

BACK-END FOR MONITORING MESH NETWORK ACTIVITY

By

Ajayi Olabode Oluwaseun

A thesis submitted in partial fulfillment of the requirements for the degree of

BSc. Honours Computer Science

University of the Western Cape

2013

Date: May 24, 2013

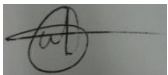
Acknowledgements

I take this opportunity to express my thoughtful gratitude and profound regards to my supervisor Professor Bill Tucker, and my co-supervisor Mr. Michael Norman and Mr. Carlos Rey-Moreno in the Computer Science department for their great guidance and constant encouragement through the course of this first part of my project thesis delivery. Your encouragement towards this project and where I am, where I will be, and where I am going shall go a long way in the journey of my life. In addition, I take this opportunity to express my deep sense of gratitude to Thrip, Centre of Excellence (CoE) for sponsoring my honours project.

My most respected brother and reliance friend, and guardian Dr. T.E Oni, I am very happy to have come across you and your family. My encounter with you has brought in success. It has, and will always give me hope to live up to my dreams and achievement. My prayer to you and your family is that God will raise you beyond your doubt and fear, and you will be celebrate for as long you are alive. I will never forget that moment you cloth me, shelter me, and taking care of me as a son and brother. You are the best. God bless you and your house-hold.

Lastly, I thank almighty, my parents, brother, sisters and friends for their constant encouragement without which this assignment would not be possible.

Yours Sincerely,



Ajayi Olabode Oluwaseun

Keywords

b.a.t.m.a.n-adv

Backend Application

Campus

Data information

Mesh Potatoes

Node

Network Infrastructure

Software Application

Software Architecture

Security

Wi-Fi

Wlan

Abstract

Wireless mesh networks are complex to monitor and institution such as University of the Western Cape is finding some way to solve the complexity problems facing wireless mesh network activity. Managing and monitoring remote or local internet connection on university campus and rural area (e.g. Eastern Cape, South Africa) involves high cost of maintenance and the cost of installing physical equipment. Thus, mesh network have the advantage of easy-to-deploy and fault-tolerant and, Organization believes that wireless technologies are less cost-effective and scalable. Wireless technologies such as access to WIFI networks or wireless local area networks (WLAN), or remote wireless internet connection on organization network (e.g. UWC free4ALL wireless connection) is becoming easier and accessible by any smart device (technologies changes). Students cell phone or end user devices can connect anytime anywhere on the network. In addition, wireless mesh networks require less network infrastructure, as compared to wired network connections and it also saves companies the cost of buying expensive network equipment or infrastructure. To manage and monitor mesh network activity it needs to be scalable, performance driven, load-balancing, and have greater efficiency on handling heavy traffics on the networks. Therefore, this project proposed the management of wireless mesh network using frontend and backend monitoring system to manage mesh network. The frontend application of this project will utilize a wireless mesh network topology visualization tools to monitor the mesh network activities on the network. The frontend system will further presents an interactive tool for visualizing mesh network topologies as well as presenting router configuration information and other network statistics on the network. For instance, in a wireless mesh network, link quality varies and, therefore the frontend visualization tool will dynamically displays the various link qualities metric for each Mesh Potatoes link. In addition, the back-end application of this project will obtain network configuration information from different Mesh Potatoes devices and the obtained values or data will be store in a file, and its size will be manage. That is, collected information at different moments in time according to the granularity defined on each node will be stored and managed. Particularly, the output values (data stored) obtained on each node will be parse to a centralized database server when the traffic on the network is expected to be low (at night) and the log files will be age when acknowledged by the server. This project will further demonstrate the frequency for checking the values and/or mechanisms to compress the data before sending the collected data to the centralize database server. This project will demonstrate a practical complete system that offers solutions to the complexity, and provide solutions to some key areas of the network

problems while carry out network monitoring activities and maintenance. This work also introduces a configuration tool designed to adapt the topology view to different wireless mesh environments.

List of Tables

Table 1 Hardware Specifications of the Mesh Potatoes Routers	20
Table 2 ‘node_status’	39
Table 3 ‘node_Configuration_information’	40
Table 4 ‘node_neighbours’	40

List of Figures

Figure 1	Diagram showing an illustration of designer’s interpretation of problem domain.....	17
Figure 2	The illustration Of the System Architecture For Requirements Analysis	19
Figure 3	Diagram of Asymmetric routing for Mesh Network	24
Figure 4	The general overview of the project design.....	26
Figure 5	Mesh Potatoes design set-up	26
Figure 6	The design interaction between PCs and Mesh Potatoes	27
Figure 7	Using a PC to collect data log file from the mesh potatoes and populate the server	28
Figure 8	CLI for Extracting and Collect the Log Files from the Mesh Potatoes	29
Figure 9	The Mesh Potato Design Architecture for Monitoring Mesh Network.....	31
Figure 10	Flow chart representing project file extractor sequence.....	33
Figure 11	Network Manager Use Cases	34
Figure 12	The Database Server Logical Design Architecture For The Backend Application.....	35
Figure 13	The Entity Relationship Diagram For The Mesh Dash Design Data Schema	36
Figure 14	The role based security permission	37

Table of Contents

Acknowledgements.....	2
Abstract.....	4
CHAPTER 1	11
PROJECT PROLOGUE.....	11
1.1 Background	12
1.2 Planning a Mesh Network.....	13
1.3 Monitoring of Mesh Network	13
CHAPTER 2	14
REQUIREMENTS DOCUMENT	14
Overview	14
2.1 User’s (Network Manager) View of the Problem	14
2.2 Brief Description of the Problem Domain	14
2.3 What Is Expected From The Software Solution?.....	15
2.4 Constraints.....	15
2.5 System Users (Network Manager)	15
CHAPTER 3	17
REQUIREMENTS ANALYSIS DOCUMENT (RAD)	17
Overview	17
3.1 Designer’s interpretation of the user’s requirements	17
3.2 System Architecture	18
3.3 Hardware Configuration Component.....	19
3.4 Mesh Potatoes (MP).....	19
3.5 Backend Database Servers (MP_DB Server).....	21
3.6 Software Configuration Package Component	21
3.6.1 Mesh Potatoes Small Campus Enterprise Network (SCEN) or Openwrt Firmware	22
3.6.2 Application File Extractor Module	22
3.6.3 Better Approach to Mobile Ad hoc Networks (b.a.t.m.a.n-adv).....	22
3.6.4 MySQL Database Application Software	23
3.7 High-Level Design of the Solution	23
3.8 Data Collection Design	23

3.9	Routing Protocol Functions.....	23
4.0	Behaviors of Wireless Mesh Networks.....	24
	Summary	24
CHAPTER 4	25
	USER INTERFACE SPECIFICATION (UIS).....	25
	Overview	25
4.1	Design Overview of the Project Integration.....	25
4.2	Mesh-Mode Wireless Network Design	26
4.3	Design Interaction Interface between PC and Mesh Potatoes (MP) Using Command Lin Interface (CLI).....	27
4.3.1	Data Collection Interface	27
CHAPTER 5	30
	HIGH LEVEL DESIGN.....	30
	Overview	30
5.1	Mesh Potato Design Architecture For Monitoring Mesh Network Only For The Backend App. 30	
5.2	Bash Script Monitoring Component	31
5.2.1	Network Data Values	31
5.2.2	Node Configuration Information	31
5.2.3	Node Status	32
5.2.4	Nodes Statistics.....	32
5.2.5	Data file Extractor App.....	32
5.2.6	Logical Database Architecture for Only the Backend Application	34
5.4	MeshDash Design Schema.....	35
5.4.1	Data Schema	36
5.5.	Business rules.....	36
CHAPTER 6	38
	LOW LEVEL DESIGN	38
	Overview	38
6.1	Pseudo-code for mesh routing.....	38
6.2	Network Input Parameters.....	38

6.3	Script Modules	38
6.3.1	Node Configuration Information:	39
6.3.2	Node Status	39
6.3.3	Node Statistics	39
6.5	User (Network Manager) Design Role (Role-Based Security)	41
	REFERENCES	42
	APPENDIX.....	43
	Project Plan for the Backend Application	43

CHAPTER 1

PROJECT PROLOGUE

A mesh network is a wireless local area network (WLAN) where each node is connected to others nodes. Wireless mesh networks are configured to allow wireless connections to be routed around fragmented paths with each signal hopping from node (source) to other node until it reaches their destination. Wireless mesh network is self-healing and self-configuring. That is, self-healing is the ability of the wireless mesh node or Mesh Potatoes (MP) to be perceive as not operating correctly and, without human involvement, it then further make the necessary changes to restore itself to normal operation. Thus, when such nodes in the network become broken, self-healing mechanisms aim at reducing the impacts from the failure, for example by adjusting parameters in mesh mode using some configuration commands so that other nodes can support the users that were supported by the failing node. Likewise, self-configuration strives towards the plug-and-play paradigm in the way that new mesh typology setup shall automatically be configured and integrated into the network. However, wireless mesh networks are difficult to monitor and its network activity (such as performance) is also difficult to manage and maintain, and because wireless mesh network operate between layers 1 and layer 3, protocols and services are becoming more complex. Wireless mesh networks are packet-switched networks with a static backbone [1]. More so, wireless mesh network (i.e. Wireless-Fidelity (Wi-Fi)) nodes which comprises of Access point (AP), and clients access (CA). Each node supports wireless connections that are low cost-effective and a convenient way to setup a mesh mode network for department (e.g uwc computer science department), and in rural area (e.g. Mankosi in Eastern Cape, South Africa) that don't have internet connections. In addition, the each node will operate not merely as a host but also as routing devices (Mesh Potatoes) that will serve as either a gateway or remote gateway to assist in Wi-Fi connection. Using this device (Mesh-Potatoes) as a router (Gateway) on the network will provide a secure data transmission and will help to forward packets to other nodes that may not be within uninterrupted wireless transmission range of their destinations [2]. Furthermore, managing a data communications network requires constant network monitoring. The goal of this project will be to create a backend application that will enable the management of data, the configuration of a remote gateway on mesh network, and the rate used with each neighbor on active links (Quality of link). More so, the frontend application of this project is to use visualization tools to collect network information and display network nodes as icons on a topographical map and some indication of link quality between nodes. Additionally, the application is to overlay the entire network

display on a geographical map, and provide node information when the corresponding node icon is clicked on. These are important features in network visualization, and are something which SCUBA lacks. This project will enable wireless network to be integrated with wireless fidelity (Wi-Fi) and other wireless technologies such as WLAN etc. [3, 4].

