTEXT01.pdf > [Accessed 10 April 2022]

Lee, W. K. and Mohamad, I. N., 2013. *Flood Economy Appraisal: An Overview of the Malaysian Scenario*. 10.1007/978-981-4585-02-6_23.

Lim, W. S., 2021. *Disaster Resilient Mesh Network With Data Synchronization Using Nervenet*. Available through: UTAR Malaysia, Final Year Project website: <http://eprints.utar.edu.my/4057/> [Accessed 18 August 2021].

Prajzler, V., 2019. *LoRaWAN Comfirmations and ACKs.* Available at: <https://medium.com/@prajzler/lorawan-confirmations-and-acks-ba784a56d2d7> [Accessed 6 August 2021].

Rong, C., Zhao, G., Yan, L., Cayirci, E. and Cheng, H., 2013. Wireless Network Security, Editor: John, R. V., *Network and System Security (Second Edition),* Syngress, 2014, Pages 291-317, ISBN 9780124166899, doi: 10.1016/B978-0-12-416689-9.00010-1.

The Things Network, 2021. *End Device Activation*. Available at: <https://www.thethingsnetwork.org/docs/lorawan/end-device-activation/> [Accessed: 13 August 2021].

The Things Network, 2021. *Frequency Plans by Country*. Available at: <https://www.thethingsnetwork.org/docs/lorawan/frequencies-by-country/> [Accessed: 13 August 2021].

The Things Network, 2021. *LoRaWAN Architecture*. Available at: <https://www.thethingsnetwork.org/docs/lorawan/architecture/> [Accessed: 14 August 2021].

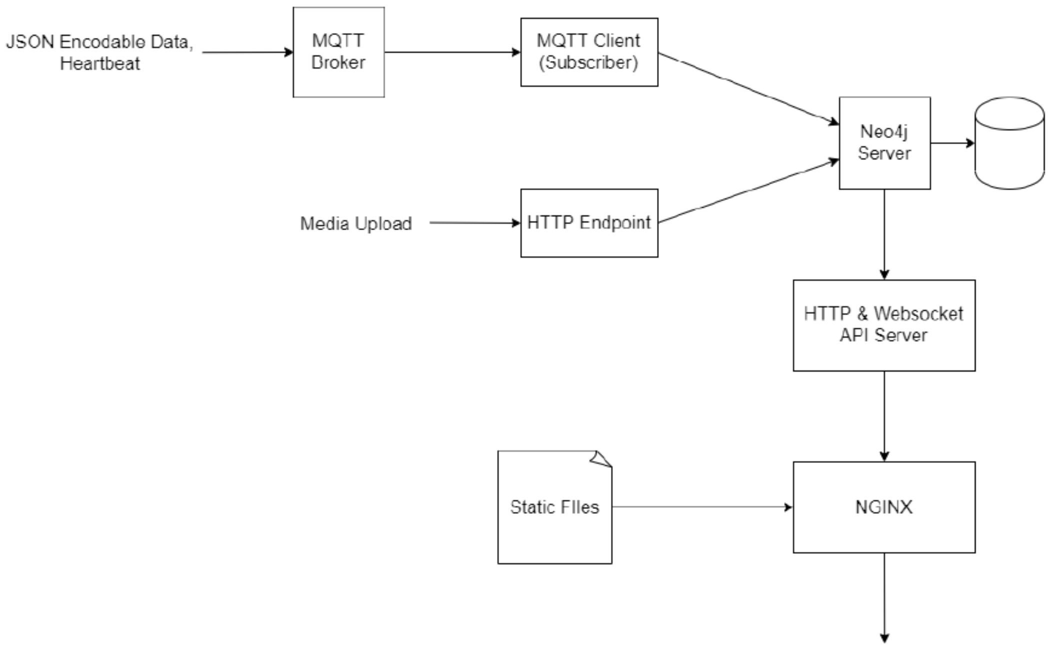
The Things Network, 2021. *Message Types*. Available at: <https://www.thethingsnetwork.org/docs/lorawan/message-types/> [Accessed: 14 August 2021].

The Things Network, 2021. *Spreading Factors*. Available at: <https://www.thethingsnetwork.org/docs/lorawan/spreading-factors/> [Accessed: 15 August 2021].

