

A Lightweight Ad-hoc ICT Infrastructure for Post-Disaster Situations

Vaibhav Garg

A Thesis Submitted to
Indian Institute of Technology Hyderabad
In Partial Fulfillment of the Requirements for
The Degree of Master of Technology



भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Department of Computer Science and Engineering

June 2015

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

Vaibhav Garg

(Signature)

Vaibhav Garg


(Vaibhav Garg)


CS12M1014


(Roll No.)

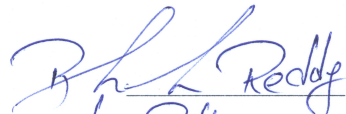
Approval Sheet

This Thesis entitled A Lightweight Ad-hoc ICT Infrastructure for Post-Disaster Situations by Vaibhav Garg is approved for the degree of Master of Technology from IIT Hyderabad


Mohammed Zafar Ali Khan (Examiner)
Dept. of Electrical Engineering
IITH


Subrahmanyam Kalyansundaram (Examiner)
Dept. of Computer Science and Engineering
IITH


(Dr. Kotaro Kataoka) Adviser
Dept. of Computer Science and Engineering
IITH


T. Bheemarajyn Reddy (Chairman)
Dept. of CSE
IITH

Acknowledgements

This thesis work required a lot of research, dedication and hard work and it was made possible because of the efforts of a lot of people. First of all, I would like to thank my guide Dr. Kotaro Kataoka, who proposed this idea and provided me all the resources to carry out this research. Without his constant support, motivation and valuable feedback, I would not have been able to write this thesis. I am also grateful to Mr. Rohith TSS, intern from Manipal Institute of Technology and Mr. Sahil Sachdeva, B. Tech. final year student, who helped me in some part of thesis work. Special thanks are due to Mr. Rohit Katiyar who assisted me in conducting field trials. Lastly, I would like to thank almighty, friends and family for their blessings, support and constant encouragement.

Abstract

One cannot expect communication resources to be readily available at disaster-affected site. Moreover, Information and Communication Technology (ICT) rescue supplies tends to be for heavy-duty and limited in quantity. This affects the information exchange and may cause delay in rescue operation, which is highly undesirable. In this thesis, we aim to utilize the time between rescue workers reaching the disaster-affected site and when the Internet connection is made available. The idea is to start the local communication and information exchange without waiting for the recovery of Internet connection. We propose a self-contained software package in the form of Linux Live USB flash drive. Our solution requires that Linux Live USB flash drives, laptop computers and external USB Wi-Fi NICs are provided to rescue workers as rescue supplies. Live USB consists of all the packages that are required for supporting ad-hoc networking. Rescue workers boot the laptop computer using Live USB flash drive to get access to computing and networking resources of laptop computer. Once booted, Live USB creates ad-hoc network using USB Wi-Fi NIC, which can be used for providing services such as web server, file server, text/audio/video communication within local network. Internal Wi-Fi card of laptop computer can be used to provide connectivity to user devices such as smartphones using Wi-Fi hotspot. Live USB does not access host Operating System (OS) and/or internal storage (Hard Disk/SSD) for any operation. For the same reason, any available laptop at disaster-affected site can be used. Graphical User Interface (GUI) is provided for all the operations so that end-user need not have any knowledge of guest OS on Live USB. If available (through LAN, satellite or 3G), Internet connection can be easily shared with all the devices including user terminals. The experiments and field trials showed that this kind of network can be used in emergency situations such as post-disaster recovery.

Contents

Declaration	ii
Approval Sheet	iii
Acknowledgements	iv
Abstract	v
Nomenclature	vi
1 Introduction	1
1.1 A Case Study of ICT Environment Recovery Activities after 2011 Tohoku Earthquake and Tsunami in Japan	2
1.1.1 Challenges in Post-Disaster Recovery of ICT Environment	3
1.2 Overview of the Work	3
1.3 Thesis Outline	3
2 Wireless ad-hoc networking using Live USB	4
2.1 Expected Deployment Scenario	5
2.2 Wireless Ad-hoc Networking in Post-disaster Situations	6
2.2.1 Single Channel Scenario	6
2.2.2 Multi Channel Scenario	7
3 System Design and Architecture	8
3.1 Live USB Server and Client	8
3.2 Network Design	9
3.2.1 IP Addressing in Single Channel Scenario	9
3.2.2 IP Addressing in Multi Channel Scenario	9
3.2.3 IP Addressing for Wi-Fi Hotspot	12
3.2.4 Routing in Wireless Ad-hoc Network	12
3.3 Local Online Services	13
3.4 Web Based Communication and Information Exchange	13
3.5 Remote Troubleshooting	14
3.6 System Monitor and Health Check	14
4 System Implementation	16
4.1 Hardware and Software Details	16
4.2 Creating an Ad-hoc Network	16

4.2.1	Configuring Wi-Fi Interface in Ad-hoc Mode and Starting Mesh	17
4.2.2	IP address Assignment	17
4.3	Wi-Fi Hotspot	18
4.3.1	Configuration for DHCP Server	18
4.3.2	Configuration for Hotspot	19
4.3.3	Starting the Hotspot	19
4.4	Sharing Internet between two interfaces	20
4.4.1	Obtaining IP Address from DHCP	20
4.4.2	Setting up and down interfaces for Internet sharing	20
4.4.3	Setting up Squid as Transparent Proxy	20
4.4.4	Forwarding HTTP traffic to Squid Proxy	21
4.4.5	Routing Using RIPv2	21
4.5	GUI	22
4.5.1	Server GUI	22
4.5.2	Client GUI	23
4.6	Handling the Disconnections	25
5	Evaluation	28
5.1	Tested Hardware	28
5.2	Phase 1: Single Channel Scenario Using Batman-adv	28
5.3	Phase 2: Single Channel vs Multi Channel Batman-adv vs OLSR	30
5.3.1	UDP Throughput	31
5.3.2	Packet Loss Rate	31
5.3.3	TCP Throughput for Parallel Connections	33
5.4	System Evaluation	34
6	Related Work	36
7	Conclusion and Future Work	38
	References	39

Chapter 1

Introduction

An important aspect of post-disaster recovery is to restore the Internet connection at disaster-affected site. Lack of Internet connectivity leads to limited or no communication, since people rely on Internet connection (e.g., Google Hangouts, Facebook), even when they wish to communicate locally. Even if some form of communication is available, it may get choked, if a large number of people try to access it simultaneously. Due to no reliable source of communication, rescue operation may get affected. Moreover, people will get mentally stressed if they are not aware of well-being of each other. Given the critical nature of communication in post-disaster aftermath, we propose a self-contained software package in the form of Linux Live USB flash drive. The proposed software packages run on the guest operating system booted from a Live USB flash drive so that a laptop or desktop computer acts as Live USB node. Live USB node forms an ad-hoc connection with adjacent one to enable local, and if available, global communication in the Internet. This system has following features:

- *Light-weight and Small*: The software suite is independent of any heavy and big machinery. A laptop computer is enough to boot from Live USB and start serving for other devices. It enables rescue supplies to be carried easily to the disaster-affected site.
- *Inexpensive*: USB flash drives are inexpensive and easily available. Laptop computers need not have any special configuration. USB wireless cards are also available at low price.
- *Easy to setup*: Our system is preconfigured and plug-and-play. If any configuration is required on-the-fly, GUI helps users without special skills and knowledge.
- *Remote Troubleshooting*: Once Internet connection is available, Live USB node connects to remote server, present in global Internet. Thus technical staff can stay at any place and troubleshoot issues through the remote connection.
- *Everything through Wireless*: Live USB node deploys Wi-Fi hotspot through which many user devices can communicate. Multiple Live USB nodes can establish wireless ad-hoc network to form a larger network coverage. If Internet connection through LAN, 3G, satellite, etc. is available, then, user devices in the domain can also use the Internet connection through hotspot and ad-hoc network.
- *Local Online Services*: Our system provides local online services within the local network that do not depend on availability of Internet connection at disaster site.

1.1 A Case Study of ICT Environment Recovery Activities after 2011 Tohoku Earthquake and Tsunami in Japan

Utani et al. [1] summarized the activities after a massive earthquake of 9.0 Mw magnitude hit Japan on March 11, 2011 at 14:46 JST. The earthquake resulted in tsunami, upto 40.5 m above sea level, surging coastal regions of north eastern Japan. Tsunami affected nuclear power stations, destroying reactor's necessary cooling units and other vital mechanisms. This resulted in unfortunate explosions, causing radioactive fallouts. At least 2,70,000 buildings and 4,00,000 people were affected as a result of this, see Figure 1.1.



Figure 1.1: Affected sites after 2011 Tohoku Earthquake and Tsunami that impacted the existing ICT infrastructure. People took shelter at school building and had minimum access to rescue supply such as food, clothes, and medical treatment. ICT infrastructure was under constrained condition because of damages and slow recovery of land line cables.

Since Japan is a country known for frequent earthquakes, their government has many systems for disaster relief in place. However, due to chaotic situation caused by tsunami and Fukushima-daiichi nuclear power plant accidents, power management system and underground optical fiber cable took a hit, and many of the systems could not perform as expected.

- Earthquake Early Warning (EEW) [2] is a system to alert people about incoming seismic waves. The alert is sent to TV broadcasts, cellphones, transportation services, and other systems, in order to allow people to prepare appropriately. This system performed as expected.
- System for Prediction of Environmental Emergency Dose Information (SPEEDI) [3] is a prediction system for radioactivity diffusion, towards rapid assessment of pollution in the surrounding areas of the disaster site. The system could not work properly due to power outage and some other factors.
- Emergency Medical Information System (EMIS) [4] is a system for exchanging information about hospitals, beds, patients, and departments in emergency situations. However, the communication network needed for information exchange was damaged causing disconnections. Also, wireless services could not work due to damage to their base stations.

1.1.1 Challenges in Post-Disaster Recovery of ICT Environment

Kataoka et al. [5] shared their experience and lessons learnt from ICT environment recovery after the disaster occurred.

- Internet connection was not available even after one week of disaster.
- An expensive system is not sustainable to support multiple distributed sites.
- A big or heavy system introduces difficulty of logistics including packaging, dispatching, transportation, storage, planning and management.
- There might not be enough human resource to setup a technically complicated system. Moreover, handling multiple distributed sites will also be difficult.
- System should be up and running for a longer duration to reduce maintenance cost. Remote access is also important for health check and maintenance of the system.
- Some softwares/programs are written such that they need Internet connectivity to work even if people wish to communicate locally.
- Due to widespread use of smartphones, serving Wi-Fi hotspot provided connectivity to everyone.
- Thanks to the good preparedness for the disaster, power supply was sufficient to support additional devices. However, disaster sites faced power cuts.
- Some desktop and laptop computers were available but could not be used since they were password protected.
- Mode of information dissemination was basically paper/notice board based in disaster sites. Digitizing such information and making it available online can drastically improve the effectiveness of recovery efforts.

1.2 Overview of the Work

As part of this work, we implement Live USB node with capability of enabling (i) wireless ad-hoc network, (ii) Wi-Fi hotspot for user devices like smartphones or tablets and (iii) local online ICT services that do not depend on Internet connectivity. We provide GUI for the users that are not comfortable with scripts and commands. The health of Live USB nodes can be checked in real-time. The work is well documented so that user can get familiar with the system. The project homepage is available at [6].

1.3 Thesis Outline

The thesis is structured as follows. Chapter 2 presents how wireless ad-hoc network can be supported using Linux Live USB flash drive. In chapter 3, we describe our System Design and Architecture. Implementation details are discussed in chapter 4. Chapter 5 covers System Evaluation. Related Work is discussed in Chapter 6. We conclude the thesis in chapter 7.

Chapter 2

Wireless ad-hoc networking using Live USB

Our solution assumes that laptop computers, Linux Live USB flash drives and USB Wi-Fi Network Interface Cards (NICs) are provided to rescue workers as rescue supplies. Rescue workers boot laptop computer using guest OS on Live USB flash drive and start using the packages present on Live USB. This solution considers most of the problems discussed in Section 1.1.1 and has three major benefits.

- The maintenance of software suite such as installing, deleting, upgrading and testing applications and the guest operating system can be done at a single place. So, users can save a huge amount of time and network bandwidth for setting up the disaster rescue mode of Operating Systems.
- Users need not worry about what Operating System is installed on the laptop computer. They need not have IT skills to handle available features as User Interface can be designed to be beginner-friendly. There is no need to know the password, if any, of the host Operating System installed on laptop computer. Almost any laptop computer can be used unless that system's Basic Input/Output System (BIOS) is password protected. Moreover, if a system is password protected, it can be booted using Live USB and connect to Wi-Fi hotspot to access services.
- A Live USB node can be configured to connect to remote system, also known as Virtual Private Network (VPN) server, for remote login. This VPN server is present in global Internet, and Live USB node connects to it if and only when it can access Internet. VPN connection will enable human resources such as IT engineers to stay outside of disaster sites without needing to travel too much. If the remote login does not help in trouble shooting, the simple solutions are system reboot or replacement of the USB flash drive, that are not beyond the skills of people in a disaster site.

There are two approaches to form an ad-hoc network. In first approach, all the nodes are configured in the same frequency channel, forming a very large single mesh cloud. Configuration of such mesh network is very easy, but handling interference and other performance parameters becomes an issue. Second approach is to configure only two adjacent nodes in the same channel, forming

multiple meshes containing two nodes in each mesh. This can be achieved by using multiple Network Interface Cards (NICs) on the same node. The nodes in different meshes can be connected using mesh gateways.