1.1 Background

Like other institution (e.g. the University of the Western Cape) has seen the usage of wireless technology inflatable over the past few years. It was not long ago that some universities discover the use of wireless technology. Wireless mesh networks requires proper network database management system for its data transfer or communication. The need for a frontend and backend monitoring application system for wireless mesh network is to save the network manager from manually storing and analyzing network accounting information data. Since monitoring mesh network is complex, the project proposes to develop a system that will help the network manager to monitor the network activity (both frontend and backend) system automatically. In other words, the system will help the network manager to minimize the space of the log file by ageing it from the network device (Mesh Potatoes) when outputting data information for decision making process as well as frequently checking the values or mechanisms to compress the data explored before sending it to the centralized database server, and also to determine the flexibility of packet exchange between each nodes on the network. That is, a visualization tool which shows the quality of links between routers (and nodes) in the network is useful because a network manager can use the application too to detect poor-quality links. A poor quality link between two routers could indicate a problem with one or both of the routers, and hence a problem in the network can be identified. Hence the display of link quality by a visualization tool can aid in Fault Management (because problems in the network can be identified), Performance Management (network performance can be improved if poor-quality links are avoided in routing), and Accounting Management (a poor-quality link could indicate an extremely high amount of network traffic on that link).

Furthermore, because of the growing number of data on mesh devices, this project will develop an application system, particularly a backend application for monitoring mesh network activity. More so, this project proposes to use an agile methodology approach. The reason for this approach is to manage changes and reduces risk management at any time anywhere. Therefore, this project finally proposes to help the network manager to solve and evaluate the network metrics, and determine the quality of the links on the network.

1.2 Planning a Mesh Network

For effective communication and information sharing institution (e.g. UWC) are increasingly relying on computer networks and communication tools, such as computers. This project will set up a mesh mode network that will enable a user (mesh manager computer network) to access remote databases and applications of the same organization or other private or public networks. However, implementing this set up is a complex task. This project will need to consider the goals of the organization while making a decision on which accounting information is needed. This project will also consider the network infrastructure which consists of the physical and logical components that are required to meet the networking needs of an organization. The physical components of the mesh network infrastructure are computers, server, routers (Mesh Potatoes), switches and network cables. The logical components are the software that will be used to enable the flow of data transfer or communication across the mesh network.

1.3 Monitoring of Mesh Network

This project proposes some network parameters, that is, how to use and setup a mesh network with b.a.t.m.a.n-advanced. The idea is to collect and store these network values and use it to monitor the mesh network activity. For instance, batctl holds the commands ping, traceroute, tcpdump that provides a suitable way to configure the batman-adv kernel module as well as showing debug information (such as originator tables, translation tables and the debug log) [4]. This project will further configure interfaces, check the quality of links and performance, nodes connectivity, and rate info. However, this project will limit its discussion on some batman-adv commands only as the full topic is beyond the scope of this project.

CHAPTER 2

REQUIREMENTS DOCUMENT

Overview

This chapter discusses and presents the user requirement for the proposed system. It explains what the user view of the problem is, and briefly describe the problem domain. The chapter further explains what is expected, and what constraints are applicable.

2.1 User's (Network Manager) View of the Problem

Monitoring mesh network activities can sometime be complex. This ranges from obtaining and storing network values, different instances of the value, links quality, and performance. Some network parameters are kept and cause huge amounts of data on the mesh network. Currently, the network manager is using a bash scripts on the mesh router by executing it to collect data information on the network and store them in a text file. The data collected are huge and the network manager only require less of the data for managing and monitoring the mesh network activity as well as for configuring the remote network nodes. The typical data required by the network manager would be the values outputting from Mesh Potatoes when debugging using the following batman-adv commands: athstats and ifconfig to do some aggregation, and batctl o, wlanconfig, ath0 list, and rate info parameters for doing the dynamic routing that depict link quality and performance. The network manager will want to inject traffic, copy the data and view the information to have a continual awareness and to avoid traffic jam or congestion. In addition, the network manager will want a centralize database server to manage the network values that are collected in order to feed the frontend of the application. The relevant data that are being stored in the backend application will be presented to the frontend application to allow the network manager to monitor and visualize the network activity within the internal nodes of the network.

2.2 Brief Description of the Problem Domain

A user such as a network manager needs a network monitoring system to help manage and monitor mesh network activity. The network manager will execute different network protocol commands to collect data and store them into a centralize database. The used of a configuration files will help the network manager to populate the centralize database server with the most relevant data and to keep track of different parameters used during network debugging. More so, the network manager is looking

for a way to curtail the time expending on collecting data from each mesh nodes, and to increase the effectiveness of the data transfer between a node and the centralized database servers.

2.3 What Is Expected From The Software Solution?

The software is expected to establish a remote access network connection between the local organization (e.g. UWC) and the remote site (e.g. rural area) where network activity are taken place and can be monitor. The network values or executed parameters for connection is expected to utilize low resource storage space of Mesh Potatoes devices, be secure and reliable, and increase the effectiveness of the data transfer between the batman-adv and the centralized database servers. More so, the network manager is expecting the software to manage and monitor mesh network activity in an effective way. The software should investigate the batman-adv routing protocol that will offer an essential diverse approach to route network traffic of each nodes to any destination in the network. Likewise, the software is expected to help the network manager to save time expending on data collection. That is, the more time the network manages requested output from the mesh network activity the better the quality of the information he will get or have to make decision. Therefore, the software is expected to reduce the device storage space (of the mesh potatoes) to a minimum level when populating the centralize database server with the most relevant data information.

2.4 Constraints

The software is not expected to override existing system functionality. It is important that this is taken into consideration when planning and designing the software solution. The use of the software should not disrupt the network activity, and connection between each node on the network when sending or receiving packets from source and destination. More so, the software is not expected to consume device space and network resources.

2.5 System Users (Network Manager)

The network managers or system administrators are the people who will be using the system to perform administrative tasks. They will be able to efficiently administer the network with a minimum of personnel and stress. Likewise, they will gain a broad, cost-effective view of what is required to setup an efficient software application for monitoring network activity. In addition, the software developers are fundamental in this project as they are the primary programmer of this particular software. Thus, with this software for monitoring mesh network activity, it will help and target the time network manger expends collecting and aggregating data by reducing it to a minimal level and

also reduces the manually made mistakes when computing the most important data for decision-making process.

CHAPTER 3

REQUIREMENTS ANALYSIS DOCUMENT (RAD)

Overview

This chapter discusses the requirements analysis of the system using the Village Telco device (the Mesh Potato routers) as a key factor for requirement analysis. The chapter further discusses the hardware, software and high level requirements needed to implement the user requirements of the problem domain.

3.1 Designer's interpretation of the user's requirements

The following diagram figure 1 will depict the network design requirement (network infrastructure diagram) for the proposed backend application system. Since a user (Network manager) wants systems that will help him carry out daily network monitoring a mesh mode infrastructure will be set-up to address how the user's requirement will be implemented later at implementation stage of this project.

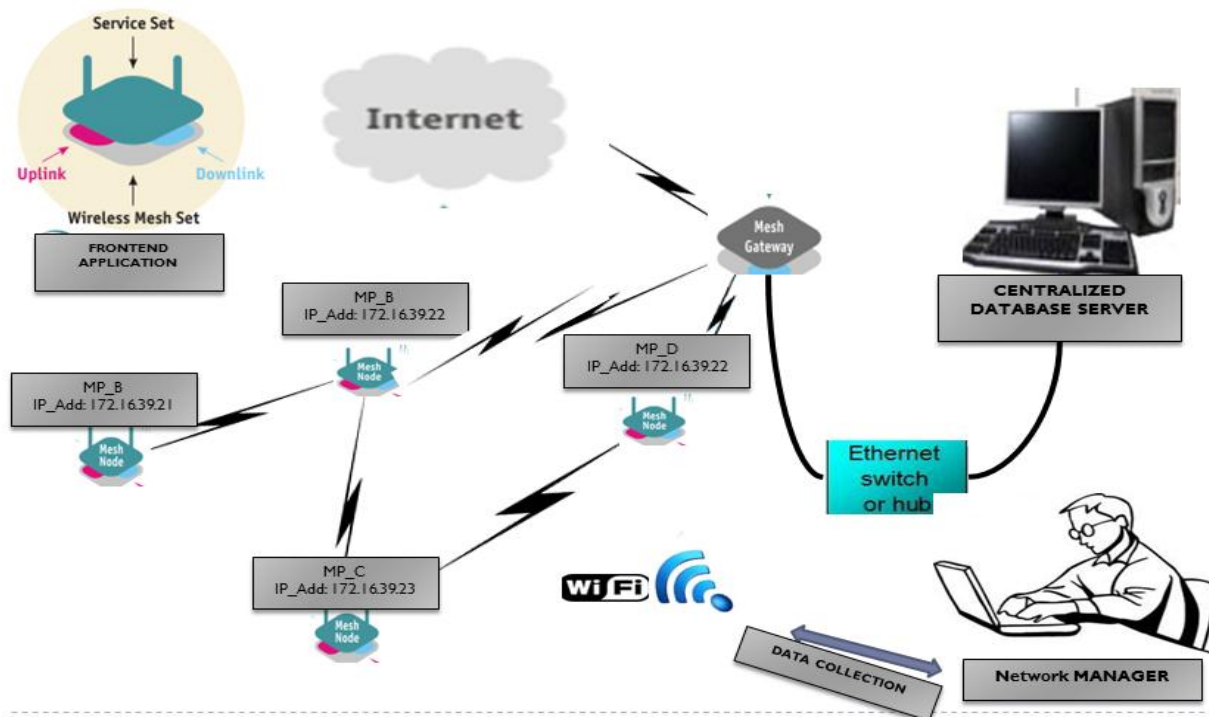


Figure 1 Diagram showing an illustration of designer's interpretation of problem domain

The above figure shows a backend network infrastructure diagram, to the right of the design requirement; the network manager was trying to configure a router on the wireless mesh network using the Linux as the based operating system. And on the left of the diagram, a wireless mesh network was

setup to allow wireless local area network (WLAN) or Wi-Fi connectivity so that the network manager who wants to configure, monitor or query all nodes on the network for their connectivity information can configure the router (Mesh Potatoes) locally or remotely using the remote access gateway. Each node MAC address will be marked with their hostname to enable easy node identification and for easy lookup of the IP address (e.g. 172.16.*.*). More so, the network manager will want to examine and monitor the quality of the link, node connection and signal strength. This setup can be seen to be a complex network setup however; this project proposes a solution to solve the network complexity. The network parameters set up will be break down and will be analyzed. After breaking down the problem, the information (Network values) from the network will be computed into database management system to assist the frontend application system. This suggested solution will best suit the operation and running process of the wireless mesh network. That is, organization will be able to increase the number of routers as per the requirements and increased the network traffic. Each node will learn routes using a very stigmeric approach [1].

