University of Nevada, Reno

A DEVICE-TO-DEVICE SERVICE SHARING MIDDLEWARE FOR HETEROGENEOUS WIRELESS NETWORKS

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of Master of Science in

Computer Science and Engineering

by

Sandeep Mathew

Dr. Murat Yuksel / Thesis Advisor

December 2015

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

SANDEEP MATHEW

Entitled

A DEVICE-TO-DEVICE SERVICE SHARING MIDDLEWARE FOR HETEROGENEOUS WIRELESS NETWORKS

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Murat Yuksel, Ph.D, Advisor

Shamik Sengupta, Ph.D, Committee Member

Jihwan Yoon, Ph.D, Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

December, 2015

ABSTRACT

Wireless devices with diverse capabilities are ubiquitous and will continue to flourish for the next foreseeable future. There is heterogeneity with respect to capabilities as well as wireless transmission technology used. A critical issue to address in the face of growing wireless device usage is the increasing lack of spectrum and wireless resources. Sharing the wireless resources and bandwidth is considered by many to be a reasonable approach to the spectrum scarcity problem. This thesis describes a system where users may share wireless services with other users in a seamless manner. We develop a smart phone application for service sharing. By using the application as a middleware, users can get more work done via sharing, increase the spectrum utilization, create their own private network without a centralized infrastructure and offload network traffic to a less congested network path.

We present architectural details of a smart-phone application and offer various ways one can implement such a wireless service sharing framework. Our application may pave way for a model in which users can sell their own services to other end users and establish a device-to-device service sharing market beneficial to the entire user set.

ACKNOWLEDGEMENTS

I thank UNR for giving me an opportunity to study and grow. I would like to express my sincere gratitude to my adviser Dr Murat Yuksel and My co adviser Dr Shamik Sengupta for constant encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

Wireless devices offer a variety of features and services that are unique to each and every device. The main motivation of this work is to address the concern of enabling communication facilities in environments where a central communication infrastructure cannot be relied upon, improving the RF spectrum utilization and further enabling the sharing of mobile facilities and services available in contemporary mobile devices.

Complete dependence on infrastructure can be a major bottleneck during disasters or emergency situations. In disaster scenarios, the central infrastructure facilities may be unavailable [17], and in these scenarios communication is crucial. Existing commodity mobile devices are already equipped with hardware facilities to communicate between each other without a central infrastructure. Device-to-device sharing of wireless resources is a great way of circumventing this dependence on infrastructure.

Another motivation for sharing services is that the RF spectrum is becoming a critical resource as it is getting scarcer and the mobile data demand is increasing super-linearly every year [13]. We need better approaches to utilize the RF spectrum efficiently. Mobile devices communicate with the base station even if they are close by (few meters) for various services. Device-to-device service sharing mechanism can be used to eliminate the communication with the base station. Hence, we

are able to harvest more out of the scarce spectrum by not bothering the spectrum with connections to the base station. Spectrum utilization can be further improved by opportunistic communication over multiple hops. This will extend the reach of the base stations hence further improve spectrum utilization. Our approach also makes use of heterogeneous radios further improving the spectrum utilization by offloading network traffic to less congested wireless paths.

Sharing of the Internet connectivity is already a mechanism built into many mobile devices in the form of tethering but it is essentially single hop and typically works only with homogeneous links. Tethering may also be limited by the carrier's permission and some carrier's impose an additional fee for tethering. However, existing solutions involve device specific techniques and are not agile to multiple carriers. In this thesis, we have developed a smart-phone application that extends this notion of sharing to "service sharing" and performs the sharing at the user-level over multiple hops involving a heterogeneous set of wireless links. This type of sharing is a great example of how wireless connectivity can be extended via device-to-device sharing.

Our application allows to share services so that a device is able to access services which are not available to a particular user. This can be illustrated by the following scenario. The Figure 1.1 below shows three different users Bob, Joe and Sando having different wireless services accessible to them individually. Bob is a user having a mobile device with a high resolution camera, Joe is a user with free SMS (text messaging) and Sando is another user with unlimited 3G service. After the application is installed in all the mobile devices, Bob initiates a service discovery request to see what all services he can use. Bob's neighbors respond with services

available to them, unlimited SMS and 3G Service. Bob will be given an option to activate these services. Once activated, Joe will be able to utilize both free SMS service and unlimited 3G service.

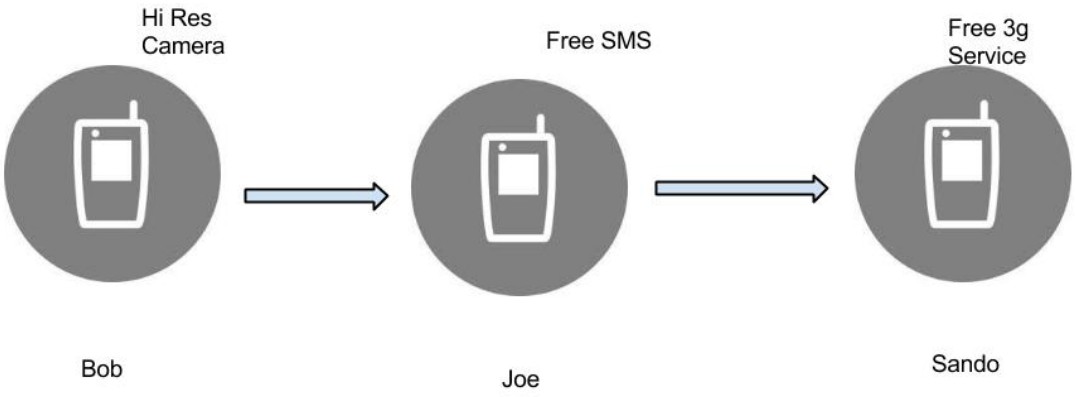

Figure 1.1: A sample scenario

Before designing and implementing our smartphone application for sharing services following scenarios were considered:

Simple sharing of services

This scenario is very similar to a barter system in which two co-operating nodes with dissimilar services are able to utilize both the services. This is depicted by Figure 1.2. In this scenario Bob has unlimited SMS service and Joe has unlimited Internet access via 3G service. After the sharing of services both Bob and Joe get unlimited SMS and unlimited Internet access respectively.