2.1 Expected Deployment Scenario

Figure 2.1 presents the expected deployment scenario of the system. Laptop computers boot using Live USB flash drives that are carried to or prepared in a disaster affected site. The guest OS is pre-configured to support wireless ad-hoc networking. Once the guest OS is booted, Wi-Fi devices of laptops try to connect to neighbor Live USB nodes.

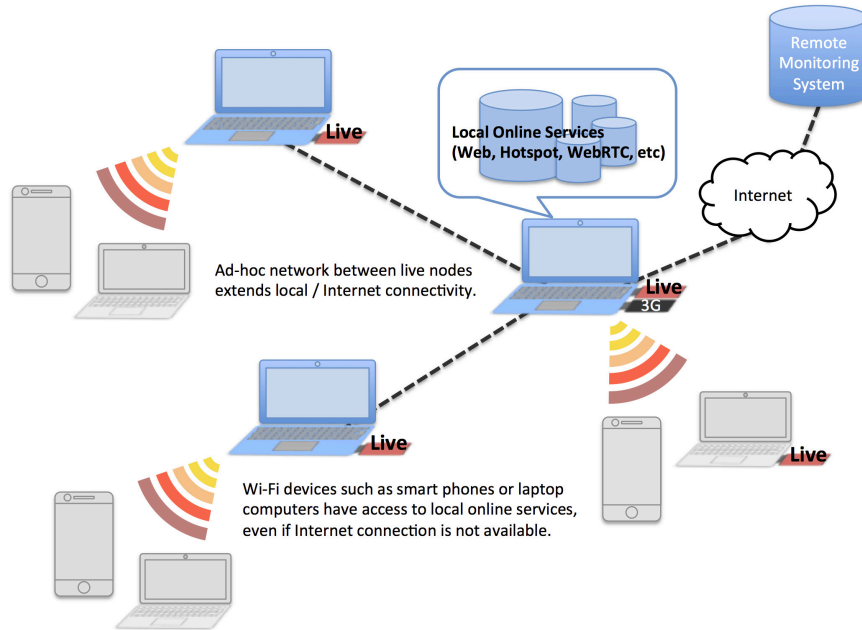


Figure 2.1: Laptop computers booting from Live USB and connecting to each other forming ad-hoc network.

Multiple meshes can be created by using multiple Wi-Fi NICs on same Live USB. On some of the laptops, USB wireless cards can be used to create Wi-Fi hotspot, so that smartphones, laptops, sensors and any other Wi-Fi devices can also connect to the network. This network can be extended by simply booting a laptop using Live USB and plugging-in USB Wi-Fi NIC. The choice of wireless over wired network was made for the following reasons.

- Preparing LAN cable in itself is a time consuming process. Moreover, deploying LAN cables takes time and careful planning.
- In evacuation shelters, it is not realistic to deploy LAN cables for networking. If LAN cables are deployed on the floor, people may fall down.
- Making Internet connection to be sharable is important and wireless networks are easier to extend as compared to wired network.

2.2 Wireless Ad-hoc Networking in Post-disaster Situations

Wireless ad-hoc network is an infrastructureless, decentralized network that does not require central Access Point to transmit data between two hosts. It is quickly deployable, unplanned and hence, is useful in situations of public emergency like post-disaster recovery. Since there is no single node acting as master in an ad-hoc network, each node is responsible to route data from source to destination. The advantage of wireless ad-hoc networks is that they are inexpensive compared to wired network and easily expandable. However, they suffer from all the traditional wireless network problems, such as significant packet loss, hidden and exposed terminal problem. Interference between adjacent nodes does not help the situation either. Moreover, since wireless nodes are mobile, the routing table has to be updated regularly. Based on how it is implemented, there are two approaches for creating wireless ad-hoc network.

2.2.1 Single Channel Scenario

In Single Channel implementation of wireless ad-hoc network, all the nodes are configured in same frequency channel as shown in Figure 2.2. Thus, nodes that are in transmission range are at one-hop distance from each other.

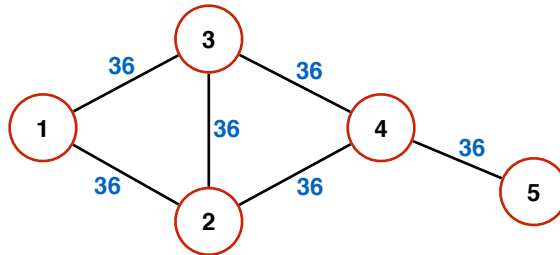


Figure 2.2: Ad-hoc Networking using Single Channel Scenario.

Pros:

- **Easy setup:** This approach is very simple to implement in Live USBs and requires minimal engineering effort. Since all the mesh nodes join single logical subnet, IP addressing can be easily handled.
- **Redundant paths:** Since multiple Live USB nodes are in transmission range of each other, redundant paths exist. If a Live USB node goes down, communication does not cease to exist.

Cons:

- Since all the nodes are configured in same channel, the network quality suffers due to interference.
- The broadcast packet; e.g., ARP and DHCP; from one Live USB node will reach all other Live USB nodes, thus consuming significant amount of resources.
- Hidden terminal and exposed terminal issues.

2.2.2 Multi Channel Scenario

In Multi Channel Scenario, only adjacent nodes are configured in the same channel as shown in Figure 2.3. In this approach, instead of a single mesh, multiple meshes will be formed. Communication between different meshes can be done using mesh routers and mesh gateways.

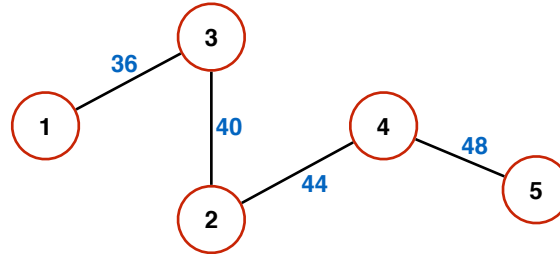


Figure 2.3: Ad-hoc Networking using hop-by-hop scenario.

Pros:

- **Less interference:** Since a limited no. of clients are configured in same channel, interference decreases by significant amount.
- **Limited Broadcast Domain:** The broadcast traffic from one Live USB node will be limited to the nodes configured in same mesh.
- **Better Performance:** Improvement in backhaul network leads to better service quality.

Cons:

- More no. of Wi-Fi NICs will be required at a Live USB node.
- IP addressing needs to be handled properly, so that each mesh segment is configured in a different subnet.
- Routing will be required to forward the traffic from one mesh to another.

Chapter 3

System Design and Architecture

3.1 Live USB Server and Client

A Live USB node can act either as a server or a client. Multiple clients can be active at a site, however, only one server can be active at a time. Server provides local online services such as web, online storage, video/audio/text communication, etc. Live USB server can have Internet connectivity, if it is available at disaster site. The user can choose to share the Internet connection with other devices in the network. Thus, server acts as a gateway to Internet for clients and user terminals. Also, server initiates wireless ad-hoc network to let other Live USB clients access the local online services and share the Internet connection.

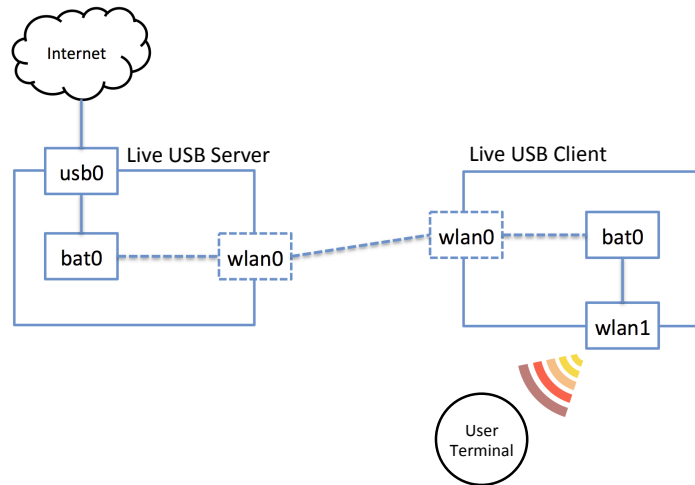


Figure 3.1: Physical connection of Live USB server, client and user terminals.

A Live USB client joins the wireless ad-hoc network in order to access the services provided by server. Also, client is responsible for creating Wi-Fi hotspot, so that user terminals e.g., smartphones, tablets, etc. can be connected. Thus, client helps in extending the connectivity from server to user terminals. Server can also start Wi-Fi hotspot, if required. The physical topology of Live USB server, client and user terminals is shown in Figure 3.1. $usbX$, $batY$ and $wlanZ$ are the various interfaces as recognised by the Operating System for carrying out networking operations.

3.2 Network Design

The following subsections discuss network design of mesh and hop-by-hop scenario.

3.2.1 IP Addressing in Single Channel Scenario

In Single Channel scenario, as discussed in Section 2.2.1, Live USB nodes enable wireless mesh network in the same channel to communicate with each other and share the local and Internet connection. As this network forms a single layer 2 domain, it is important to reduce the amount of unwanted traffic. All the Dynamic Host Configuration Protocol (DHCP) requests from Live USB clients are addressed by Live USB server. Server also acts as the Network Address Translation (NAT) router for the clients. On the other hand, Live USB client addresses DHCP requests from user terminals that are connected to it using Wi-Fi hotspot. Figure 3.2 shows the logical topology of mesh scenario. *bat0* interface of Live USB server has static IP address so that all the clients and user terminals can access services provided by the server.

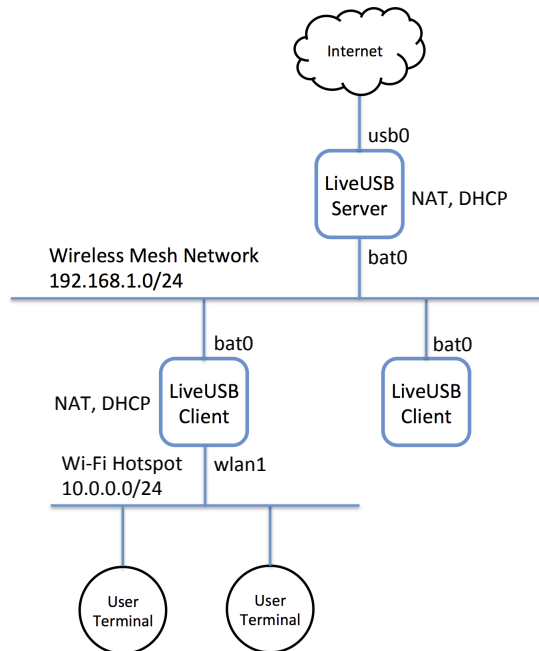


Figure 3.2: Logical Topology of Live USBs and other devices in Single Channel Scenario.

3.2.2 IP Addressing in Multi Channel Scenario

The Multi Channel scenario of wireless ad-hoc networking was discussed in Section 2.2.2. In this scenario, multiple meshes are formed, and each mesh is in different IP subnet. We provide DHCP Pool Allocator that works similar to Prefix Delegation [7] in IPv6 and makes sure that each mesh and Wi-Fi hotspot is in different subnet. DHCP Pool Allocator runs at Live USB Server and assigns a unique subnet whenever a client starts a new mesh or hotspot. Server has two types of subnets available, one for ad-hoc networking and other for Wi-Fi hotspot. We used 192.168.0.0/16 subnet for ad-hoc networking and 172.16.0.0/16 for Wi-Fi hotspot. Figure 3.3 shows the logical topology of hop-by-hop scenario.

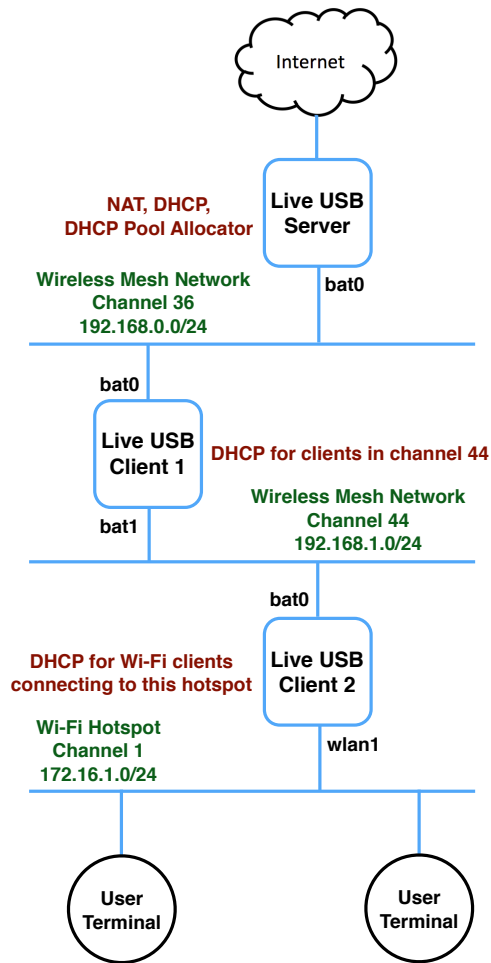


Figure 3.3: Logical Topology of Live USBs and other devices in Multi Channel Scenario.

In order to extend this kind of network, a new client should connect to one of the existing meshes and then start its own new mesh and/or Wi-Fi hotspot. This will make sure that all the clients and/or user terminals are able to reach each other. In terms of graph theory, the network topology should be such that a tree is formed, and not forests. For this purpose, three types of self-explanatory messages are supported by Live USB Client.

- Join a mesh
- Start a new mesh
- Start Wi-Fi hotspot

Figure 3.4 presents various kinds of messages exchanged between Live USB nodes. Each pair of Request and Response message has been shown in the same colour. The number on the message shows the topological ordering of the messages. This ordering does not mean that messages always follow same order, it is just shown as an example. In fact, in real time, the topological ordering of the message will rarely follow this particular order. The messages are explained below.