3.2 System Architecture

The system architecture for this project (e.g. backend application for monitoring mesh network activity) will comprised of two main components namely; the hardware configuration and the software configuration package. These two components will be outlined and then broken down into more defined sub-components at a later stage of the software development life cycle. This project will employ the concept of data mining using Extract, Transform, and Load (ETL) for its database management system. Each sub-component will be responsible for the remote or local configuration for mesh networks. The hardware component will further discuss the hardware of the mesh devices (Mesh Potatoes) and the software necessary for the data collection for storing the networks values into the centralize database server. Likewise, the software component will be used to gather the required necessary network accounting information that the network manager wants to see on the frontend application. This project projected to help and make available information needed to supply the front end monitoring system for easy access, nodes visualization, and monitoring. However, to gather the required network information, it is important to ensure that the mesh devices (Mesh Potatoes), computers system, and switches are part of the network and have the right configuration setup properly. Therefore, nodes on the network will communicate with each other nodes in the network. More so, a java program will be used in this project to help query the entire mesh node on the network for relevant data information. Such information will include network of the next neighbors, link quality

on the mesh network and connection strength to neighbors, network traffic, and rate info for maximum and minimum throughput of the nodes on the network. The following figure 2 shows a sample of system architecture to be implemented for this project:

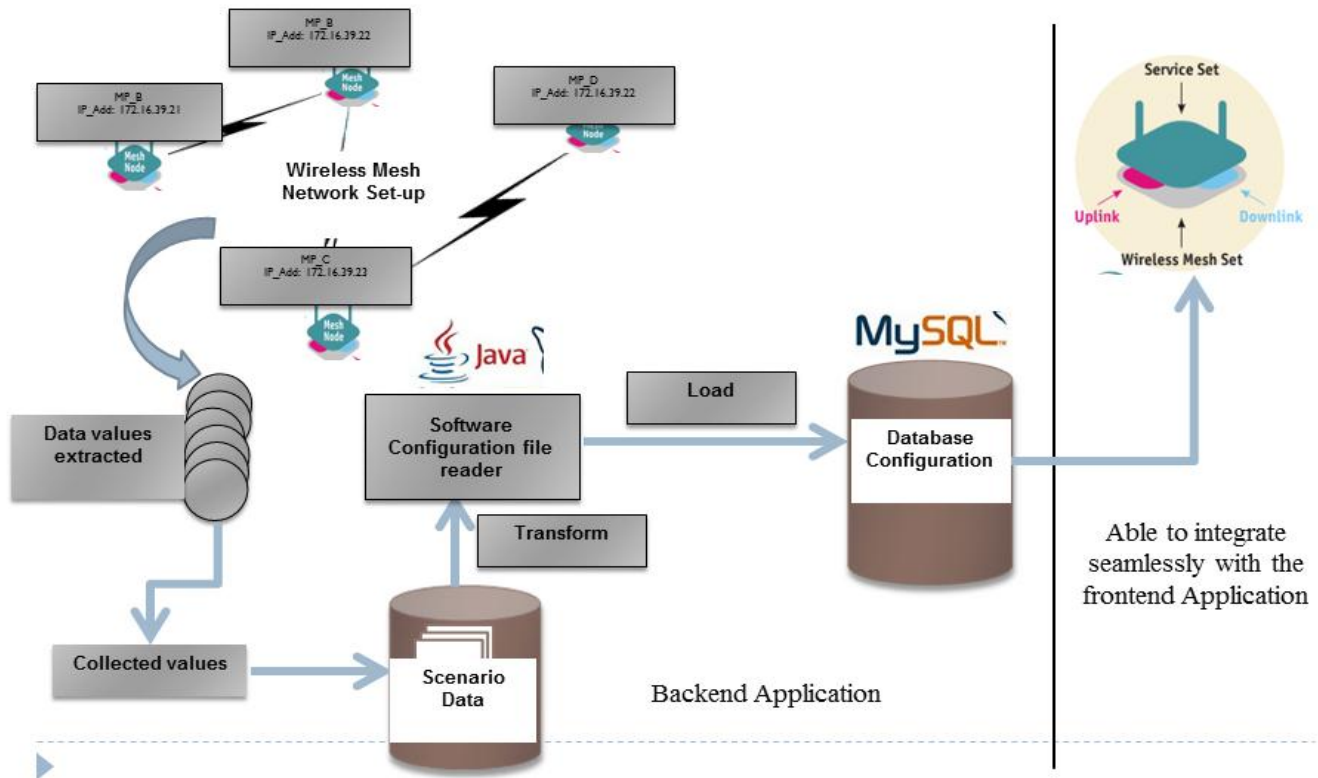


Figure 2 The illustration Of the System Architecture for Requirements Analysis

3.3 Hardware Configuration Component

The hardware configuration component will include the mesh potatoes device, switches, network cable, and computer system. This part of the project will be implemented incrementally to tackle the network manager view of the problem. The fundamental sub-components in this project include mesh potatoes and the backend hardware's.

3.4 Mesh Potatoes (MP)

The mesh potatoes (MP) which is a wireless router will be connected to the switch, and the computer system will be part of the network to form an ad hoc mesh network. This ad hoc mesh network will enable the execution and evaluation of the back end application for monitoring the network activity. The diagram illustrated in the table 1 shows each network node and their main contribution to the ad hoc wireless mesh network. Furthermore, the mesh potatoes (MP) devices utilize an Open System

Interconnection (OSI) Layer 2 protocol, even layer 1 of the OSI protocol and will basically act as one main router (Gateway), transparently connecting all the attached devices together. Mesh Potatoes have the Village Bus module installed, which act as an interface for all the data stored on them, such as their accounting information and connection to neighbors. The hardware specifications of the Mesh Potatoes routers are given below in the table 1:

Hardware	Details
Processor	MIPS 4k 180 MHz
Memory	16Mb Ram 8Mb flash EEPROM
Wireless LAN, WiFi	IEEE 802.11b/g 2.4 to 2.462 GHz frequency band Omni directional Antenna
Wireless network configuration	Ad hoc mode
Firmware	Linux kernel 2.26.3 Small Campus Enterprise Network software (SCEN) or Customized OpenWRT B.A.T.M.A.N-Advance routing potatoes

Table 1 Hardware Specifications of the Mesh Potatoes Routers

The Mesh Potatoes hardware configuration and software management shows in the table above in this project will be accessible via browser or Linux terminal sessions with an access to the core Linux operating system kernel, OpenWRT or the Small Campus Enterprise Network software (SCEN). Mesh potatoes use batman-Advance (as seen above in the table) for its mesh network routing protocol. A number of this device will be used in this project to provide data for the network activity. Each MP device will provide a Wi-Fi Access Point for remote data collection and for data analyses. An Ethernet cable will be used to connect the Mesh Potatoes (nodes) to flash the device for the wireless local area network (WLAN) configuration setup. In addition, a personal computer (PC) and network cable with a network switch will be connected together using the Ethernet port of the Mesh Potatoes node (router) to gain access to the availability of the Wi-Fi to allow the network manager connecting anytime anywhere. One key factor that is essential to take note in this project is that once a Mesh Potatoes

device is connected via Ethernet cable to a LAN router (MP), then devices on the LAN will gain access to local or internet connection as well as gain access to the LAN resources to help the network manager to connect to any of the ports on the network.

3.5 Backend Database Servers (MP_DB Server)

The backend database server is a central repository component of this backend application system and it will have the storing capability which will allow the network manager to query nodes that are on the ad hoc mesh network [3]. A computer system will be setup to cater for the operation and running of the mesh network. The backend database server will have the following specifications: IBM Sun Solaris Ultra 20 Workstation, AMD Opteron 64bit, 1800 +CPU, 1GB RAM and 80GB SATA2 hard drive. This database server will be able to manage a very large data communication on the mesh network and also requires constant network monitoring. Although, research shows that IP networks are difficult to manage (a side effect of their decentralized nature); protocols and services are becoming more complex. However, with this backend database server its primary goals are to minimize data storage by ageing log file in the MP during the local or remote configuration set-up. Analysis of network traffic, performance log file during monitoring and debugging will also be kept at a reduce space on the Mesh Potatoes. In addition, this project has foreseen the need for a structure query environment and adaptability which is most evident to the network manager. One significant problem with mesh network is the high rate of data generating per day. Furthermore, if the analysis of data could be done on-the-fly, this project will offer capabilities that current methods cannot provide. This project therefore decided that the backend database server should be lightweight stream query processing systems instead of the traditional database type (one that cannot handle large data) and that will at least be as fast as a hand-written system when querying the database to present at the frontend application which will allow the user (network manager) to by-pass the query system as needed.

3.6 Software Configuration Package Component

This project will implement a software application for downloading and collecting the mesh network data. This collected data will be analyzed and stored into the backend database server for easily accessibility for the frontend application. How this will be achieved will include the following sub-components lists namely: Small Campus Enterprise Network (SCEN) or OpenWRT, Application File Extractor module, Better Approach to Mobile Ad hoc Networks (B.A.T.M.A.N-ADV), and MySQL Database Application Software. Brief description of the sub-component lists is as follows:

3.6.1 Mesh Potatoes Small Campus Enterprise Network (SCEN) or Openwrt Firmware

This project will use the Small Campus Enterprise Network firmware that is designed to allow a collection of Mesh Potato (MP) devices to provide a data and internet network for a small campus. The intended use will typically be for a small/medium size organization which needs to set up mesh network for a geographical area. This will allow the network infrastructure to be wirelessly connected without the use of conventional LAN cabling. The meshed MP devices will utilize an OSI Layer 2 protocol and act as one large switch that will connect all other network devices together. In this project, each MP device will provides an Ethernet cable connection, and a Wi-Fi Access Point. PCs and other network devices will be connected to the Ethernet port of a Mesh Potatoes, or they will be connected wirelessly to the Wi-Fi Access Point of each Mesh Potatoes.