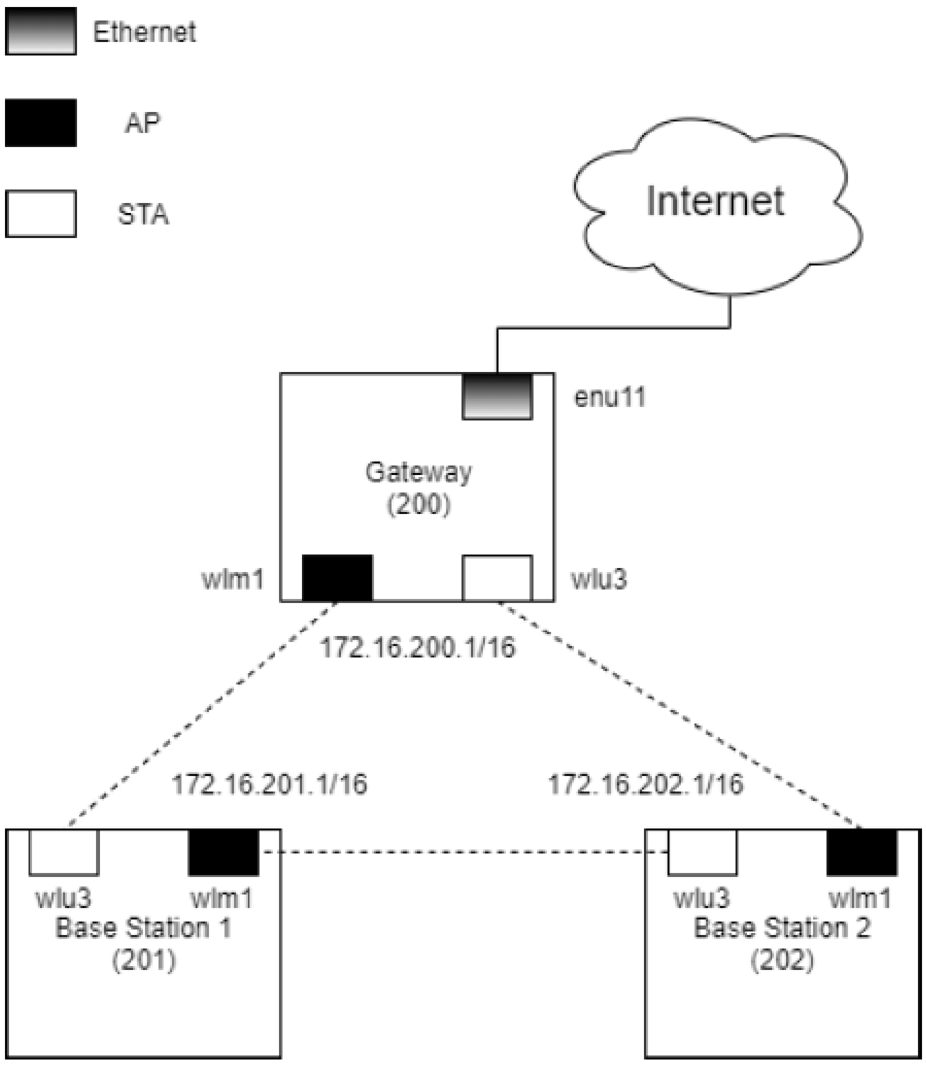
The Things Network, 2021. *What are LoRa and LoRaWAN?* Available at: <https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/> [Accessed: 15 August 2021].