0. Live USB Server boots and starts mesh in default channel. Server has static IP address i.e., 192.168.0.1/24.

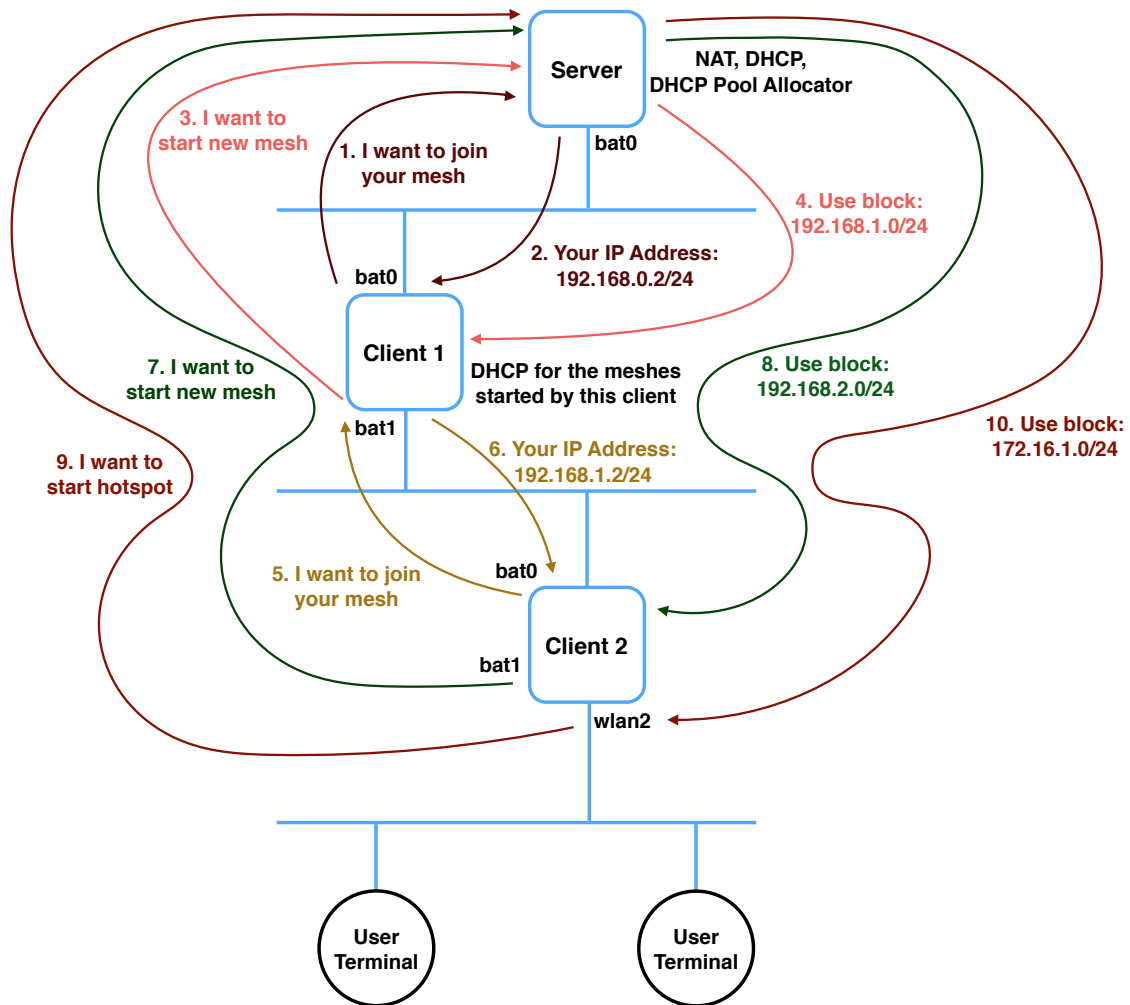


Figure 3.4: Messages Exchanged between Live USB nodes in Multi Channel Scenario.

1. Client 1 joins the mesh started by Server. It requests for IP address allocation.
2. Server assigns the IP address 192.168.0.2/24 to Client 1.
3. Client 1 wants to start a new mesh. It requests Server to allocate a free subnet for the mesh.
4. Server allocates 192.168.1.0/24 subnet to Client 1 and removes it from the list of free subnets. Client 1 uses first available IP address of this subnet to add to its own mesh interface and adds this subnet to DHCP list for other nodes joining this mesh. Also, this subnet is added to the routes advertised by routing protocol so that other clients can know how to reach this subnet.
5. Client 2 joins the mesh created by Client 1. It requests for IP address allocation.
6. Client 1 allocates the IP address 192.168.1.2/24 to Client 2.
7. Client 2 wants to start a new mesh. It requests Server to allocate a free subnet for the mesh. This request reaches Server through Client 1.
8. Server allocates 192.168.2.0/24 subnet to Client 2 and removes it from the list of free subnets. Client 2 uses first available IP address of this subnet to add to its own mesh interface.

and adds this subnet to DHCP list for other nodes joining this mesh. The subnet is also added to the routes advertised by routing protocol.

9. Client 2 wants to start a Wi-Fi hotspot. It requests Server to allocate a free subnet for the hotspot. This request reaches Server through Client 1.
10. Server allocates 172.16.1.0/24 subnet to Client 2 and removes it from the list of free subnets. Client 2 uses first available IP address of this subnet to add to its own Wi-Fi hotspot serving interface and adds this subnet to DHCP list for user terminals connecting to this hotspot. Routing protocol will start advertising this subnet.

3.2.3 IP Addressing for Wi-Fi Hotspot

Addressing for Wi-Fi hotspot can be done in multiple ways as described below.

1. Use same subnet for Wi-Fi hotspot on each of the Live USB node by enabling NAT on interface that starts Wi-Fi hotspot. This will enable us to use same subnet for all the Wi-Fi hotspots served by different Live USB nodes but, clients connected to Wi-Fi hotspot served by different Live USB nodes will not be able to connect to each other directly.
2. Relay the DHCP request from end user to Live USB server. In Single Channel scenario, this will further expand the broadcast domain of the network. In Multi Channel scenario, this process will take time since DHCP Relay packet might have to travel multiple hops before it finally reaches the server.
3. Whenever a Client wishes to start Wi-Fi hotspot, it requests Sever to allocate a subnet. Once allotted, this subnet can be used for Wi-Fi hotspot and all the DHCP requests can be handled by Client itself. This process was explained in detail in Section 3.2.2.

3.2.4 Routing in Wireless Ad-hoc Network

Routing is not required in Single Channel scenario, since it forms single Layer 3 domain. On the other hand, packets should be routed between different subnets in Multi Channel scenario. There are various protocols that use different strategies for forwarding data in an ad-hoc network. B.A.T.M.A.N. Advanced (a.k.a. batman-adv) [8] and OLSR [9] are two popular mobile ad-hoc implementations. While OLSR operates at layer 3, batman-adv operates entirely at ISO/OSI layer 2. Thus, OLSR can exchange information about different subnets associated with a client. In OLSR terminology, this is called HNA (Host to Network Association). However, since batman-adv does not care about layer 3 information, user should explicitly run routing protocol e.g. RIP, IGRP, etc. on top of batman-adv so that information about different subnets can be shared with all the clients in the network. Other solution might be to add static routes for each subnet in each of the Live USB node. This solution is feasible only if network is small. As the size of the network grows, routing protocol will be required to add the routes automatically.

3.3 Local Online Services

To immediately enable ad-hoc communication and information exchange among users, Live USB server provides local online services that are pre-configured and ready to be deployed in a disaster affected site. Instead of installing variety of applications on each Live USB individually, our system integrates most of the applications on Live USB server that can be accessed by clients and other devices. The following two features introduce a lot of benefits that make ICT service and environment to be prepared for instantaneous information exchange.

- It reduces the maintenance cost of individual software client.
- By locally hosting services at Live USB server, these applications are "Internet-free" or offline at the moment of deployment. It will also save bandwidth.

Once the Internet connection becomes available, Live USB nodes and other user terminals can switch to "Internet-dependent" which is online mode.

3.4 Web Based Communication and Information Exchange

Web-based content management systems like MediaWiki [10] (Figure 3.5) and WordPress [11] help to facilitate local information exchange. All these systems can be accessed via web browser and require single click operation. All the links are provided on WordPress homepage, see Figure 3.6, so that end-user does not have to remember all the links individually.

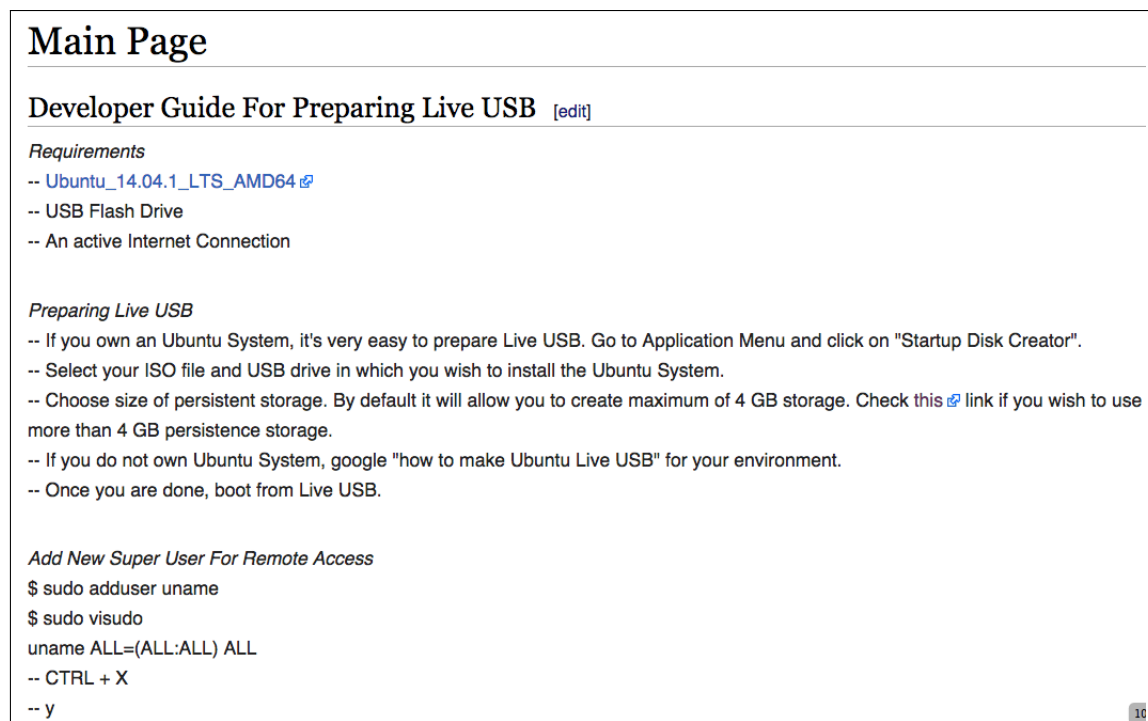


Figure 3.5: MediaWiki page on Live USB Server.

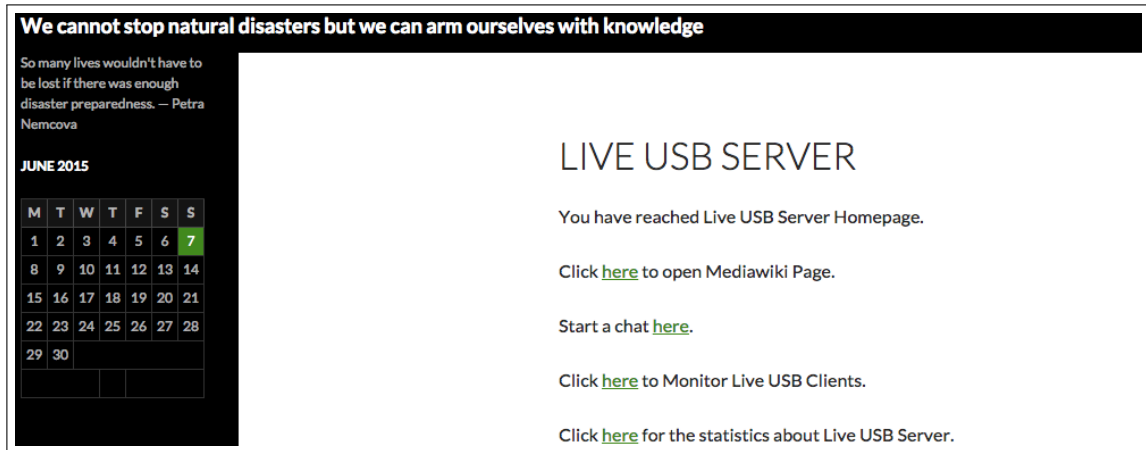


Figure 3.6: WordPress UI on Live USB Server.

This system can also be used to enable text, audio and video chat using WebRTC [12] in the browser itself without needing to install any extra packages in Live USB. WebRTC is a JavaScript API that enables Peer-to-Peer (P2P) connection between user's browsers. In case P2P connection is not successful (mainly due to NATs and/or firewalls), it falls back to Server-Client model by relaying data through some server in global Internet [13].

3.5 Remote Troubleshooting

Virtual Private Network (VPN) server is a system present in global Internet to which Live USB server tries to connect to. Once connection is successful, technical issues with Live USB server can be taken care of by support staff from a remote place. This allows the technical team to stay at any place instead of having them be physically present at the disaster-affected site. Moreover, if there are some critical issues in the Live USB server, then its data can be evacuated to the VPN server and a new Live USB server flash drive can be used. VPN can also be used to collect some statistics from the Live USB server such as how many clients are connected to it, currently. It helps to monitor the Live USB server so that it can continue to function as expected. Since the Live USB server knows about all the clients in the network, technical issues with Live USB clients can also be taken care of by VPN connection through the Live USB server. Of course, VPN connection will be successful only when Internet connection is made available to the Live USB server.