3.6.2 Application File Extractor Module

An application file extractor module will be programmed using the java programming language. This will help load data into the database server. It will also extract the relevant data from the Mesh Potatoes devices (configuration files) to minimize the Mesh Potatoes space capacity. The file extractor will help to collect relevant configuration and usage information from the mesh network activity and load them in the back end database to help the network manager for making decision based on the quality of link (QoS) getting from the ad hoc mesh network.

3.6.3 Better Approach to Mobile Ad hoc Networks (b.a.t.m.a.n-adv)

In this project, the B.A.T.M.A.N-ADV routing protocol will be in the form of a Linux kernel module operating system and it is ISO/OSI layer 2 level [2, 6] protocol. This routing protocol will proactively maintain information about the existence of all nodes on the network that will be accessible via single-hop or multi-hop communication links. The main purpose of routing protocol in this project is to help determine for each destination on the mesh one single-hop neighbor or multi-hop links. These links (neighbor IP address) will be utilized as a best gateway to communicate with the destination node. However, how the batman-adv routing protocol will be achieved in this project will be that all nodes will periodically broadcast packets that are known as originator messages to its neighbors. The originator messages will consist of an originator address, sending node address and a unique sequence number. Each neighbor changes the sending address to its own address and re-broadcast the message. While at the receiving end of the mesh node, the originator does a bidirectional link check to verify

that the detected link can be used in both directions. Batman-adv will optimize data flow through the mesh potatoes when injecting traffic or correcting forward errors.

3.6.4 MySQL Database Application Software

MySQL will help the project implementation of the database server. This means that the application file extractor module (Java Module) will directly be connected to the MySQL database to help in the frontend application implementation of this project. All relevant data values will be collected and loaded into MySQL database to facilitate the frontend application system for monitoring the mesh network activity.

3.7 High-Level Design of the Solution

The network manager will interact with the backend system to troubleshoot, configure and manage the Mesh Potatoes (MP) router. The network manager will further collect the necessary values to store in the central database. This part of this project will show the internal overview of the routing protocols for mesh-node with BATMAN-ADV configuration snippet on the Small Campus Enterprise Network. The high-level configuration design will include the key data collection design, key routing protocol functions and the expected behaviors of wireless mesh network.

3.8 Data Collection Design

This sub-part of this project will include collecting the values that are relevant to the managing and monitoring of the mesh network activity. This will help the network manager to facilitate easy network communications and the link quality of each network node. However, this data collection design will be an open arena for possibilities to make changes with regard to the distribution of data collected from each MP on the network for populating the database. Whenever possible and appropriate, the database designs structures will be consider and allow direct access of the system to store data collected from either local or remote configurations.

3.9 Routing Protocol Functions

The following figure 3 illustrates routing scenario using an asymmetric routing when setting up B.A.T.M.A.N-ADV routing concept [2]. This project will use this concept to design a wireless mesh network of each node such as; node A, B and C are connected via asymmetric links then, every node will have a good transmitting power (Tx) connection to one neighbor and a good receiving connection (Rx) from the other node on the network. Hence, the routing protocol that this project will use is to

discover the best neighbor towards any of its destination. Also the routing protocol will help with the node to find the best path on the mesh network.

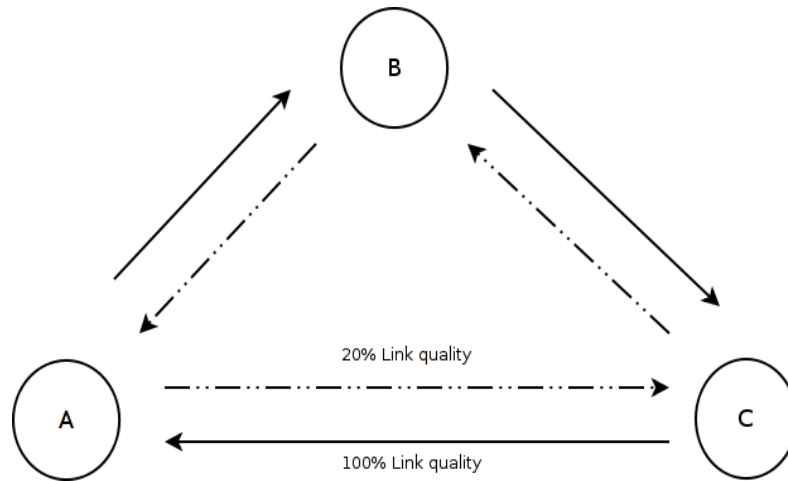


Figure 3 Diagram of Asymmetric routing for Mesh Network

4.0 Behaviors of Wireless Mesh Networks.

This project will ensure that the rate used on individual Mesh Potatoes (with each neighbor) will allow easy data communication and better quality of service (wireless links). This project will also allow the network manager to keep track of the expected time to send packet to a remote network on another region (e.g. rural area) during remote accessibility as well as avoid traffic congestion during peak hours.

Summary

The requirement analyses will help this project to focus on the system and software required or needed to implement the entire system. Moreover, the requirement analysis documents have helped to map the domain of proposed system onto the user requirements. Having analyzed the proposed system requirements, this user requirement will help and allow the system developer to design, build and implement a complete system.

CHAPTER 4

USER INTERFACE SPECIFICATION (UIS)

Overview

This chapter discusses the design of the system and describes exactly what is expected from the user interface, what it will look like, and how the user (Network Manager) will interact with the system. The user interface specification for this project will be represented in either a graphical user interface (GUI), command line interface (CLI) or an application programming interfaces (API). We developed a web based application to allow the user (Network Manager) to monitor and help in the management of the mesh network. To develop and evaluate the software, we constructed a mesh network (Ad-hoc) test bed of our own using the same hardware and software used by the Village Telco (the Mesh Potato routers SECN).

4.1 Design Overview of the Project Integration

This project will be designing and developing the frontend and backend application [9, 10] for mesh networks which will mainly focus on network monitoring system (i.e. by frequently checking the network performance, link quality etc.) and on autonomic network configuration. As a future work, we intend to integrate the frontend and backend application into a single one, using the visualization tool as the graphical interface to access all network services or activities. As we partially provided in our prototype solution figure 4, our main idea will be to build a complete management system where all kinds of data stay on the same platform and the network manager decides the type of information to view on the mesh-dash. The frontend application of this project will be aggregating data from the server and presents the information gathered as topology visualization for the network manager to view the activity on the mesh network. The motivating factor for this topology visualization was that network manager needs to view and see how the nodes on the network are naturally organized as well as their geographical displacement of nodes [11]. That is, the topology will help us to combine the aggregated data and presents them on a richer map for viewing by the network manager, allowing the network manager to quickly access all broader data about the network. For instance, the quality of all links or the number of authenticated users on each node, and with a simple request the network manager could receive more detailed data on specific nodes, such as the names of authenticated users, performance metrics like memory consumption or recent activity history. The ultimate objective of

integrating the frontend and backend application is to pave a new way for a network manager to gather needed network information in a faster and easier way than before with a disperse set of tools.

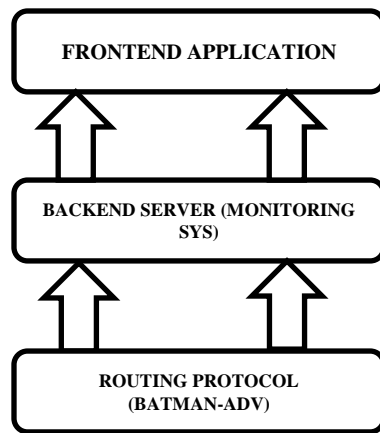


Figure 4 The general overview of the project design

4.2 Mesh-Mode Wireless Network Design

Figure 5 depicted Setup called mesh infrastructure mode. This is an important part of the project, as this is where monitoring component was developed. This design consisted of two Mesh Potatoes (MPs) and a desktop PC connected to each MP. Each MP was configured with the same subnet mask and different IP addresses to allow packets to be sent between each node. The PCs and the MP formed a wireless ad-hoc mesh network amongst themselves so that a user (Network Manager) will be able to monitor and visualization the activity of each node in the network. The following figure shows the design of Mesh Potatoes with PCs when building a shell of the user interface for the network manager test bed.

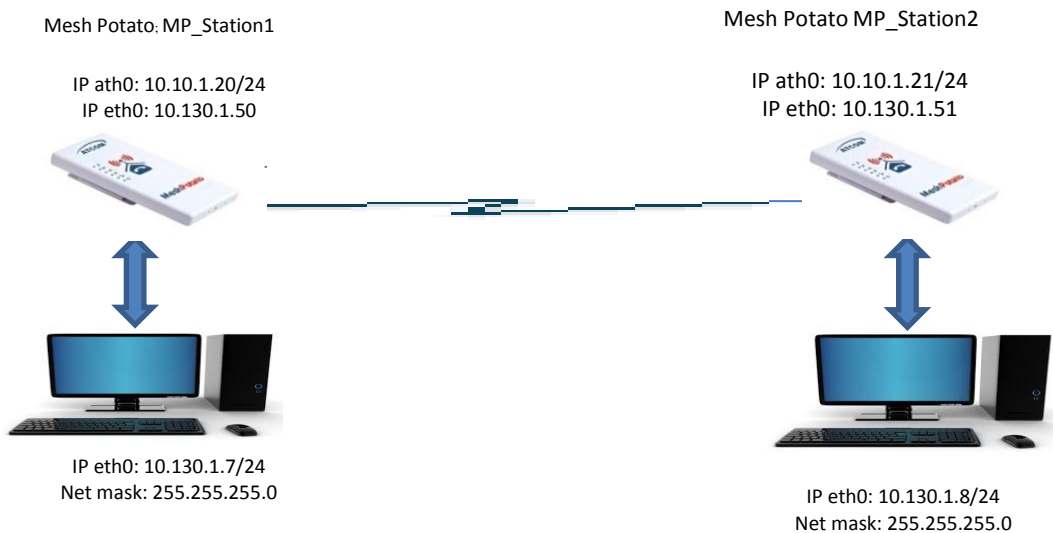


Figure 5 Mesh Potatoes design set-up

4.3 Design Interaction Interface between PC and Mesh Potatoes (MP) Using Command Line Interface (CLI)

The following figure 6 shows the process of logging into the mesh potatoes using the secure shell (ssh) on the command line interface. By ssh into the MP it will allow the network manager to configure the router for the mesh network. The process of getting into the mesh potatoes is illustrated as follows in the diagram:

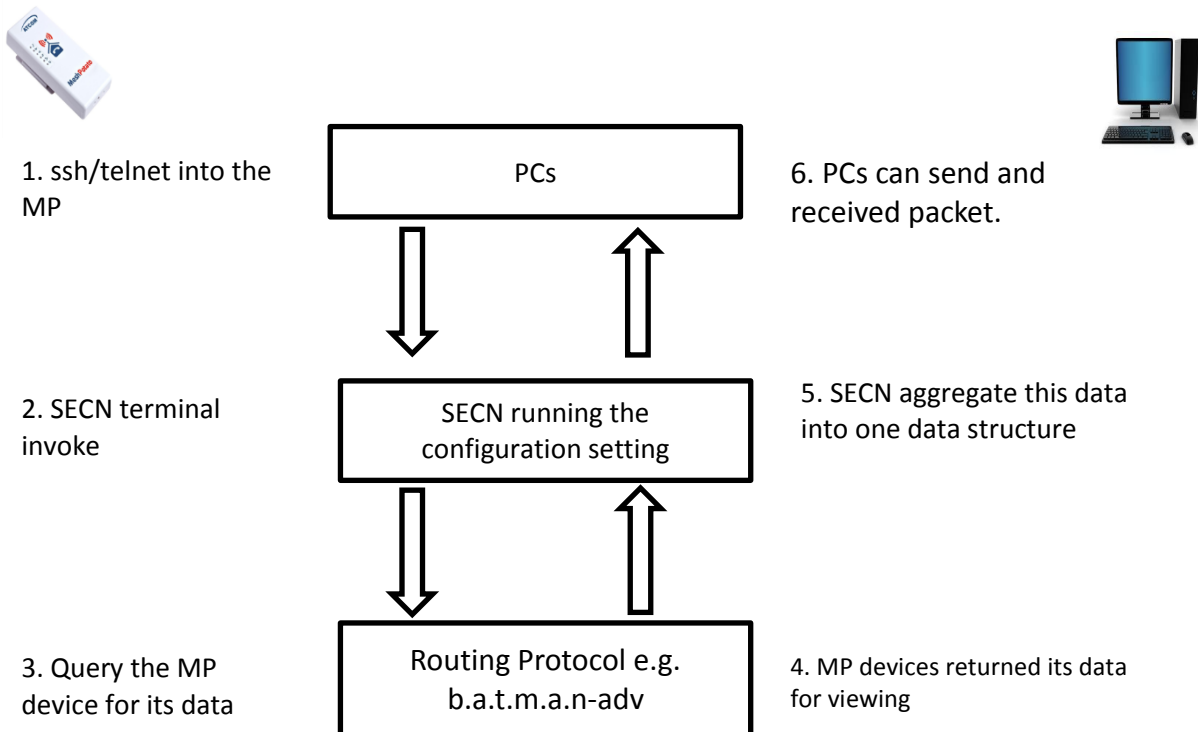


Figure 6 The design interaction between PCs and Mesh Potatoes

4.3.1 Data Collection Interface

The data collection of this backend application design interface will be represented through a command line interface (CLI). A user (Network Manager) will be able to interact with a computer program using command lines. In other words, a user interface in which a user (Network Manager) type commands instead of choosing from the menu or selecting an icon. This CLI presents the system user with several options for configuring the router (e.g. Mesh Potatoes) with a simple-to-use interface that meets both needs and wants of the user (network manager). The reason for using the CLI is that, a user (network manager) will need to configure the mesh potatoes using a secure shell (ssh). A secure shell is a network protocol that allows data to be exchanged using a secure channel between two

networked devices. More so, using the CLI, a script will be runs on the mesh potato to collect data. That data is stored in a file, and its size managed. That data will be transferred to the central server. The server employ the concept of data mining using extract, transform, and load (ETL) to parses the file to pick out cumulative and snapshot data-points that will be inserted into the database. The database in Figure 5 will be use to automatically populate the graphical user interface (GUI) of the frontend system of this project.

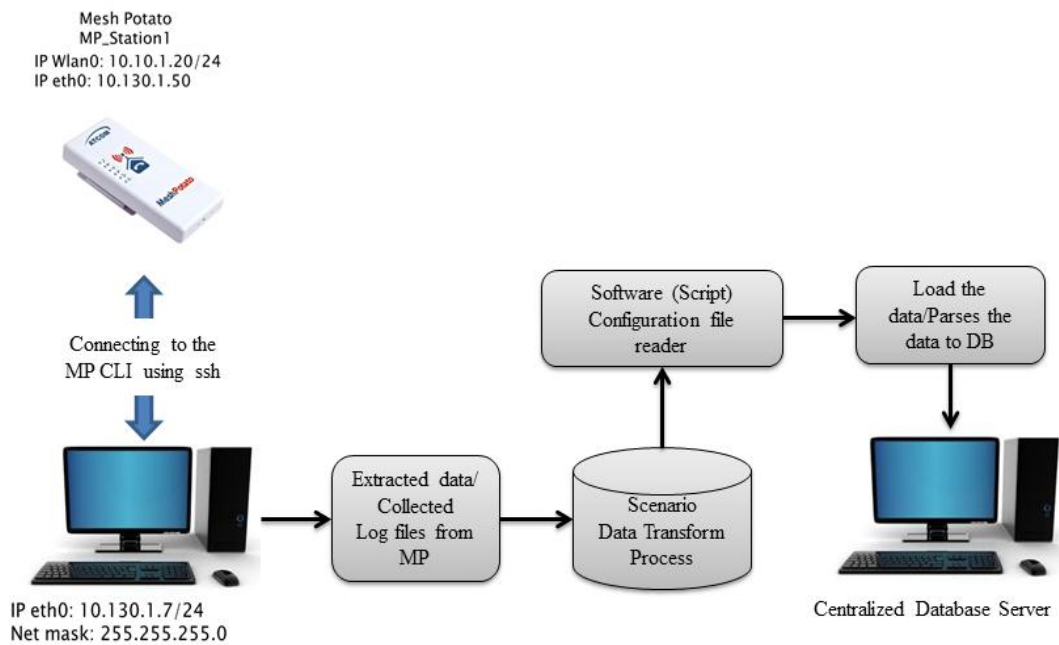


Figure 7 Using a PC to collect data log file from the mesh potatoes and populate the server

The following figure 8 shows the screenshot of the executed script using the CLI during the process of collecting the data into a log file. Figure 7 help to shows the illustration of the physical design of the process to collect data from the Mesh Potatoes. The CLI enables the user (network manager) to run a script for collecting the configuration information (e.g. ifconfig) from the mesh potatoes. By running the script (e.g. executing batctl o) on the mesh potatoes, it will gathered some configuration information and stored the output in a log file. The log file will be collected for extraction to pick out cumulative and snapshot data-points that will be inserted into the database. Before parsing the log file, necessary information will be transform using an application file extractor to parse the log file to the database server for populating the frontend GUI.

```
#!/bin/bash
touch /root/log_`hostname`_$(date +%m%d%y).log
ping 10.130.1.51 -c 10 >> /root/log_`hostname`_$(date +%m%d%y).log
ping 10.130.1.52 -c 10 >> /root/log_`hostname`_$(date +%m%d%y).log
batctl o >> /root/log_`hostname`_$(date +%m%d%y).log
batctl vm >> /root/log_`hostname`_$(date +%m%d%y).log
batctl it >> /root/log_`hostname`_$(date +%m%d%y).log
wlanconfig ath0 list >> /root/log_`hostname`_$(date +%m%d%y).log
sysctl -a >> /root/log_`hostname`_$(date +%m%d%y).log
iwconfig >> /root/log_`hostname`_$(date +%m%d%y).log
ifconfig >> /root/log_`hostname`_$(date +%m%d%y).log
uptime >> /root/log_`hostname`_$(date +%m%d%y).log
athstats >> /root/log_`hostname`_$(date +%m%d%y).log
cat /proc/net/madwifi/ath0/rate_info >> /root/log_`hostname`_$(date +%m%d%y).log
cat /etc/bat-hosts >> /root/log_`hostname`_$(date +%m%d%y).log
cat /etc/config/wireless >> /root/log_`hostname`_$(date +%m%d%y).log
```

Figure 8 CLI for Extracting and Collect the Log Files from the Mesh Potatoes

CHAPTER 5

HIGH LEVEL DESIGN

Overview

This chapter discusses high level design of the project mainly for the backend monitoring. The high level design for this project will be represented by using several subsystems. The main reason for this is that the project spans across different layer of hardware, system application and operating systems platforms. Hence, each section in this chapter follows a systematic breakdown of each of these sub components. More so, we will be presenting the object oriented view of the system, analysis of the high level design, and we will describe the objects needed to implement the system. Each one of these objects will be describe and documented, and a data dictionary providing details of each object will be provided.