APPENDICES

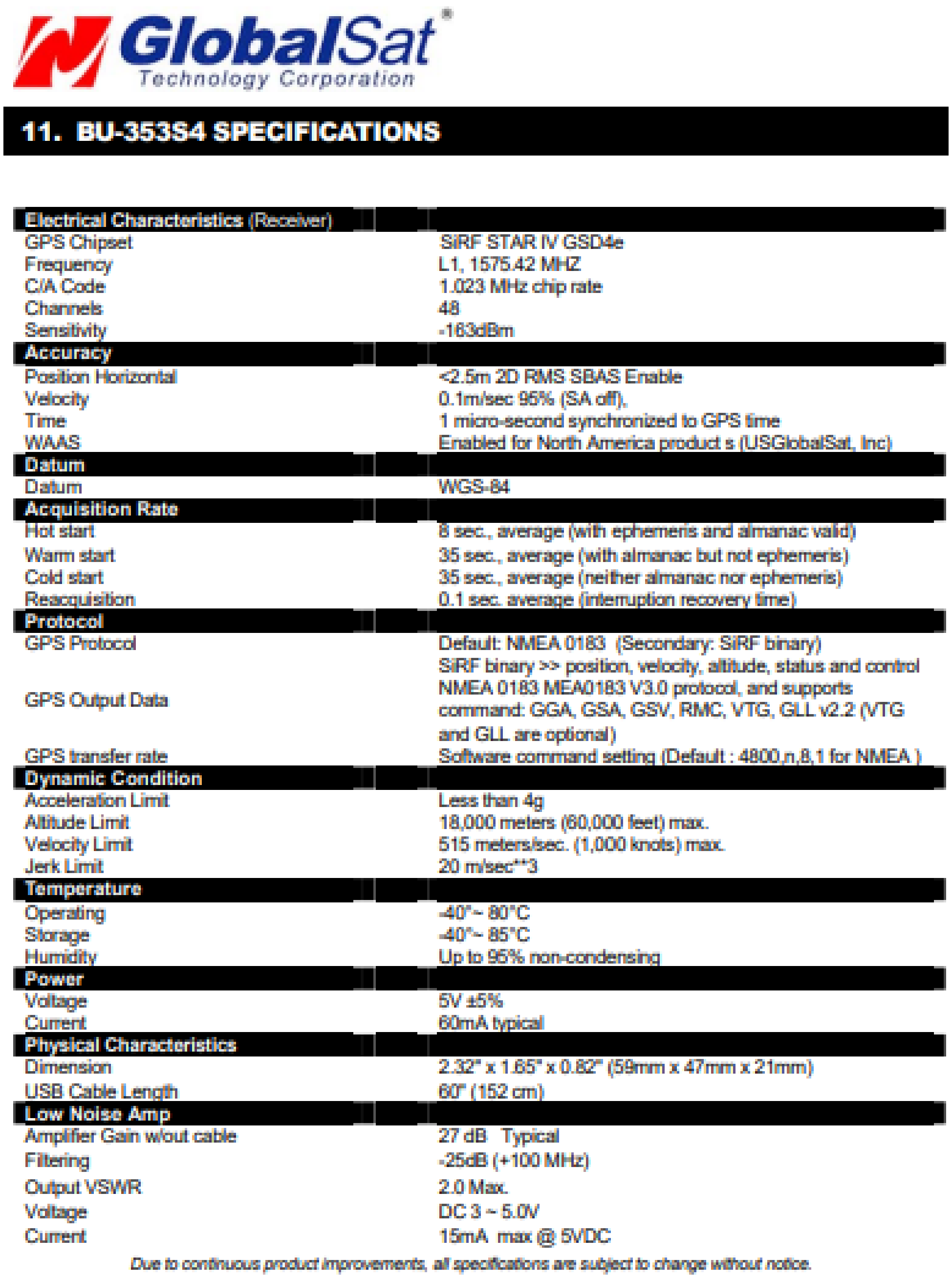
APPENDIX A: NerveNet System Deployment (Lim, 2021).



APPENDIX B: NerveNet Base Stations (Lim, 2021).