3.6 System Monitor and Health Check

Each Live USB node keeps track of its own latest information such as memory, CPU, disk, I/O, swap, logged-in accounts, common applications, network quality, etc. Linux Dash [14] is a simple web-based dashboard that provides these statistics (Figure 3.7). The Live USB server provides a link to each of the clients so that each client's latest statistics can be monitored individually through the server. This way, the end-user does not have to reach out to each Live USB client to monitor it. On the other hand, the VPN server keeps track of all the Live USB servers that are connected to it.

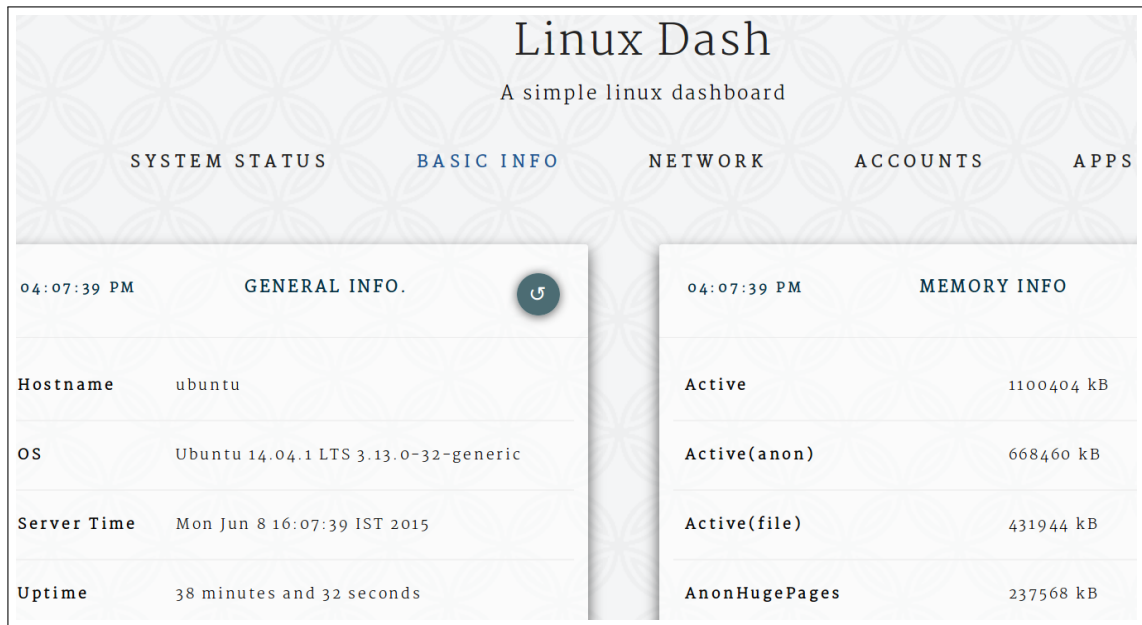


Figure 3.7: Linux Dash showing various statistics of Live USB Server.

The script at the VPN server automatically adds an entry for that particular Live USB server in SmokePing [15] Targets (Figure 3.8). Thus, each Live USB server can be monitored individually by VPN server. We send only minimal data to the VPN server to avoid unnecessary consumption of Internet bandwidth.

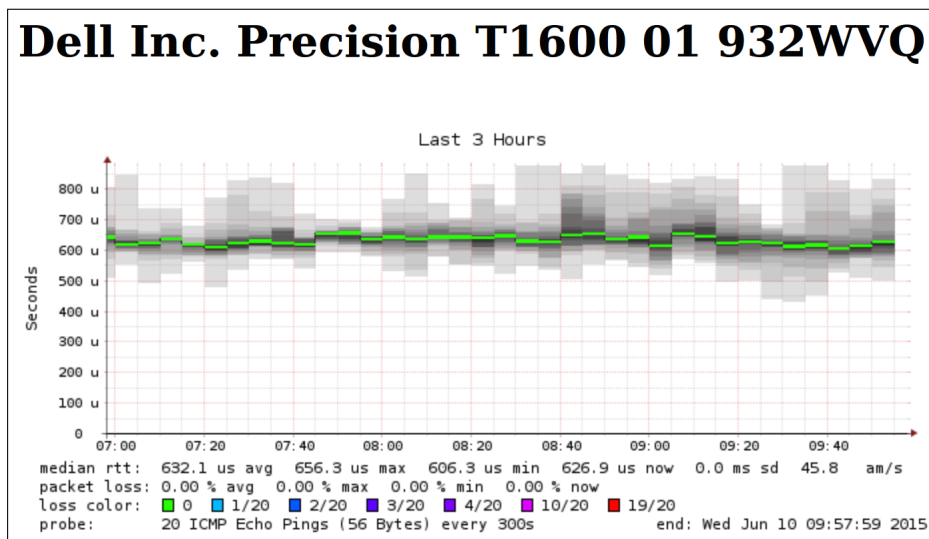


Figure 3.8: Once Live USB server successfully connects to VPN server, it's entry gets automatically added as a monitoring Target in SmokePing.

Chapter 4

System Implementation

4.1 Hardware and Software Details

Server provides various services that are not required to be present on client. In order to reduce the size of client OS image, we prepare separate guest OS images for implementing server and client mode of Live USB. Table 4.1 summarizes the implementation details of the system.

Table 4.1: Implementation Details of Live USB Server and Client

Feature	Live USB	
<i>Hardware</i>	SanDisk Extreme USB 3.0 32 GB USB Flash Drive	
<i>Guest OS</i>	Linux Ubuntu 14.04.1 LTS	
<i>Persistent Storage</i>	27 GB	
<i>Mesh Networking</i>	batman-adv-2014.3.0 [16]	
<i>Routing</i>	RIPv2 implementation of Quagga [17]	
<i>Wi-Fi Hotspot</i>	hostapd [18] and dnsmasq [19]	
	Live USB Client	Live USB Server
<i>Web Proxy</i>	-	Squid3 [20]
<i>Web Based Content Management Systems</i>	-	MediaWiki 1.25.1 [21] WordPress 4.2.2 [22]
<i>Monitoring and Health Check</i>	Live USB clients that are connected to the server can be monitored at the server.	Live USB servers can be monitored at Remote VPN server when server can access Internet.

4.2 Creating an Ad-hoc Network

There are various open-source projects that enable mesh networking on the NIC, and many of them are available in Ubuntu repository as well [23, 24]. We used batman-adv in this work for three reasons:

- It is open-source project and available in Ubuntu repository.
- The source code is well-maintained and regularly updated.
- The work is properly documented and has a good support-base.

4.2.1 Configuring Wi-Fi Interface in Ad-hoc Mode and Starting Mesh

Not all versions of batman-adv are supported on each version of Ubuntu, so one has to make sure to choose the version supported by his/her version of Ubuntu. The best way would be to install Synaptic package manager and download batman-adv using Synaptic. Once installed, following script puts Wi-Fi card in ad-hoc mode and starts mesh network.

```
#!/bin/bash
# Stop Network Manager
sudo service network-manager stop

# Increase MTU since batman-adv adds its own 32 bytes header to the frame
sudo ifconfig wlan0 down
sudo ifconfig wlan0 mtu 1532

# Configure interface in ad-hoc mode
sudo iwconfig wlan0 enc off
sudo iwconfig wlan0 mode ad-hoc essid MyMeshNetwork ap 02:12:34:56:78:9A channel 1

# Load batman-adv module and add interface
sudo modprobe batman-adv
sudo batctl if add wlan0
sudo ifconfig wlan0 up
```

In this script, *wlan0* is the interface on which we wish to start the ad-hoc network, and *bat0* is the virtual interface created by batman-adv. To configure these parameters, have a look at "iwconfig" command.

4.2.2 IP address Assignment

Since batman-adv operates at layer 2 (Data Link Layer), all the communication can be done using MAC addresses. In order to add layer 3 functionalities in this network, IP address should be assigned to *bat0* interface.

```
$ sudo ifconfig bat0 192.168.0.1/24 up
```

This network can be very easily extended by running batman-adv on some neighbor laptop computers. IP address can be assigned to *bat0* interface of each of the laptop manually. The better approach would be to run DHCP server on one of the systems and let other laptop computers obtain IP address from that DHCP server. DHCP server should be listening on *wlan0*, the interface on which

mesh was started. Following command will obtain an IP address from DHCP server automatically.

```
$ sudo dhclient bat0
```

The detailed configuration for batman-adv is available at [25]. Tcpdump or Wireshark [26] on *wlan0* interface can be used to make sure that batman-adv packets are being advertised. A screenshot of Wireshark capture of Batman-adv frame is shown in Figure 4.1.

```
$ sudo tcpdump -ni wlan0
```

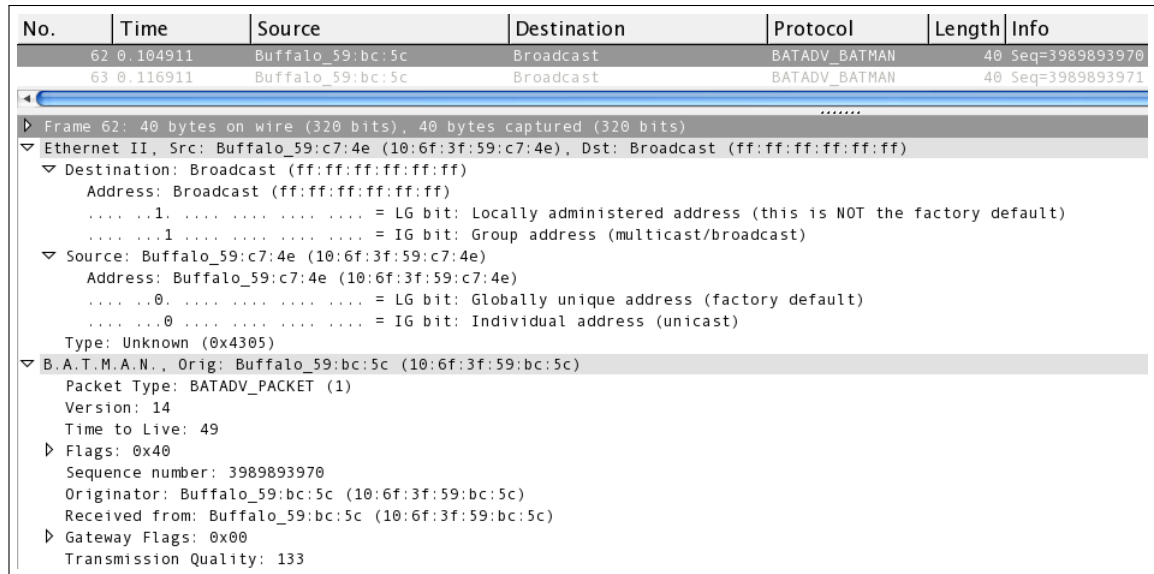


Figure 4.1: Wireshark capture of batman-adv frame.

4.3 Wi-Fi Hotspot

4.3.1 Configuration for DHCP Server

All the configuration of dnsmasq is present in “/etc/dnsmasq.conf” file. Appending these lines to the file is all the configuration required for DHCP server.

```
# disables dnsmasq reading any other files like /etc/resolv.conf for nameservers
no-resolv
# Interface to bind to
interface=wlan1
# Specify starting-range,end-range,lease-time
dhcp-range=10.0.0.30,10.0.0.200,4h
# DNS addresses to send to the clients
server=8.8.8.8
server=8.8.4.4
```

4.3.2 Configuration for Hotspot

The configuration for hostapd has to be supplied in a file.

```
interface=wlan1
driver=nl80211
ssid=00000INDIA
hw_mode=g
channel=6
```

This configuration will make the Wi-Fi hotspot open to everyone without any passphrase. For security, following configuration can be used.

```
interface=wlan1
driver=nl80211
ssid=00000INDIA
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=3
wpa_passphrase=helloworld
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

4.3.3 Starting the Hotspot

Following script can be used to start Wi-Fi hotspot on *wlan1* interface.

```
#!/bin/bash
# Pre-Requisites for hostapd
ifconfig wlan1 10.0.0.1/24 up
sleep 1
service hostapd stop
# Restart dnsmasq
killall dnsmasq
dnsmasq
# Enable IPv4 forwarding
sysctl -w net.ipv4.ip_forward=1
# Start hostapd
hostapd hostapd-test.conf 1> /dev/null
```

The detailed procedure for creating Wi-Fi hotspot on Linux systems is available at [27].

4.4 Sharing Internet between two interfaces

Internet can be easily shared from LAN or 3G connection over Wi-Fi interface. For sharing mobile data connection, USB tethering has to be enabled and smartphone should be connected to laptop using data cable. Generally, ethernet and USB interfaces will be shown as *ethX* and *usbY*, respectively in “ifconfig” command.

4.4.1 Obtaining IP Address from DHCP

This step is to make sure that we have an active Internet connection. If it is not working, the most probable cause might be that IP address is not assigned to the interface. This situation will arise if Network Manager is not running. Assuming we are using mobile data connection for Internet access and *usb0* is the USB tethered interface, run following command to obtain IP address using DHCP.

```
$ sudo dhclient usb0
```

4.4.2 Setting up and down interfaces for Internet sharing

This script can be used to share Internet connection from *usb0* interface to *bat0* interface using NAT.