5.1 Mesh Potato Design Architecture For Monitoring Mesh Network Only For The Backend App.

The following Figure 9 shows the architecture view of the backend for monitoring mesh network. This project will discuss each section of the architecture component and explain how each of the smaller modules interacts with each other. This project will be using a script (e.g. Perl script) that will enable easy communication with the network through Village Bus. That is, the Perl script will be located on the server (PC) and it will be divided into smaller modules where each module will contains the information needed to communicate with the network. This project further discusses each module as follows:

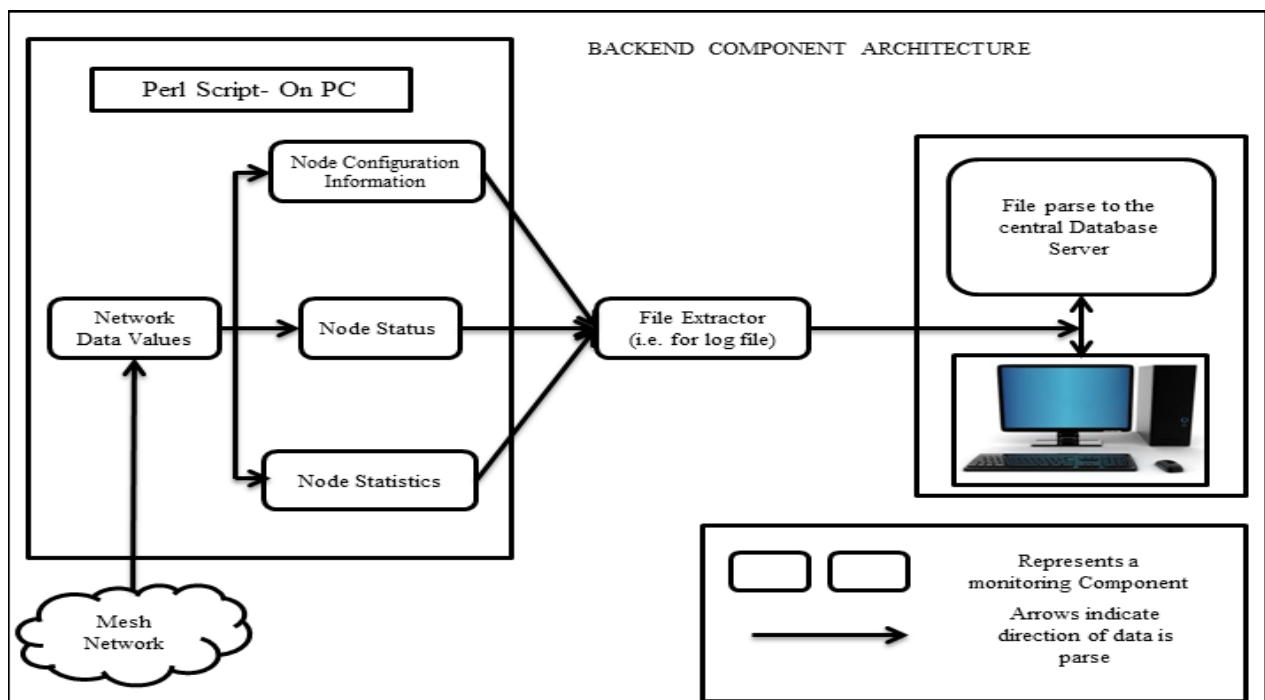


Figure 9 The Mesh Potato Design Architecture for Monitoring Mesh Network

5.2 Bash Script Monitoring Component

The bash script is placed on the server, since it is concerned with network communication. Here we will discuss all its modules.

5.2.1 Network Data Values

This module design is the main source of information that will be needed and it will be responsible for collecting data from mesh potatoes and also keep a logfile of the information collected to setup mesh network (i.e. MeshDash), and making it available to other modules component. With this logfile, we will extract or collect the relevant information needed to presents to the frontend app of this project for the network manager to visualize. However, some of these outputs are varying every instant (like the information from wlanconfig command which gives the RSSI from every neighbor, or batctl o which gives the batman metrics to reach every other node in the mesh). More so, the fluctuated instant are cumulative when executing these commands like; athstats, uptime, stats from minstrel, whereas others would remain fixed (i.e. configuration files in /etc/config/ and/etc/bat-hosts). Therefore, the idea for the backend was collecting these information's at different moments in time (according to the granularity defined for each of them), keep them in the node parsed and compressed if required/needed, populate them to a centralized server when the traffic in the network was expected to be low (at night) and age them when reception was acknowledged by the server. Note that all the process except the aging will be done using cron for scheduling the script.

5.2.2 Node Configuration Information

This layer helps collect and stored information about every node for every network on the server. The node configuration info will contain information for both a wired mesh relay, and wireless 'non-mesh' access point (AP). The wireless interface serves clients as a traditional AP. Furthermore, it will contain the list of all the routers IP addresses and their neighbors IP addresses. To collect this information we will use the Perl Script to extract the necessary data information to use populate the database. The Perl script will enables us to collect the list of each of the routers IP addresses, their byte count and their packet count. Node information like using "Batctl commands" which is the configuration and debugging tool for batman-adv gathered information such as MAC addresses of the MP(s) on the mesh. Other useful commands such as batctl translocal or transglobal commands to see the local nodes (mesh only) or all nodes accessible through the mesh (global) are also captured and store in the node

config info. Likewise, using commands like `/etc/config/batman-adv`, `/etc/config/network`, `/etc/config/wireless`, enable will us to capture the batman and wireless interface to populate the central database server.

5.2.3 Node Status

This section helps will collect and stored the information about quality of links, performance testing, client connectivity due to signal attenuation, interference from external devices, and the misbehaving or misconfigured client nodes using batman-adv routing protocol. It will also helps capture connections strenghts of various neighbors in the network. This section further will shows that a single hop IP address for a neighbor in the network might be the best next hop to it when the network manager needed to test quality of link by direct Wi-Fi link. However, the metric that batman-adv is using is not related to RSSI. Batman measures the likelihood that a transmission will reach its destination. We can have a good RSSI value, but horrible packet loss. The RSSI value alone does not give an indication that a link is actually working or performing. Therefore, the node status will helps to store the MAC addresses of each MP in the network when checking for mesh potatoes Wi-Fi settings. These can be checked using commands like `/etc/config/wireless` or via the GUI. Also we compare "iwconfig ath0" on all of the MPs to "ifconfig ath0". More so, to test for connectivity and collect the output even if the MPs interfaces are not configured (i.e. IP addresses have not been configured). We will use these commands like; `batctl tcpdump wlan0`, `batctl 00:09:45:5B:6C:98 ping`, and `batctl traceroute 00:09:45:5B:6C:98` with the MAC address of each MP to gather the node status and stored on central database server. We will observe that Ping gives an excellent indication of link quality. In other words, the ping time various greatly when a packet fails to get through, the Wi-Fi protocols are re-transmitted. Hence, re-transmission leads to variable ping times.

5.2.4 Nodes Statistics

This component helps to collect and store data information such as transmission power, node channel, transmission errors, rate-info, BSSID (i.e. is the MAC address of the AP's radio for that service set), and SSID (i.e. is the service set identifier or network name for the basic service set (BSS)). Thus, BSS is a set of stations controlled by a single coordination function on the mesh network.

5.2.5 Data file Extractor App

Figure 10 show the sequence of steps that is executed by the network manager to use this file extractor application. The file extract in Perl Scripting language is started and then various intermediate phases

are performed to eventually perform file data collection. The file extractor application will help the mesh network designer (user) to deploy proper configuration layout, access point nodes (AP) and their characteristics. More so, this design will help the network manager to minimize network infrastructure (mesh nodes and links). The constraints involved will satisfy demands that are placed by Aps (and their underlying networks). Thus, the cost of monitoring and managing mesh network will be by minimizing the mesh nodes and links.

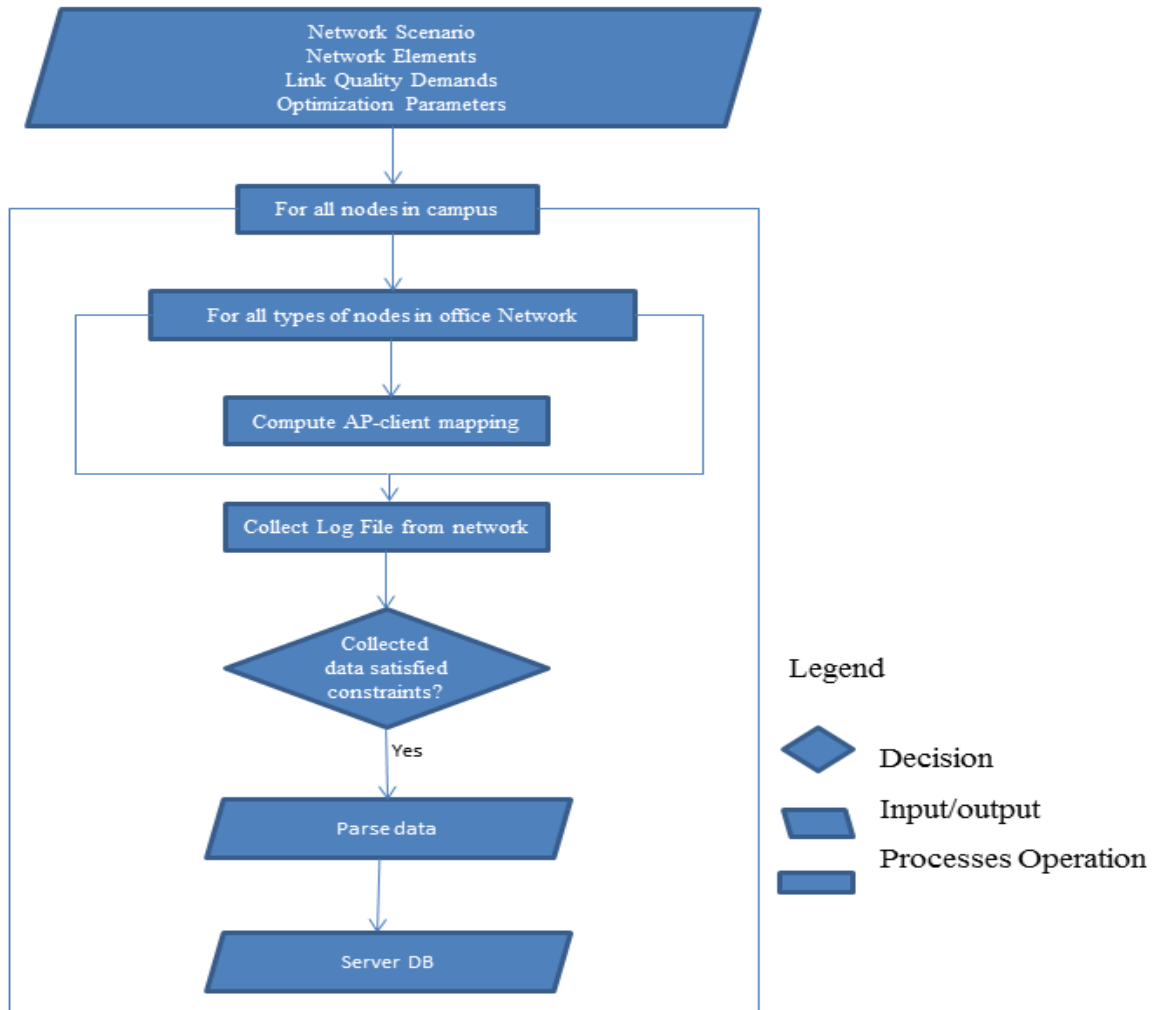


Figure 10 Flow chart representing project file extractor sequence

Furthermore, a set of scenarios for the network manager also needs to be documented as stated in the background part of this project. The chosen tool for this design is the Unified Modeling Language (UML). UML use cases will be used as a representation to the scenario taken place in the mesh network. Moreover, the UML was chosen due to the fact that it is easy and simple for the

network manager to understand the whole concept of the project design. Figure 11 shows how the network manager will interact with the system.