APPENDIX C: GlobalSat BU-353S4 Specifications



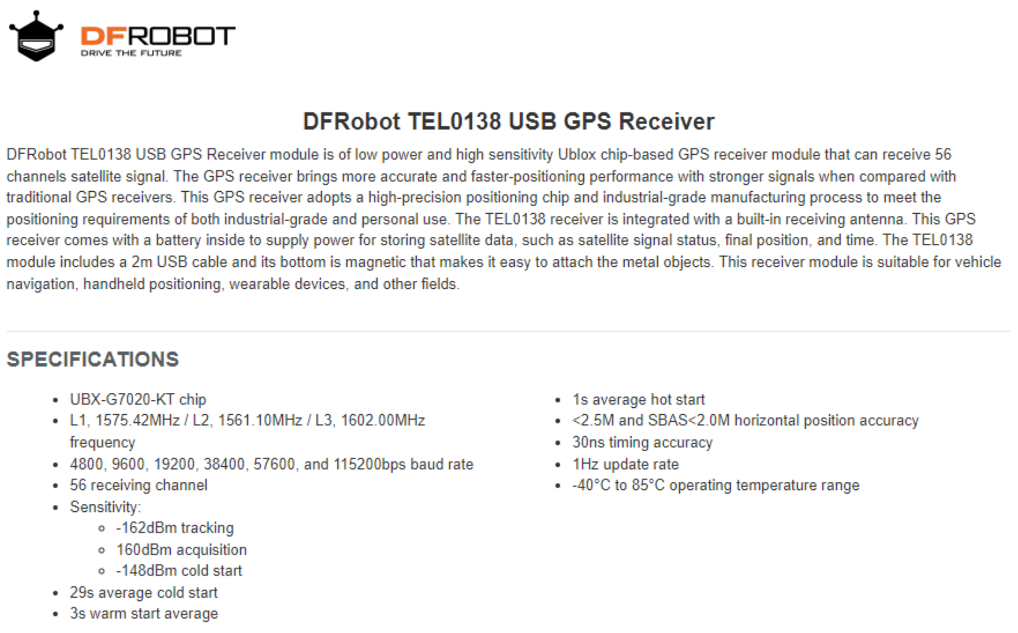
GlobalSat® Technology Corporation

11. BU-353S4 SPECIFICATIONS

Electrical Characteristics (Receiver)	
GPS Chipset	SiRF STAR IV GSD4e
Frequency	L1, 1575.42 MHZ
C/A Code	1.023 MHz chip rate
Channels	48
Sensitivity	-163dBm
Accuracy	
Position Horizontal	<2.5m 2D RMS SBAS Enable
Velocity	0.1m/sec 95% (SA off).
Time	1 micro-second synchronized to GPS time
WAAS	Enabled for North America product s (USGlobalSat, Inc)
Datum	
Datum	WGS-84
Acquisition Rate	
Hot start	8 sec., average (with ephemeris and almanac valid)
Warm start	35 sec., average (with almanac but not ephemeris)
Cold start	35 sec., average (neither almanac nor ephemeris)
Reacquisition	0.1 sec. average (interruption recovery time)
Protocol	
GPS Protocol	Default: NMEA 0183 (Secondary: SiRF binary)
GPS Output Data	SiRF binary >> position, velocity, altitude, status and control NMEA 0183 MEA0183 V3.0 protocol, and supports command: GGA, GSA, GSV, RMC, VTG, GLL v2.2 (VTG and GLL are optional)
GPS transfer rate	Software command setting (Default : 4800,n,8,1 for NMEA)
Dynamic Condition	
Acceleration Limit	Less than 4g
Altitude Limit	18,000 meters (60,000 feet) max.
Velocity Limit	515 meters/sec. (1,000 knots) max.
Jerk Limit	20 m/sec**3
Temperature	
Operating	-40°~ 80°C
Storage	-40°~ 85°C
Humidity	Up to 95% non-condensing
Power	
Voltage	5V ±5%
Current	60mA typical
Physical Characteristics	
Dimension	2.32" x 1.65" x 0.82" (59mm x 47mm x 21mm)
USB Cable Length	60" (152 cm)
Low Noise Amp	
Amplifier Gain w/out cable	27 dB Typical
Filtering	-25dB (+100 MHz)
Output VSWR	2.0 Max.
Voltage	DC 3 ~ 5.0V
Current	15mA max @ 5VDC

Due to continuous product improvements, all specifications are subject to change without notice.

APPENDIX D: DFRobot TEL0138 USB GPS Receiver Specifications



DFROBOT
DRIVE THE FUTURE

DFRobot TEL0138 USB GPS Receiver

DFRobot TEL0138 USB GPS Receiver module is of low power and high sensitivity Ublox chip-based GPS receiver module that can receive 56 channels satellite signal. The GPS receiver brings more accurate and faster-positioning performance with stronger signals when compared with traditional GPS receivers. This GPS receiver adopts a high-precision positioning chip and industrial-grade manufacturing process to meet the positioning requirements of both industrial-grade and personal use. The TEL0138 receiver is integrated with a built-in receiving antenna. This GPS receiver comes with a battery inside to supply power for storing satellite data, such as satellite signal status, final position, and time. The TEL0138 module includes a 2m USB cable and its bottom is magnetic that makes it easy to attach the metal objects. This receiver module is suitable for vehicle navigation, handheld positioning, wearable devices, and other fields.

SPECIFICATIONS

- UBX-G7020-KT chip
- L1, 1575.42MHz / L2, 1561.10MHz / L3, 1602.00MHz frequency
- 4800, 9600, 19200, 38400, 57600, and 115200bps baud rate
- 56 receiving channel
- Sensitivity:
  - -162dBm tracking
  - 160dBm acquisition
  - -148dBm cold start
- 29s average cold start
- 3s warm start average
- 1s average hot start
- <2.5M and SBAS<2.0M horizontal position accuracy
- 30ns timing accuracy
- 1Hz update rate
- -40°C to 85°C operating temperature range