```
#!/bin/bash

# Enable IPv4 traffic forwarding
sysctl -w net.ipv4.ip_forward=1

# Add iptables rules for NAT interfaces
/sbin/iptables -t nat -A POSTROUTING -o usb0 -j MASQUERADE
/sbin/iptables -A FORWARD -i usb0 -o bat0 -m state --state RELATED,ESTABLISHED \
-j ACCEPT
/sbin/iptables -A FORWARD -i bat0 -o usb0 -j ACCEPT
```

Following script can be used to clear routes from iptables.

```
#!/bin/bash
/sbin/iptables --flush
/sbin/iptables --table nat --flush
/sbin/iptables --delete-chain
/sbin/iptables --table nat --delete-chain
```

4.4.3 Setting up Squid as Transparent Proxy

Squid3 configuration is present in “/etc/squid3/squid.conf” file. Squid can be configured as transparent proxy so that user need not manually configure proxy at his side. To setup squid, following

directives must be added or modified.

```
http_port 3128 intercept

acl localnet src 10.0.0.0/8          # RFC1918 possible internal network
acl localnet src 172.16.0.0/12      # RFC1918 possible internal network
acl localnet src 192.168.0.0/16     # RFC1918 possible internal network
acl localnet src fc00::/7          # RFC 4193 local private network range
acl localnet src fe80::/10         # RFC 4291 link-local (directly plugged) machines
```

4.4.4 Forwarding HTTP traffic to Squid Proxy

Following commands will forward all the traffic, destined to port 80, to the port on which Squid is running (i.e., 3128). This value should be changed if proxy is running on some other port.

```
$ /sbin/iptables -t nat -A PREROUTING -i bat0 -p tcp --dport 80 -j REDIRECT \
--to-port 3128
$ /sbin/iptables -t filter -A INPUT -p tcp --dport 3128 -j ACCEPT
```

Restart Squid and networking process to make the changes.

```
$ sudo service squid3 restart
$ sudo service networking restart
```

If everything is working fine, access logs should be present in “/var/log/squid3/access.log” file.

```
$ sudo tail -30 /var/log/squid3/access.log
```

4.4.5 Routing Using RIPv2

For sharing routing information among clients, we need to enable zebra and rip daemons in “/etc/quagga/daemons” file.

```
zebra=yes
ripd=yes
```

All the RIP configuration can be specified in “/etc/quagga/ripd.conf” file. If we specify the interface name, ripd will automatically share the information about subnet on that particular interface.

```
password helloworld
router rip
  version 2
  network bat0
  network wlan1
line vty
```

Restart the quagga service for the changes to take effect.

```
$ sudo service /etc/init.d/quagga restart
```

RIP routes can be checked by telnet on localhost port 2602.