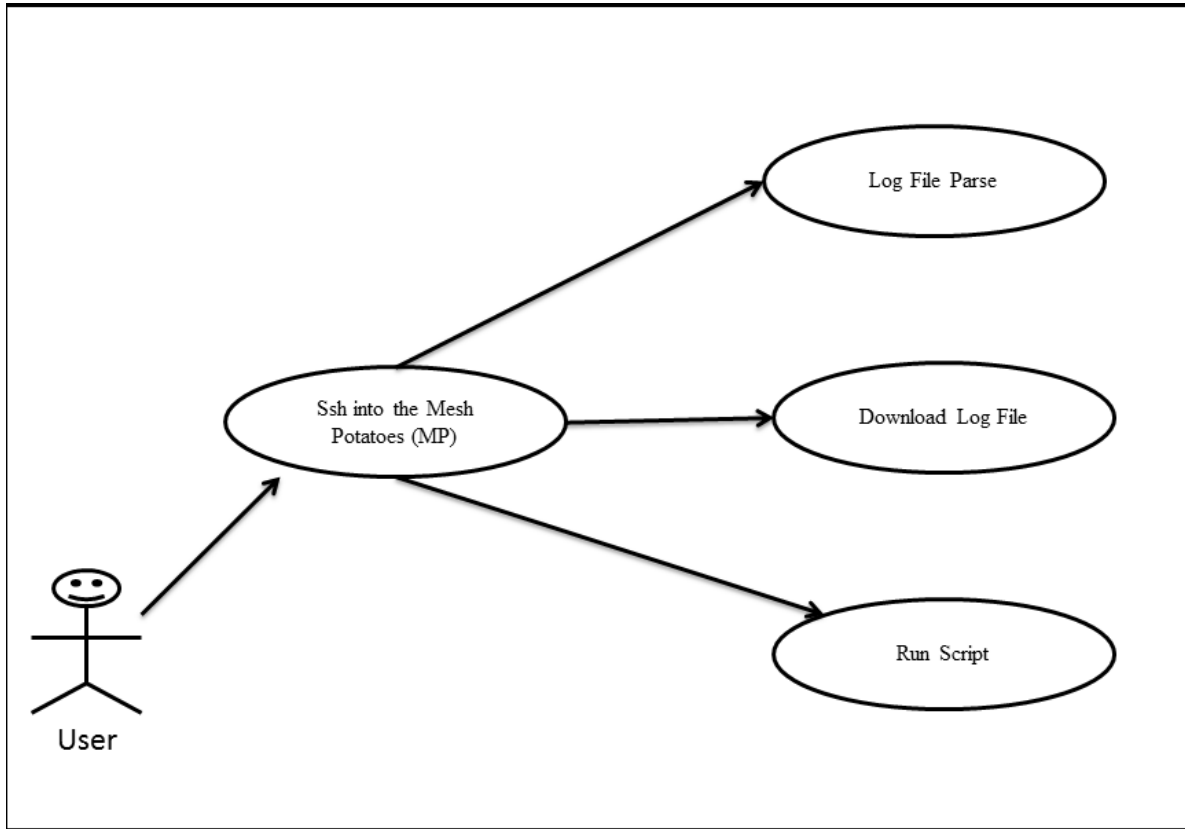


Figure 11 Network Manager Use Cases

5.2.6 Logical Database Architecture for Only the Backend Application

Figure 12 shows the database server architecture. The central database server will be made up of script that takes data from the file extractor, stored it on MySQL database, and presents it to the front-end application (MeshDash), as well as the script that takes node input, process it, and update the database. This database server enables the network manager to know the relationships between the data in the MySQL database. Hence, this database server uses type method to classify its structure. That is, we have; view task and Process task where each task has depend on each other input and output relationships. Below is a description of each task expected within the database architecture structure.

5.2.6.1 View Tasks

- 1.1 Query the database for network information to display

- 1.2 Automatically change the input of the node configuration settings on the database server
- 1.3 Present the information to the frontend app for network manager to visualize data.

5.2.6.1 Processor Tasks

- 1.1 Receive HTTP requests from the nodes containing network status information and write that information to the view
- 1.2 Generate responses to those HTTP requests based on network configuration settings in the view
- 1.3 Receive network configuration data as form input from the view components and write the data to the database.

The database design structure uses the view-processor logical architecture [5]

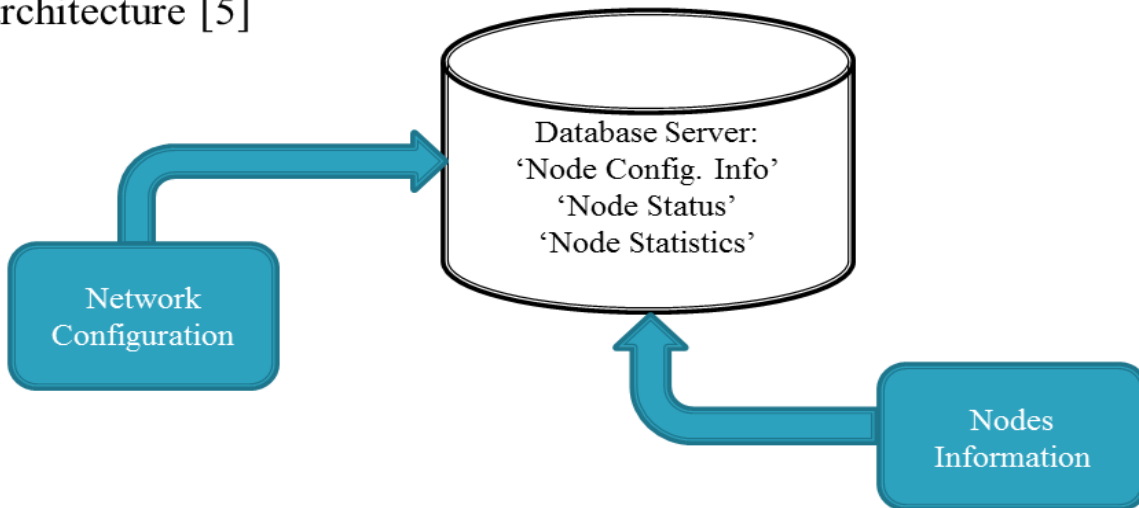


Figure 12 The Database Server Logical Design Architecture For The Backend Application

5.4 MeshDash Design Schema

The database structure will be to include the information about the mesh network and the user (network manager) who will be responsible for managing and maintaining the system. That is, data will be contained in the backend application MySQL database. By default we will be using the username “Mesh-Dash” to access the database, with default password “default”.

5.4.1 Data Schema

The figure below (Figure 13) shows the entity relationship of the data schema for the proposed system. It consists of the three dimension tables and one fact table. The three dimension tables are node_status, node_config_info and node_neighbours. Each of these tables contains a number of fields and a description of data types.

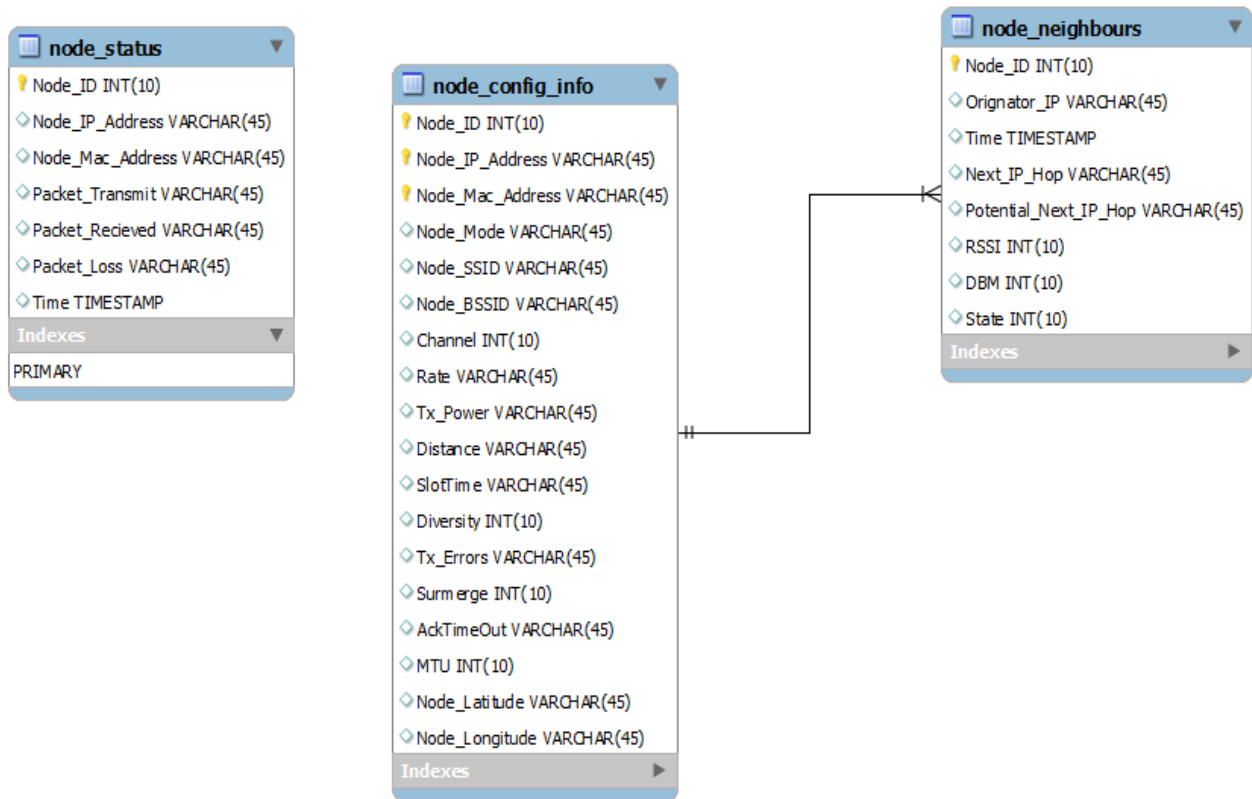


Figure 13 The Entity Relationship Diagram For The Mesh Dash Design Data Schema

5.5. Business rules

Figure 14 shows the role based security permission for the design system for both frontend and backend application. However, the following business will be apply to the design model of the

1. There will be only one network manager (user) for a given network; means there can be only one login for network monitoring. He is like a super user for monitoring the network
2. Network manager (user) can be a local user as well.
3. A single network can contain more than one local users and each local user have different kind of rights/permissions can be configured by network manager.
4. Only remote user allowed per network and rights/permissions can be configured by network manager.