```
Router> en  
Router# conf t  
Router(config)# show ip rip
```

4.5 GUI

For using Live USBs, user need not remember these commands and/or run scripts since Graphical User Interface is provided for all the technical work. User Interface (UI) has been designed to support beginners as well as expert users. Following subsections describe basic functionality of Server and Client GUI.

4.5.1 Server GUI

The main window of Server UI is shown in Figure 4.2. Server has basically three functions: Starting a mesh, Starting Wi-Fi Hotspot, and Sharing Internet connection.

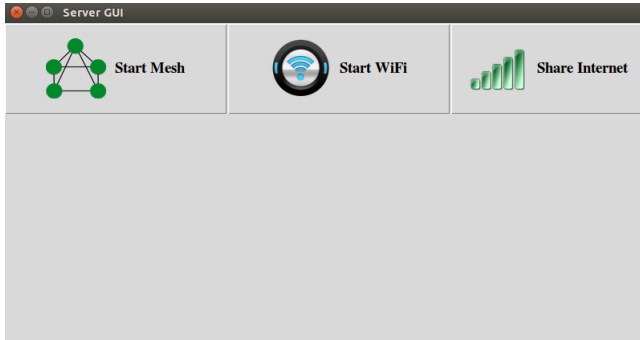


Figure 4.2: Server UI: Main Window.

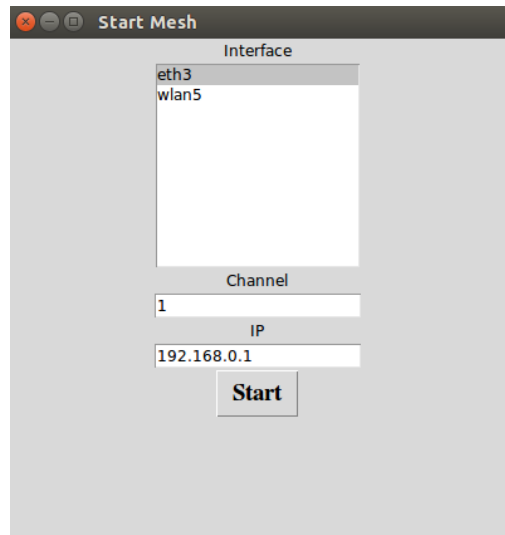


Figure 4.3: Server UI: Start Mesh Window.

Once *Start Mesh* button is pressed, user can select the interface on which to start the mesh, channel and IP address to be given to that interface as shown in Figure 4.3. Default values are provided for everything so that even a novice can use this system. Similar choices are provided for *Start Wi-Fi* (Figure 4.4) and *Share Internet* (Figure 4.5) as well.

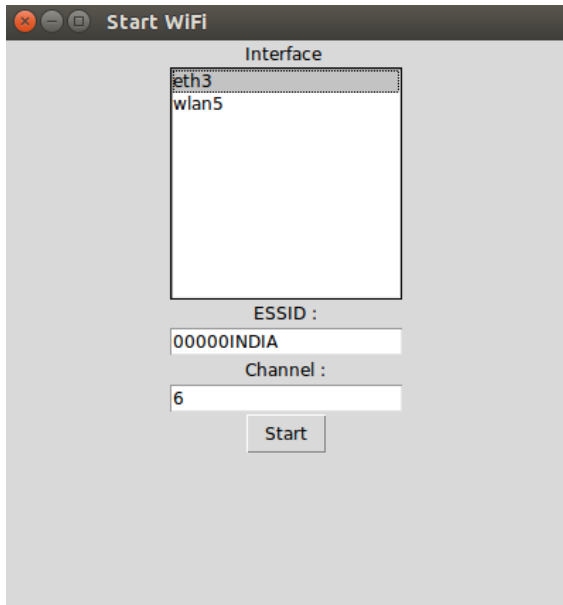


Figure 4.4: Server UI: Start WiFi Window.

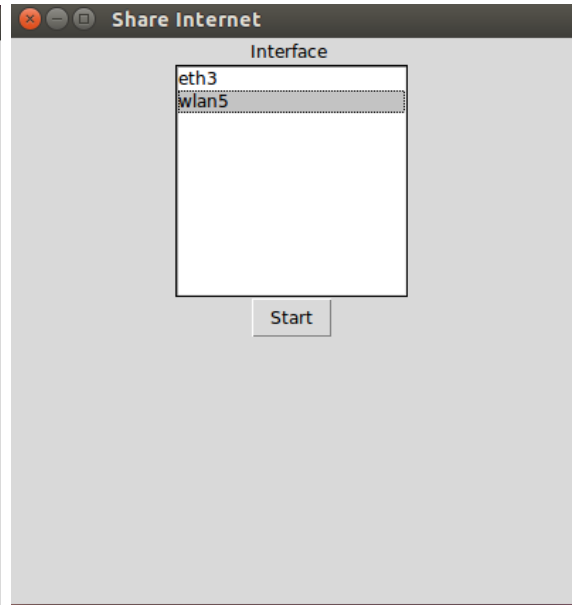


Figure 4.5: Server UI: Share Internet Window.

4.5.2 Client GUI

Client's UI has been designed in such a manner that supports both scenarios of ad-hoc networking i.e., Single Channel and Multi Channel. Figure 4.6 presents the main window of Client UI. Client's functionalities are categorized into Joining a mesh, Starting a mesh, Starting Wi-Fi Hotspot and Disconnecting a mesh.

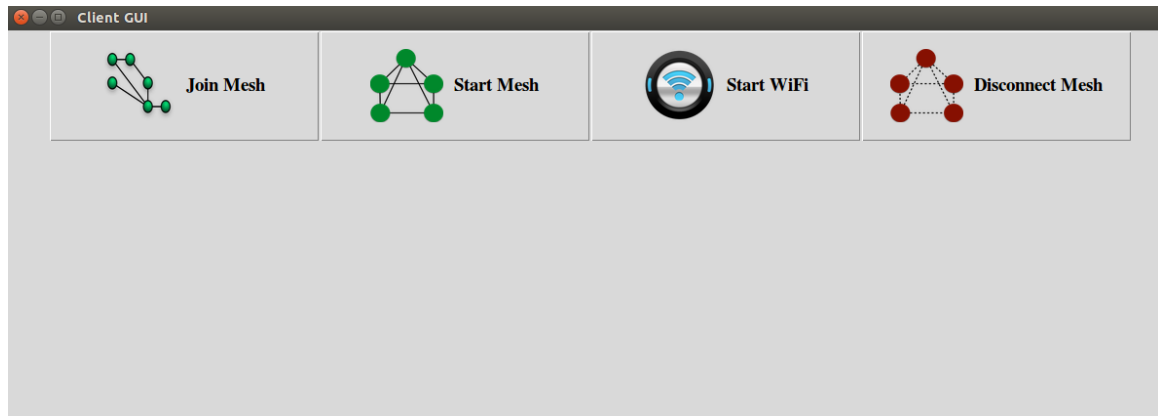


Figure 4.6: Client UI Main Window.

For all the functions to work, client should join a mesh first. Without joining a mesh, no other function will work. Once *Join Mesh* button is pressed, the list of available interfaces is presented. Once user selects the interface, a scan is performed for available ad-hoc networks. Selecting an ad-hoc gives details about that network, such as channel, cell, essid, etc. All these steps are depicted in Figure 4.7.

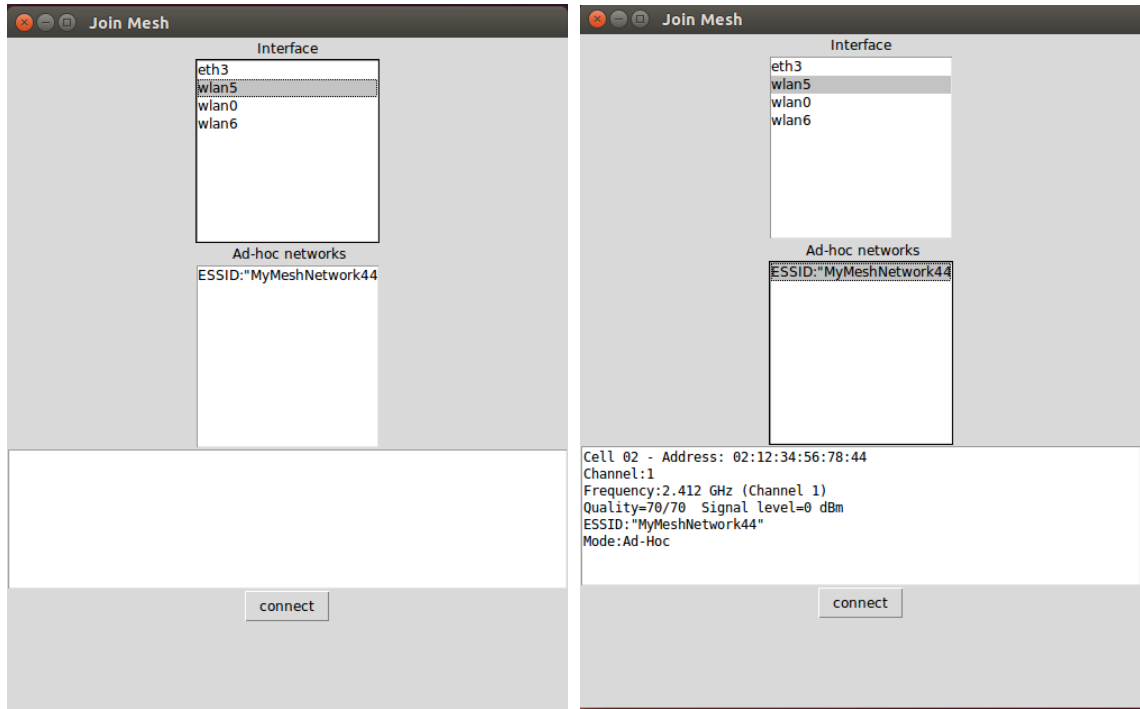


Figure 4.7: Client UI: Join Mesh Window.

Start Mesh and *Start WiFi* work same as that of Server. The only difference being that whenever a new mesh or Wi-Fi hotspot is started by user, Client requests Server to assign a unique subnet. Figure 4.8 shows what happens when user tries to start a mesh without joining any mesh. This error comes due to the fact that Client requests Server to allocate a new subnet for each new mesh or Wi-Fi hotspot that is started by the Client. If Client does not join any mesh, it will not be able to contact Server.

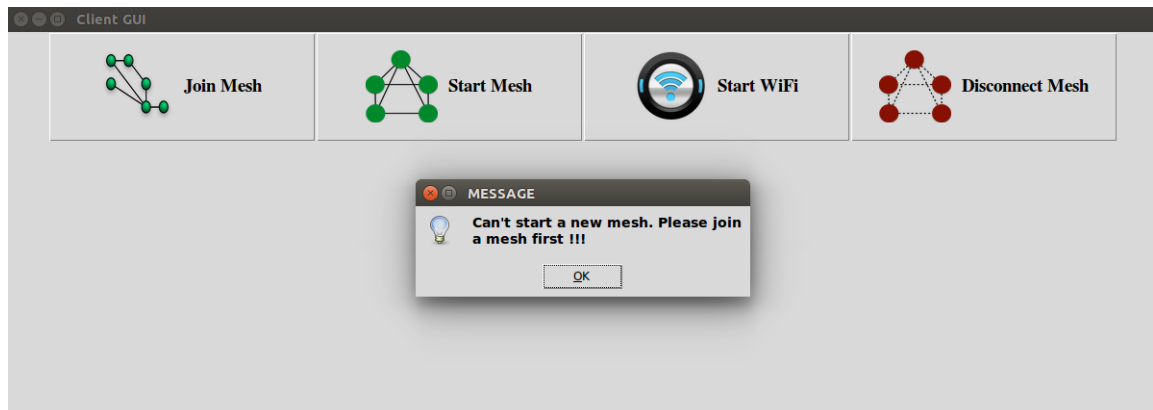


Figure 4.8: Client UI shows an error when user starts a new mesh without joining existing mesh.

If none of the Client starts a new mesh, and all of them join the mesh started by Server, this would form Single Channel scenario of ad-hoc networking. On the other hand, if any one of the Client starts a new mesh, it would form Multi Channel scenario. Client maintains a list of all the meshes i.e., joined and started by the Client. When user clicks on *Disconnect Mesh* button, this

information is presented to the user. User can then select the mesh to disconnect from, and click on Disconnect button to leave the mesh as shown in Figure 4.9.

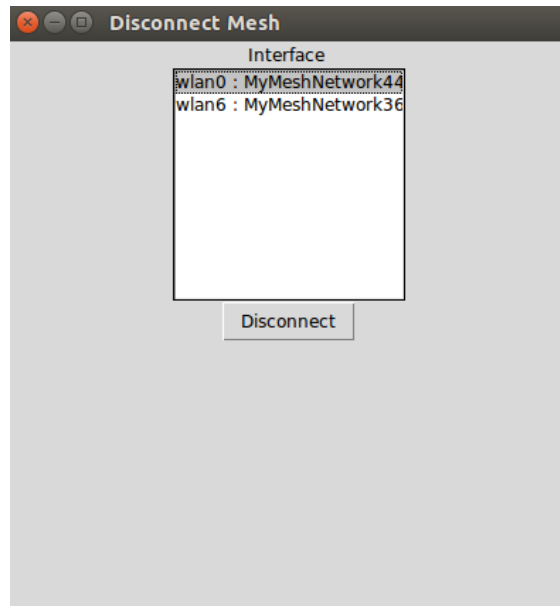


Figure 4.9: Client UI: Disconnect Mesh Window.

Figure 4.10 shows the error that occurs if user clicks on *Disconnect Mesh* button when Client is not connected to any mesh.

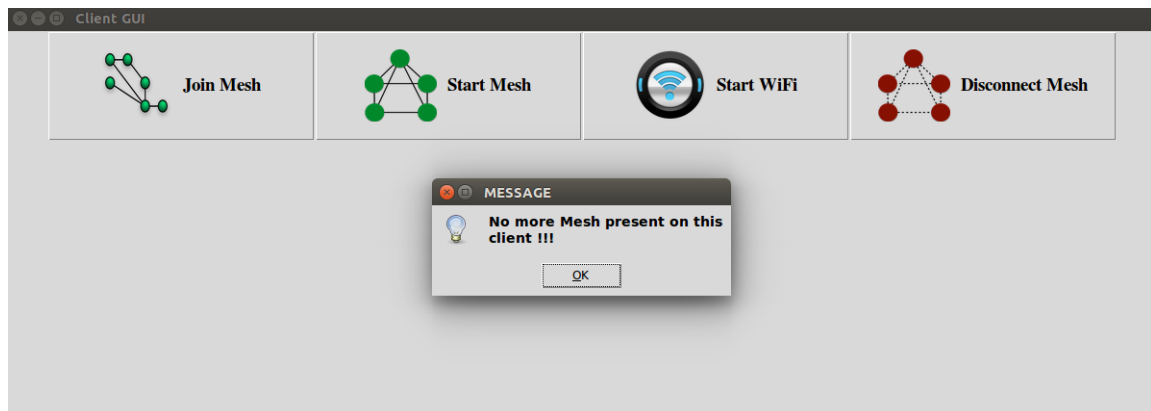


Figure 4.10: Client UI shows an error if user clicks on Disconnect Mesh button when no mesh is present on Client.

4.6 Handling the Disconnections

Our system has been designed to handle disconnections in a systematic manner. If the disconnection occurs for some reason, Live USB clients within each subsystem still continue to talk to each other. For instance, consider the initial topology as shown in Figure 4.11. Server and Client 1 are part of one mesh and Client 1 and Client 2 form another mesh. Each subnet is shown in different colour so that they can be easily differentiated. Let after sometime, Client 1 and Client 2 move further

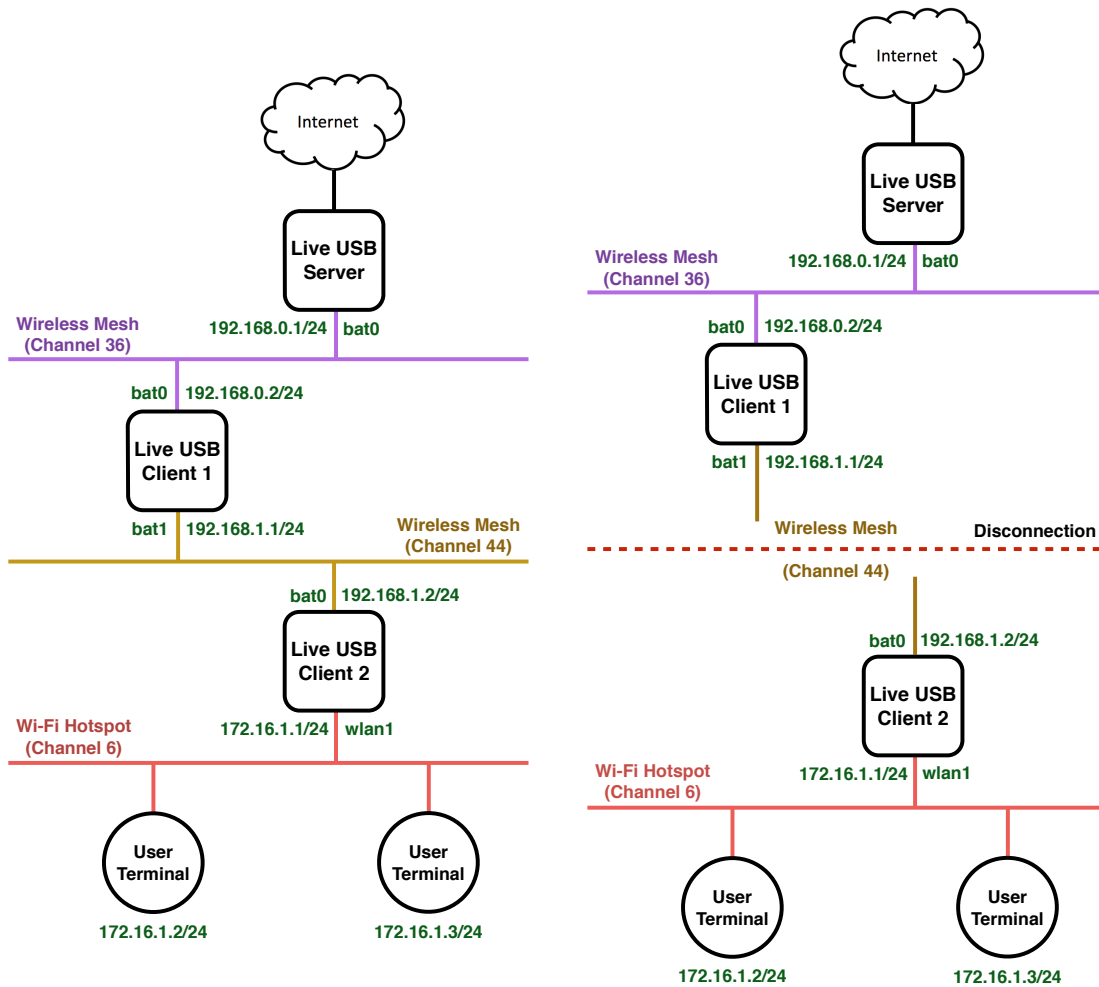


Figure 4.11: Initial topology of Live USB nodes. **Figure 4.12:** Topology after disconnection occurs.

apart such that they are not in the transmission range of each other. Dashed line in the Figure 4.12 represents the disconnection. The radio interfaces of both the nodes are still configured in ad-hoc mode and they continue to advertise about their existence in the network, but both of them won't receive each other's packets. In such a scenario, user terminals connected to Wi-Fi hotspot served by Client 2 will be able to reach Client 2, Server can reach Client 1 and vice-versa. In other words, internal network within each of the subsystem still continues to function, but obviously connection across subsystems won't work. After sometime, if some intermediate node e.g., Client 3 appears between Client 1 and Client 2 satisfying following criteria:

- At least one of its radio interface is configured in the same frequency channel as that of common channel between Client 1 and Client 2.
- Client 3 is positioned in such a fashion that it is in transmission range of both Client 1 and Client 2.

If both of these conditions hold, then communication between Client 1 and Client 2 will start immediately without requiring any manual operation on user's part. This situation is shown in Figure 4.13. Dotted line between Client 1 and Client 2 means that they both are not in transmission

range of each other, but they can still communicate via Client 3, since all 3 of them are configured in the same frequency channel.

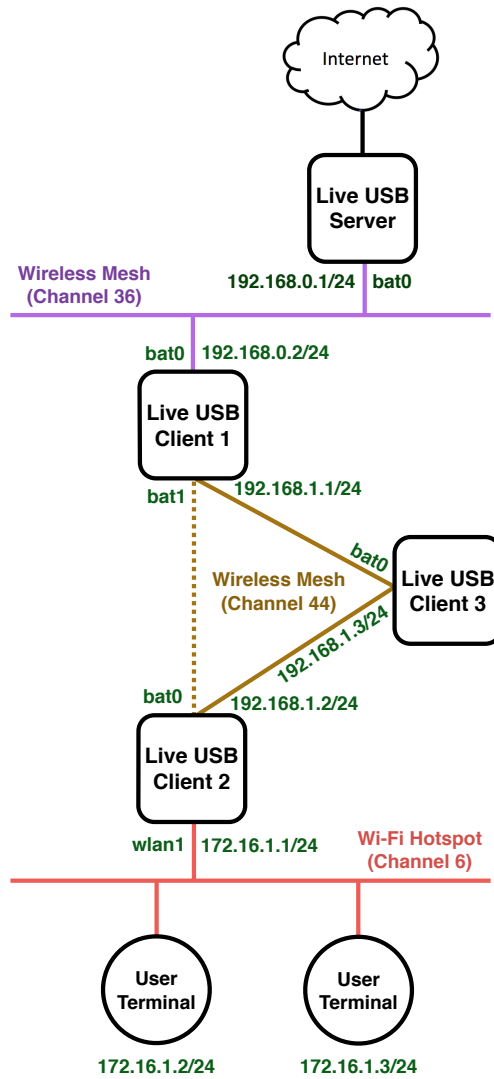


Figure 4.13: Topology after new node comes between disconnected nodes.

Chapter 5

Evaluation

5.1 Tested Hardware

We have confirmed that hardware summarized in Table 5.1 supports wireless ad-hoc networking using batman-adv and Wi-Fi hotspot using hostapd with dnsmasq as DHCP server on Ubuntu 14.04.1 LTS.

Table 5.1: Verified Hardware and Drivers

Laptop Model	Internal Wi-Fi Model
Acer Aspire 5750G	Atheros AR5B97 802.11b/g/n
Sony VAIO E Series	Atheros AR9285 VPCEB14EN 802.11b/g/n
HP Pavilion G6 1200tx	Broadcom 4313GN 802.11b/g/n
Wireless LAN Adapter Model	Drivers Used
Logitech LAN-W150N/U2	rt2800usb
Buffalo WLI-UC-AG300N	rt2800usb

5.2 Phase 1: Single Channel Scenario Using Batman-adv

This section summarises the results of field trials from Single Channel scenario of ad-hoc networking. Following is a summary of configuration for this setup:

- **MTU on wireless interface:** 1532 bytes.
- **Encryption on wireless interface:** Off
- **Wireless mode:** Ad-hoc
- **MAC Protocol:** IEEE 802.11a/b/g/n
- **Frequency:** 5.18 GHz
- **Channel:** 36

All other parameters were kept unchanged.

In order to verify the performance of Single Channel scenario in multi-hop setup, we created a mesh of 4 laptop computers. In this setup, the transmission power of Wi-Fi NIC was manually managed to make sure that only adjacent laptops were in direct line-of-sight of each other as shown in Figure 5.1.



Figure 5.1: Field Setup for Multi-hop Single Channel Scenario.

This test was conducted in hostel premises, and there was interference from Access Points. Also, a lot of people were moving around, blocking the signal between laptop computers. Iperf [28] server was enabled on Laptop 1 and Iperf client was used to test the throughput from each of the subsequent laptop to Laptop 1. TCP throughput was averaged over 3 runs, while in case of UDP, we consider the minimum, maximum and average throughput by changing the rate of data transfer. Figure 5.2 presents the results of this setup.

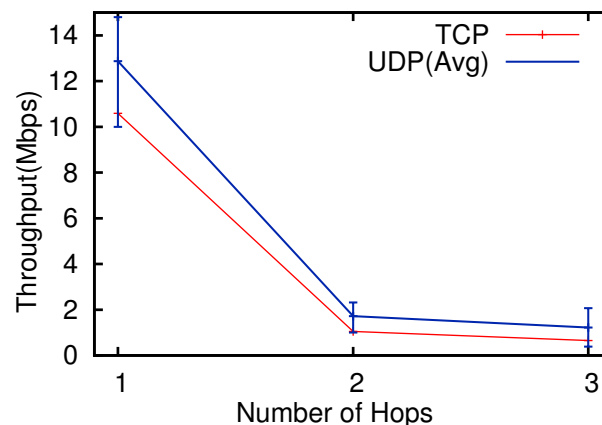


Figure 5.2: Results of Multi-hop setup in Single Channel Scenario of Phase 1 Testing.

5.3 Phase 2: Single Channel vs Multi Channel Batman-adv vs OLSR

In this phase of testing, we tested both Single Channel and Multi Channel scenarios using batman-adv as well as OLSR. We used 3 desktop systems and 1 Laptop computer and placed them at various places in IITH campus to conduct this test. The locations of computers is SSRL-2, Server Room, Store Room and Ph.D. Faculty Cabin. The node in SSRL-2 was booted using Live USB Server and the others were booted using Live USB Clients. All these systems are represented as S, C1, C2 and C3, respectively. The logical topology for Single Channel and Multi Channel setup are shown in Figure 5.3 and 5.4, respectively. We observed that C2 (Laptop in Store Room) was not able to receive any signal from S or C1 even after using maximum transmission power on USB Wi-Fi card i.e., 20 dBm. So, we used USB Extension Cable to attach a Wi-Fi card that could receive the signal from S and C1. For conducting the tests, Iperf server was enabled on S and Iperf client was run from C1, C2 and C3 to S. All the values were averaged over 3 runs.