The standard way we will model permissions for the system is through Role-Based Security. The typical model design will look something like this:

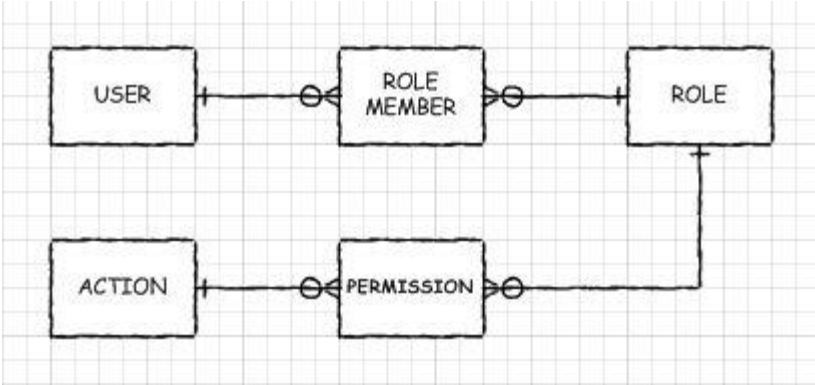


Figure 14 The role based security permission

The way it works is you have three tangible things: Users, Allowable Actions (the access that is being controlled) and Roles. Roles are groups of both users and allowable actions, therefore there are two intersection tables which record the people in each role and the actions that role permits. Some people include additional information in the role membership and permission intersection tables, like when the record was created and by whom. Some people keep that kind of audit log information in a separate table.

CHAPTER 6

LOW LEVEL DESIGN

Overview

This chapter discusses low level design of the project mainly for the backend monitoring. The low level design for this project will be represented algorithm pseudo code of the subsystems.

6.1 Pseudo-code for mesh routing.

For all ON/OFF combination of mesh_nodes do

 // on mesh nodes which have been

 Switched ON

 For all ON/OFF combination of links &

 num_of_mesh_links < max_links do

 For all demands do

 If demand < remaining_link_capacity() then

 Cost = cost_of_shortest_path() if cost < cost_min then cost_min = cost

 adjust_link_capacity()

 End

 End

End

6.2 Network Input Parameters

1. Network elements: Number of AP and potential mesh nodes
2. Network element properties: Properties of nodes and their associated links
3. Network scenario strategy: Properties of deployment layout and node distribution
4. Traffic demands: User generated traffic demands for each AP and clients
5. Link cost functions: Cost functions for fixed and variable transmit powers (TX power) and packet loss
6. Optimizer parameters and heuristics: Heuristics and initial settings for the optimizer

6.3 Script Modules

The following modules are described for the design model of the proposed system for the backend application only.

6.3.1 Node Configuration Information:

Create a network scenario generator that enables to create scenario based on deployment layout, configuration parameters and the number of nodes as well as a scenario to create locations of AP nodes and potential mesh Nodes.

6.3.2 Node Status

For every link constructor uses heuristics to generate list of potential links, an optimization preprocessor that enable us to constructs inputs for optimizer and demand matrix for the constraint file extractor application verifier. The optimizer must invoke to solve MP problem.

6.3.3 Node Statistics

A constraint verifier must be set to verify capacity constraints imposed on scenario by comparing optimizer output with demand matrix. Hence, a topology generator (Mesh Dash) must be constructs to corresponding to the capacity constrained topology. Therefore, the network manager must carry out a simulation by using an external simulator to validate the topology generated in order to visualize what is on the Mesh Dash.

6.4 Detailed Data Definitions for the MeshDash Entity Relationship Diagram

Data is contained in the 'MeshDash' MySQL database. The following table gives the detained description of each table contained in the MySQL database.

Table 2 'node_status'

Name	Type	Description
Node_ID	Int (10)	Unique network id, primary key (auto-increment)
Node_IP_Address	Varchar (45)	Node IP address
Node_Mac_Address	Varchar (45)	MP Device Mac Address
Packet_Transmit	Varchar (45)	Node transmission power
Packet_Recieved	Varchar (45)	Node transmission received
Packet_Loss	Varchar (45)	Node transmission loss
Time	TIMESTAMP	Give time stamp for network

Table 3 ‘node_configuration_information’

Name	Type	Description
Node_ID	int (10)	Unique network id, primary key and also a foreign key, (auto-increment)
Node_IP_Address	Varchar (45)	Node IP address, a foreign key, and
Node_Mac_Address	Varchar (45)	MP Device Mac Address
Node_Mode	Varchar (45)	Node mode either for client or server
Node_SSID	Varchar (45)	A unique name for the name
Node_BSSID	Varchar (45)	A unique station ID
Channel	int (10)	Signal channel values
Rate	Varchar (45)	Packet rate transmission/ Packet transmission loss
Tx_Power	Varchar (45)	Packet transmission values
Distance	Varchar (45)	Packet distance covered
SlotTime	Varchar (45)	Node interval time
Diversity	int (10)	Distance
Tx_Errors	Varchar (45)	Packet transmission error
Surmerge	int (10)	Submerged distance
AckTimeOut	Varchar (45)	Network acknowledgement time
MTU	int (10)	Message transfer control unit
Node_Latitude	Varchar (45)	Node distance cover on map latitudinal
Node_Longitude	Varchar (45)	Node distance cover on map longitudinal

Table 4 ‘node_neighbours’

Name	Type	Description
Node_ID	Int (10)	Unique network id, primary key and also a foreign key, (auto-increment)
Orignator_IP	Varchar (45)	Source (node) gateway IP address
Time	TIMESTAMP	Time stamp for node transmitting
Next_IP_Hop	Varchar (45)	Next Source (node) gateway IP address

Potential_Next_IP_Hop	Varchar (45)	Nearest Source (node) gateway IP address
RSSI	int (10)	RSSI in Unsigned values
DBM	int (10)	DBM values
State	int (10)	Node state

6.5 User (Network Manager) Design Role (Role-Based Security)

The users login into same system. In this system we will have three types of people to use the system.

They include;

1. Administrator [Network Manager]
2. Management User login [Local or Remote]
3. Local User

The following entity

2. NetworkMgr - detail about the network manager
3. Remote User - include detail about remote or Local user
4. Local User

REFERENCES

- [1]. Aichele, C., Wunderlich, S., Lindner, M., and Neumann, A. *Better Approach to Mobile Ad-hoc Networking (B.A.T.M.A.N.)* draft-wunderlich-openmesh-manet-routing-00. 2008.
- [2]. Self-study: *Broadening the concept of b.a.t.m.a.n-adv routing protocols. (2006)*. Retrieved March 03, 2013, from open mesh website: <http://www.open-mesh.org/projects/batman-adv/wiki>
- [3]. Crow, B.P., Indra, W., Sakai, P.T., and Jeong Geun, K. IEEE 802.11 *Wireless Local Area Network*. *IEEE Communications Magazine*, September (1997), 116-126
- [4]. Marlo, J. (2006). *GPRS remote access and data collection for rural telehealth project (Honours thesis, University of the Western Cape)*. Retrieved from <http://www.uwc.ac.za/~mjooste>
- [5]. M.E.M. Campista, P.M. Esposito, I.M. Moraes, L.H.M. Costa, O.C.M. Duarte, D.G. Passos, C.V.N. de Albuquerque, D.C.M. Saade, and M.G. Rubinstein, "Routing Metrics and Protocols for Wireless Mesh Networks," *IEEE Network*, vol. 22, Jan. 2008, pp. 6-12.
- [6]. Self-study: *Understand Mesh Potato devices and firmware. (2007)*. Retrieved March 01, 2013, from the Village telco website: <http://villagetelco.org/mesh-potato/>
- [7]. Self-Study: *Rendered the concept of a backend and frontend of a system. (2013)*. Retrieved March 25, 2013. *How to set up mesh network for backend system*: www.shutterstock.com
- [8]. Shepherd M. N. (2011). *An overview of Wireless Mesh Network Protocols and Voice over IP considerations (Honours thesis, University of Cape Town)*. Retrieved from <http://www.uct.ac.za>
- [9]. *ReMesh, Mesh Network Workgroup*, available at <http://mesh.ic.uff.br>, accessed in May 12, 2013.
- [10]. J. Duarte, D. Passos, R. Valle, L. Magalhães, D. Saade, C. Albuquerque. *Management Issues on Wireless Mesh Networks, Latin-American Operation and Network Management Symposium – LANOMS 2007, Petrópolis, RJ, Brazil. September 2007.*
- [11]. R. De T. Valle, D. Passos, C. Albuquerque, D. C. Muchaluat. *Mesh Topology Viewer (MTV): an SVG Based Interactive Mesh Network Topology Visualization Tool*. To be published in *IEEE Network Magazine*, 2007.

APPENDIX

1.1 Project Plan for the Backend Application

Terms	Task Descriptions
Term 1 Project Analysis User Requirements Document and Requirements Analysis Document	Understanding the problem facing Organization e.g. UWC Interview with Mr. Carlos Rey-Moreno Gather the user requirements Meeting with the supervisor Reading the specifications of the Mesh Potato and SECN firmware Installation of MySQL database, and Java database driver Still leaning towards MadWifi and Batman-adv routing protocol.
Term 2 Designing and preparing the prototype. (Completed)	Documentation(Report) System Architecture User interface specification, Creating Mesh Network Setting-up Mesh Nodes, Designing and Configuration the batman-adv routing protocol on MP
Term 3 Implementation and coding	Further research to re-modify Carlos Script for project Implementation to log real-time Network Activity. Capturing Signal received from neighbours. Setting gateway bandwidth for Uplink and downlink Speed of the gateway. Setting the Queue Discipline for the Mesh Nodes. Working with the department Mesh Node for easy Integration. E.g. VOIP and Internet Usage.
Term 4 Testing and Evaluation	Ability to test the application Remotely Qualitative evaluating Method of testing the App.