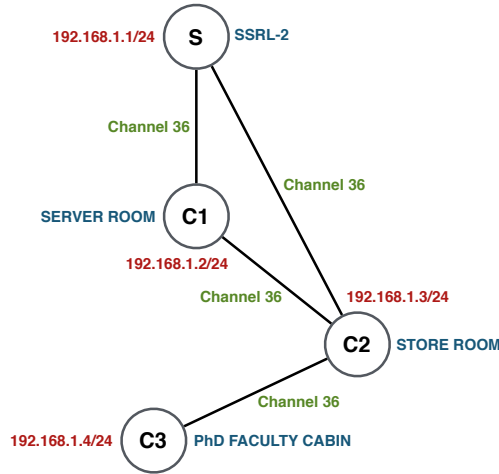


Figure 5.3: Logical Topology for Single Channel Scenario of Phase 2 Testing.

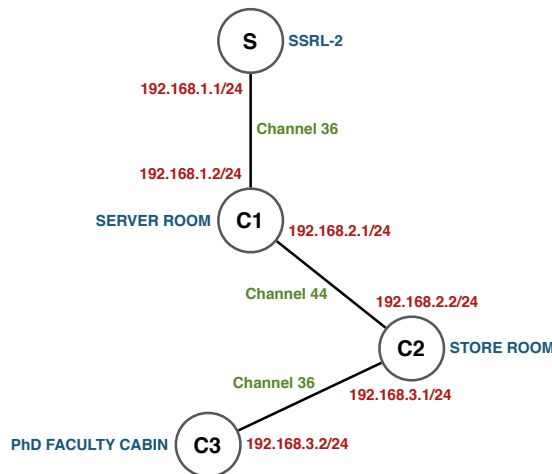


Figure 5.4: Logical Topology for Multi Channel Scenario of Phase 2 Testing.

5.3.1 UDP Throughput

Figure 5.5 shows the error graph for minimum, maximum and average UDP throughput values from C1, C2 and C3 to S satisfying two conditions:

Average Jitter over 3 runs < 3ms

Average Packet Loss over 3 runs < 2%

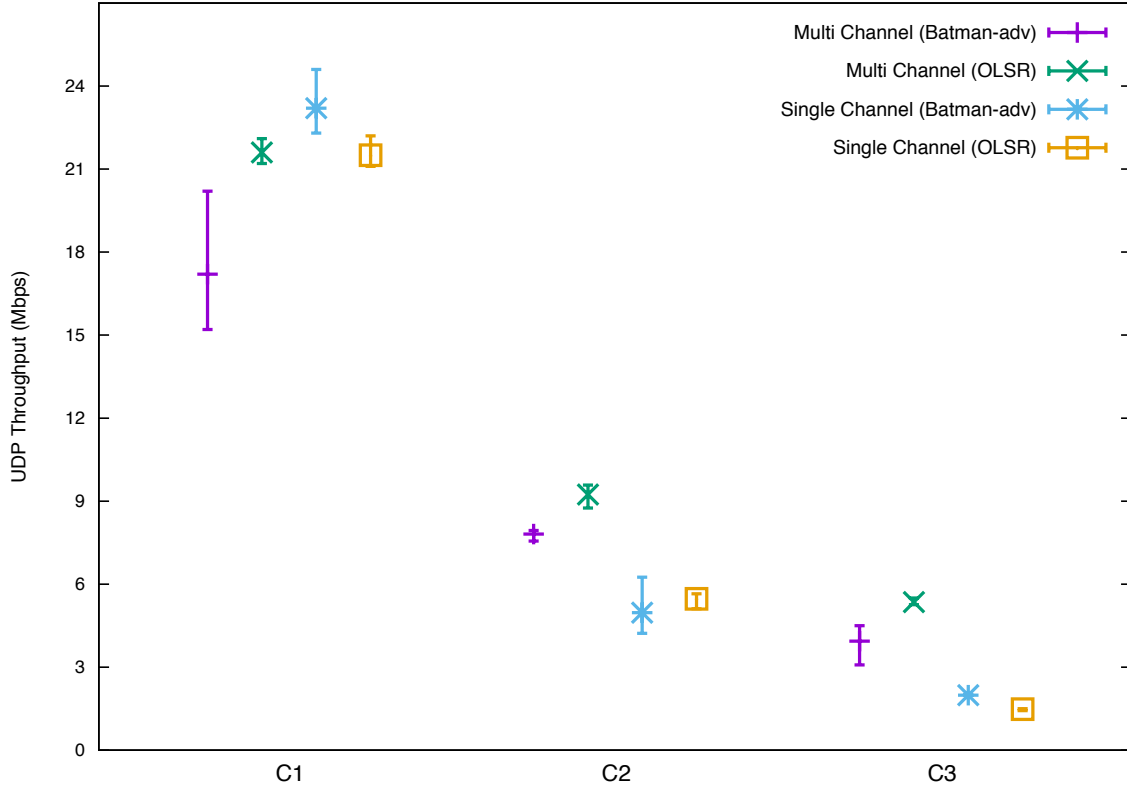


Figure 5.5: Minimum, Maximum and Average UDP throughput from C1, C2 and C3 to S for Phase 2 Testing.

Observations in UDP Throughput

At C1, average UDP throughput in case of Single Channel using batman-adv was observed to be 17.2 Mbps, while in all the other 3 setups, the observed value was at least 21.5 Mbps. In case of OLSR, Multi channel performs as good as Single channel at C1. Moreover at C1, Single channel using batman-adv performs much better than Multi channel using batman-adv. On the other hand, at C2 and C3, Multi Channel using batman-adv as well as Multi channel using OLSR give better throughput than their Single channel scenarios. At C3, Single channel using OLSR performs worst with minimum throughput value observed to be 1.43 Mbps, which is still good enough for video communication.

5.3.2 Packet Loss Rate

Figure 5.6, 5.7 and 5.8 present the average packet loss rate from C1, C2 and C3 to S against various UDP bandwidth values.

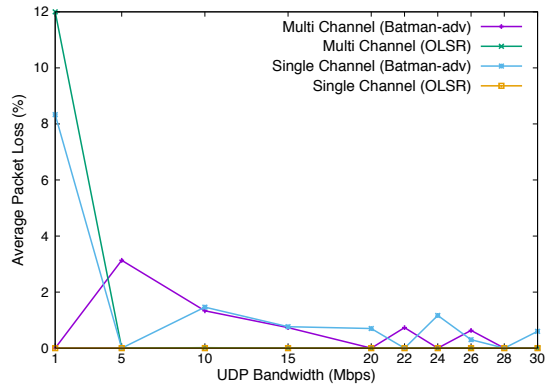


Figure 5.6: Packet Loss Rate from C1 to S for Phase 2 Testing.

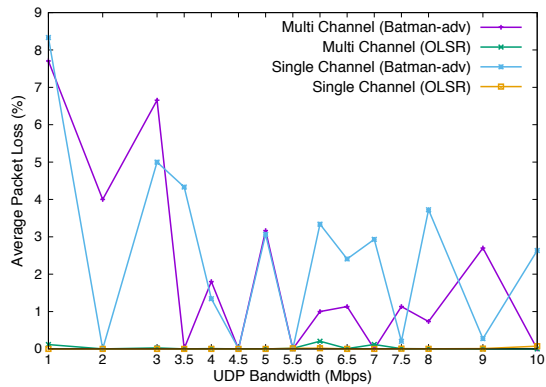


Figure 5.7: Packet Loss Rate from C2 to S for Phase 2 Testing.

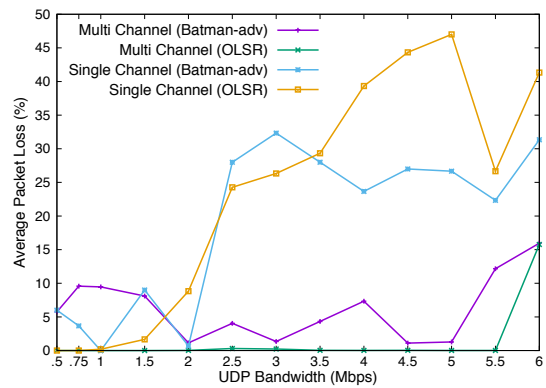


Figure 5.8: Packet Loss Rate from C3 to S for Phase 2 Testing.

Observations in Packet Loss Rate

Not much packet loss was observed at C1 where highest loss was observed to be 12% in case of Multi channel using OLSR with 1 Mbps UDP throughput. At C2, Multi channel using batman-adv as well as Single channel using batman-adv suffer from packet loss with highest value being 8.3%, while Multi channel using OLSR and Single channel using OLSR have negligible packet loss. On the other hand, at C3, Single channel setups start facing significant packet loss with highest value

observed to be 47% in case of Single channel using OLSR. Both, Multi channel setups do not have much packet loss with Multi channel using OLSR, in particular, showing almost negligible packet loss.

5.3.3 TCP Throughput for Parallel Connections

Figure 5.9, 5.10 and 5.11 show the average throughput available to each of the connection for 1, 2, 3, 4, 5 and 10 parallel TCP connections from C1, C2 and C3, respectively.

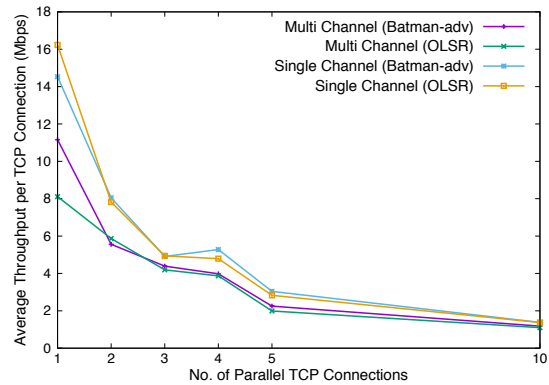


Figure 5.9: TCP Throughput from C1 to S for Phase 2 Testing.

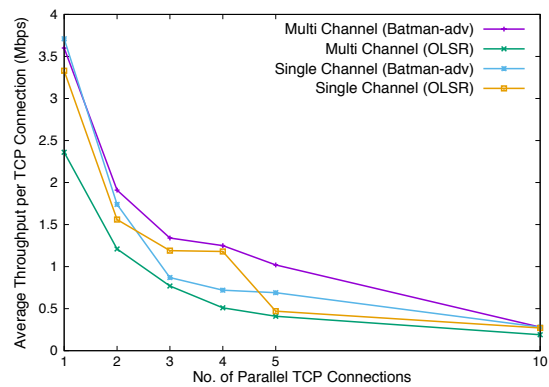


Figure 5.10: TCP Throughput from C2 to S for Phase 2 Testing.

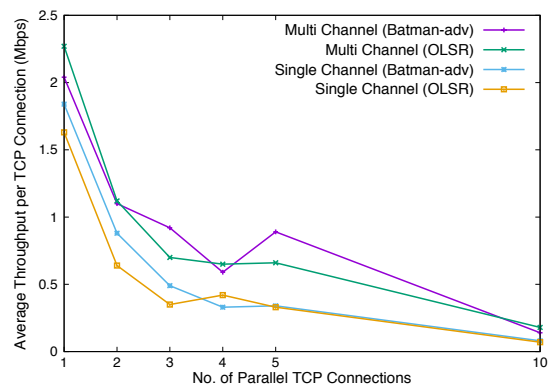


Figure 5.11: TCP Throughput from C3 to S for Phase 2 Testing.

Observations in TCP Throughput

At C1, maximum throughput value for 1 TCP connection was observed to be 16.23 Mbps. On the other hand, the average throughput to 1 connection in case of 10 parallel TCP connections was at least 1.1 Mbps, which is good enough for file transfer. Moreover, at C1, Single channel using batman-adv as well as Single channel using OLSR give better throughput than Multi channel scenarios. At C2, Multi channel using batman-adv shows better results than any other setup with highest value observed to be 3.6 Mbps, while Multi channel using OLSR performs worst with throughput value to be 270 Kbps to one connection in case of 10 parallel TCP connections. On the other hand, at C3, both, Multi channel using batman-adv and Multi channel using OLSR outperform their Single channel scenarios. The highest throughput for 1 connection was observed to be 2.04 Mbps in case of Multi Channel using batman-adv.

5.4 System Evaluation

Table 5.2 summarises the qualitative analysis of our system compared with other similar systems.

Table 5.2: Qualitative Analysis of Our Approach

Parameter	DUMBO [29]	PDRNET [5]	DISANET [30]	Our System
Size of the system	Large. There is an implementation with raspberry-pi like small box too, but it cannot be used as user terminal.	Large, but no additional device is needed to deploy the network as per its design.	Large, but does not require any additional hardware.	Small, but needs laptop or desktop computer to boot the OS image from USB flash drive. Additional Wi-Fi NICs can be used to extend the wireless service.
Technical Complexity	High. Requires users to manually configure the system.	Complex	Complex	Easy to use. GUI helps users to simply turn on and off the services.
Hardware Requirements	Laptops, PDAs	PoE enabled Wi-Fi Access Points, Digital Broadcast System	Ground tethered helium balloon, LTE Base Station, GSM System	Laptops, USB flash drives, USB Wi-Fi NICs

[31] presented an evaluation of testbed using OLSR and B.A.T.M.A.N. Their experiment was done in closed space using IEEE802.11b as MAC protocol with channel 1 used as central frequency, and included 5 nodes in the system. They tested three models: all nodes are static, only one node is moving, two nodes are moving. Since static model of their setup is closest to our system, we present the results of that model only. In static model, maximum UDP as well TCP throughput using both OLSR and B.A.T.M.A.N. was around 500 Kbps.

Chapter 6

Related Work

Mobile ad-hoc networking and wireless mesh networks have been actively proposed and examined for post-disaster recovery. [32–35] discussed communication network using MANET, where highly demanded protocols and applications, like flooding-based communication protocol, push to talk and telemedicine, are made available in post disaster situation. [36] examined Multicast in MANET to efficiently disseminate information. [37] introduced Wireless Mesh Network that can be used in both daily life and emergency situations. [38] proposed to make use of survival time of wireless sensor networks and evacuate critical data to safe zone. [39] examined combination of Overlay Network and MANET to provide redundancy and continuity of services. Their solution expects the availability of high speed network to appropriately place mirror services through overlay network. [40] puts forward a session-based mobility management in MANET which meets the requirements of rapid deployment of the network with auto-configuration. In case of emergency, the agents try to disseminate information to disaster information center.



Figure 6.1: Pictures taken by by Dr. Kotaro Kataoka during ICT environment recovery efforts after 2011 Tohoku Earthquake and Tsunami in Japan.

There has been significant amount of work done on recovering the ICT environment at a disaster site. [41] proposed a quickly-deployable package for post-disaster communication and [5] reported how such systems were used in 2011 Tohoku Earthquake and Tsunami in Japan. Figure 6.1 shows snapshots of the system in work. They proposed a self-sustainable system that included all of the packages needed for long run of the system e.g., power supply, satellite, Wi-Fi, digital broadcasting,

etc. The size of the system is large, but no additional device is needed to deploy the network as per its design. [29] presented a fully-functional prototype of long-distance multimedia wireless mesh network during the time of large-scale disaster, but their solution is aimed at networking in open space, where laptop computers used as mesh nodes may move around. There is an implementation of their system with raspberry pi too, but raspberry pi cannot be used as user terminal.



Figure 6.2: Pictures taken by by Dr. Kotaro Kataoka during field trial conducted by IIT Madras as part of DISANET demo.

[30] proposed a system that uses ground tethered helium balloon to raise radio part of LTE base station 20-25 m above ground as shown in Figure 6.2. This balloon is present at Master Operation Center, which is set up at center of disaster site as part of recovery efforts, and communicates with GSM Mobile Switching Center to switch calls between GSM handsets. Rescue workers carry GSM handsets and are allowed to make any number of voice calls. Their system is large and complex, but does not require any additional hardware. [31] analyzed the testbed using OLSR and B.A.T.M.A.N. Their setup included 5 nodes and was conducted in a closed space inside the campus. They used the Link Quality Window Size (LQWS) of OLSR evaluate the performance of OLSR. Their experiment showed that TCP throughput was better using low LQWS value of OLSR.

Chapter 7

Conclusion and Future Work

In this thesis, we considered the challenges of having easy access to computers and communication infrastructures in post-disaster settings. We proposed a solution using Linux Live USB flash drive, where the guest OS and a set of preloaded software are ready to serve. One can attach the USB flash drive to an available computer, and boot her own OS to get access to the computing and networking resources. A Live USB node can also act as a Wi-Fi hotspot in a wireless ad-hoc setting. This system is intended to provide some form of networking to people and rescue workers at disaster-affected site so that they do not have to wait for the recovery of ICT environment, which may take some time to fully recover. It does not depend on existing LAN infrastructure and can be deployed very easily by anyone. The results of field trial proved that this system can facilitate the online services and enable user devices such as smartphones and tablets to access them through Wi-Fi hotspot and ad-hoc network with or without the Internet connection.

As part of this thesis, we tested two scenarios of wireless ad-hoc networking: Single Channel and Multi Channel. The results proved that as the size and number of hops in the network increase, Multi Channel setup performs better than Single Channel Setup. A hybrid approach using more than 2 nodes in the same mesh, can be tested that will be helpful to understand the performance of this system in a better way. Also, adding one more hop to Multi Channel Setup will give more insight into the performance of batman-adv and OLSR. If OLSR performs better, then it can be used on Live USB nodes, instead of batman-adv. Simulations can be conducted to explore the appropriate size of network for larger-scale deployment. The usability of the system on platforms other than Ubuntu 14.04.1 LTS is highly doubtful since we did not test this system on other platforms. Moreover, the ad-hoc networking in 5 GHz frequency band is not very well supported. Developers are highly recommended to download and install latest drivers, corresponding to Wi-Fi chip they are using, if they wish to fully utilise the capabilities of 5 GHz band. The best approach would be to install latest drivers for Wi-Fi chips from popular manufacturers so that end-user does not have to face any issue with the Wi-Fi card.

References

- [1] A. Utani, T. Mizumoto, and T. Okumura, “How geeks responded to a catastrophic disaster of a high-tech country: Rapid development of counter-disaster systems for the great east japan earthquake of march 2011,” in Proceedings of the Special Workshop on Internet and Disasters, ser. SWID '11. New York, NY, USA: ACM, 2011, pp. 9:1–9:8. [Online]. Available: <http://doi.acm.org/10.1145/2079360.2079369>
- [2] Japan Meteorological Agency, “Earthquake Early Warning,” <http://www.jma.go.jp/jma/en/Activities/eew.html>, [Accessed: 2015-05-26].
- [3] Nuclear Safety Technology Center, “The System for Prediction of Environment Emergency Dose Information(SPEEDI),” http://http://en.wikipedia.org/wiki/Radiation_monitoring_in_Japan#SPEEDI_Network, [Accessed: 2015-05-26].
- [4] Ministry of Health, Labour and Welfare, “Emergency Medical Information System,” <http://www.wds.emis.go.jp/>, [Accessed: 2015-05-26].
- [5] K. Kataoka, K. Uehara, O. Masafumi, and J. Murai, “Design and deployment of post-disaster recovery internet in 2011 tohoku earthquake,” IEICE transactions on communications, vol. 95, no. 7, 2012, pp. 2200–2209.
- [6] Practical Networking Lab, “Networking in Post-Disaster Situation,” http://pranet.iith.ac.in/?page_id=142, [Accessed: 2015-05-29].
- [7] Wikipedia, “Prefix delegation,” http://en.wikipedia.org/wiki/Prefix_delegation, [Accessed: 2015-06-21].
- [8] Open-mesh, “B.A.T.M.A.N. Advanced,” <http://www.open-mesh.org/projects/batman-adv/wiki/Wiki>, [Accessed: 2015-06-06].
- [9] “OLSR,” http://www.olsr.org/mediawiki/index.php/Main_Page, [Accessed: 2015-06-06].
- [10] “MediaWiki,” <https://www.mediawiki.org/wiki/MediaWiki>, [Accessed: 2015-05-29].
- [11] “WordPress,” <https://wordpress.org/>, [Accessed: 2015-05-29].
- [12] “WebRTC,” <http://www.webrtc.org>, [Accessed: 2015-05-29].
- [13] “Getting Started with WebRTC,” <http://www.html5rocks.com/en/tutorials/webrtc/basics/#toc-rtcpeerconnection>, [Accessed: 2015-06-06].

- [14] “Linux Dash,” <http://linuxdash.afaqtariq.com/>, [Accessed: 2015-05-29].
- [15] “SmokePing,” <http://oss.oetiker.ch/smokeping/>, [Accessed: 2015-05-29].
- [16] “Index of /batman/releases/batman-adv-2014.3.0,” <http://downloads.open-mesh.org/batman/releases/batman-adv-2014.3.0/>, [Accessed: 2015-06-06].
- [17] “Quagga Routing Suite,” <http://www.nongnu.org/quagga/>, [Accessed: 2015-06-06].
- [18] Linux Wireless, “Hostapd,” <http://wireless.kernel.org/en/users/Documentation/hostapd>, [Accessed: 2015-05-29].
- [19] “Dnsmasq,” <http://www.thekelleys.org.uk/dnsmasq/doc.html>, [Accessed: 2015-05-29].
- [20] “Squid: Optimising Web Delivery,” <http://www.squid-cache.org/>, [Accessed: 2015-05-29].
- [21] “Download MediaWiki,” <http://www.mediawiki.org/wiki/Download>, [Accessed: 2015-06-06].
- [22] “Download WordPress,” <https://wordpress.org/download/>, [Accessed: 2015-06-06].
- [23] “Package olsrd,” <http://packages.ubuntu.com/search?keywords=olsrd>, [Accessed: 2015-05-29].
- [24] “Package batman-adv-dkms,” <http://packages.ubuntu.com/search?keywords=batman>, [Accessed: 2015-05-29].
- [25] “B.A.T.M.A.N. Advanced quick start guide,” <http://www.open-mesh.org/projects/batman-adv/wiki/Quick-start-guide>, [Accessed: 2015-06-06].
- [26] “Wireshark,” <https://www.wireshark.org/>, [Accessed: 2015-05-29].
- [27] “Hostapd: The Linux Way to create Virtual Wifi Access Point,” <https://nims11.wordpress.com/2012/04/27/hostapd-the-linux-way-to-create-virtual-wifi-access-point/>, [Accessed: 2015-05-29].
- [28] “Iperf,” <https://iperf.fr/>, [Accessed: 2015-05-29].
- [29] K. Kanchanasut, A. Tunpan, M. Awal, T. Wongsardsakul, D. Das, and Y. Tsuchimoto, “Dumbonet: a multimedia communication system for collaborative emergency response operations in disaster-affected area,” *International Journal of Emergency Management*, vol. 4, pp. 670–681, 2007.
- [30] D. Jalihal, R. D. Koilpillai, P. Khawas, S. Sampooram, S. H. Nagarajan, S. Kalpalatha, R. Bhavani, K. Takeda, K. Kataoka, “A stand-alone quickly- deployable communication system for effective post-disaster management,” in *Proceedings of the IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, August 2014.
- [31] L. Barolli, M. Ikeda, G. De Marco, A. Durresi, F. Xhafa, “Performance Analysis of OLSR and BATMAN Protocols Considering Link Quality Parameter,” in *Proceedings of the Advanced Information Networking and Applications (AINA)*, May 2009, pp. 307–314.

- [32] T. Umedu, H. Urabe, J. Tsukamoto, K. Sato, and T. Higashinoz, "A manet protocol for information gathering from disaster victims," in Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom), March 2006.
- [33] Y.-N. Lien, H.-C. Jang, and T.-C. Tsai, "A manet based emergency communication and information system for catastrophic natural disasters," in Proceedings of the 29th IEEE International Conference on Distributed Computing Systems Workshops (ICDCS), June 2009, pp. 412–417.
- [34] Y.-N. Lien, L.-C. Chi, and C.-C. Huang, "A multi-hop walkie-talkie-like emergency communication system for catastrophic natural disasters," in Proceedings of the 39th International Conference on Parallel Processing Workshops (ICPPW), Sept 2010, pp. 527–532.
- [35] J. Kim, D. Kim, S. M. Jung, C. Lee, D. Lim, S. Hong, S. K. Yoo, "Development of mobile ad hoc network for emergency telemedicine service in disaster areas, " in Proceedings of the International Conference on New Trends in Information and Service Science (NISS), June 2009, pp. 1291–1296.
- [36] M. Iqbal, X. Wang, D. Wertheim, and X. Zhou, "Swanmesh: a multicast enabled dual-radio wireless mesh network for emergency and disaster recovery services," *Journal of Communications*, vol. 4, no. 5, 209, pp. 298–306.
- [37] Y. Takahashi, Y. Owada, H. Okada, and K. Mase, "A wireless mesh network testbed in rural mountain areas," in Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization. ACM, 2007, pp. 91–92.
- [38] C. Debata and R. Roy, "Efficacy of power of two choices technique for data evacuation process in sensor networks for post-disaster relief operations," in Proceedings of the IEEE International Conference on Signal Processing, Computing and Control (ISPCC), Sept 2013, pp. 1–6.
- [39] Y. Shibata, H. Yuze, T. Hoshikawa, K. Takahata, and N. Sawano, "Large scale distributed disaster information system based on manet and overlay network," in Proceedings of the 27th International Conference on Distributed Computing Systems Workshops (ICDCSW), June 2007, pp. 33–33.
- [40] R. Li, R.-L. Yang, and D.-W. Hu, "A designing of mobility management mechanism in manet in disaster-rescue situations," in Proceedings of the 11th IEEE International Conference on Communication Technology (ICCT), Nov 2008, pp. 596–599.
- [41] K. Kataoka, K. Uehara, and J. Murai, "Lifeline station: A quickly deployable package for post disaster communications," in Proceedings of the Internet Conference, 2009, pp. 41–47.