Increase in the overall network range by co-operating

In this scenario, a node is able to access a service from a node not within its range

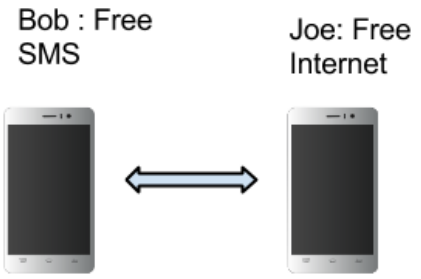

Figure 1.2: Figure showing simple sharing of services

by using a second node as a relay. In the Figure 1.3, La is not in range of Bob. But Bob is able to access the high resolution camera of La by using Joe as relay. This increases the overall network range of Bob.

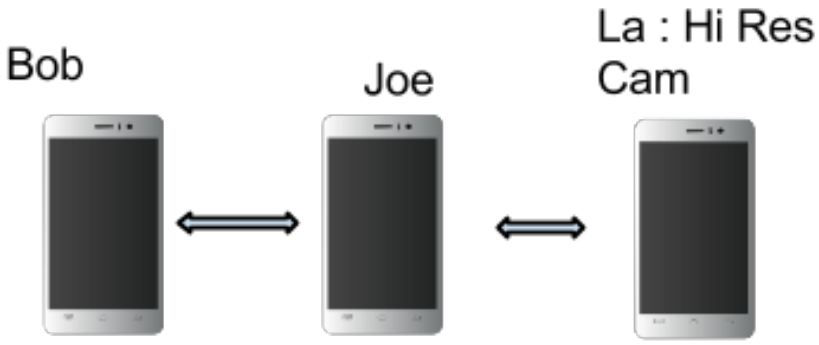

Figure 1.3: Figure showing increase of network range

Offloading network traffic to other nodes

In this scenario, a node need to fetch multiple files from the Internet, instead of fetching the file from a single node. We make use of a simple node to fetch both the files the node makes use of two different nodes to fetch the files. Thus it is able to get the work done simultaneously. The scenario is shown in Figure 1.4. Joe is

connected to Bob via Wi-Fi and Joe is connected to Sando via Bluetooth. Both Joe and Sando have Internet access while Bob does not. Joe wants to download two files from the Internet. Since Bob does not have Internet access he requests Joe to download files for him. At this point Joe can request Sando to download one of the files from the Internet and later transfer it to Bob. Thus, by opportunistically offloading the work, network utilization is improved.

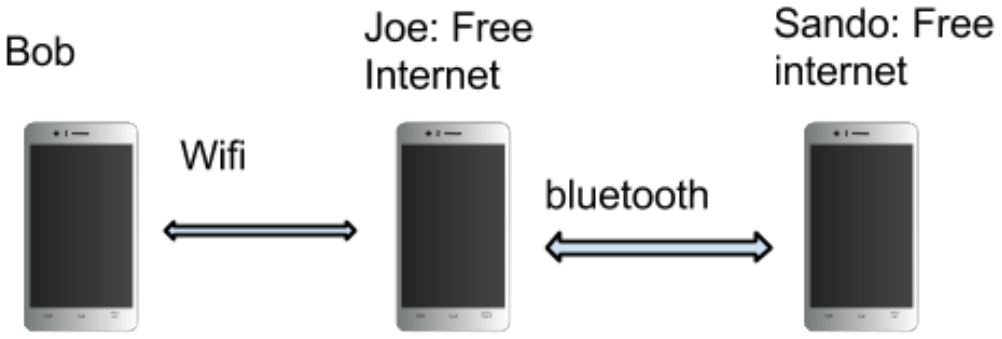

Figure 1.4: Offloading network traffic scenario

Other usage scenarios

- *Remote monitoring*: The application allows sharing of the camera. This can be used for remotely monitoring places and items which cannot be monitored directly. Crime fighting agencies like FBI may make use of this facility to use of camera of other person while the person gets benefited monetarily.
- *Redundant infrastructure*: The application can be used to form an adhoc network with redundant connections. This can be of use in emergency situations when some of the nodes may go down due to a disaster and new wireless networks can be formed easily.

- *Better spectrum utilization*: The application allows us to use heterogeneous wireless systems in a transparent manner such as 3G, WiFi and Bluetooth. Thus once a wireless link using a particular frequency range becomes congested it may use another link on a different frequency range to carry out the request. This may lead to better utilization of the spectrum.

Our smart-phone application implements all of the presented usage scenarios. Most of the prior work has been on theoretical frameworks, in our work we come up with an implementation for sharing services. We measure the delay, overhead introduced due to our application layer framework and the throughput obtained. We observe that even up to three hops the quality of service is acceptable to send video messages.

The thesis is organized as follows: We first detail application usage scenarios where device-to-device sharing is useful. Chapter 2 provides a summary of the relevant literature. Chapter 3 gives the details of our prototype design and implementation with a basic explanation of related technology. Chapter 4 summarizes our research by discussing results of experiments that we conducted with our prototype. In Chapter 5, we discuss ideas that can potentially increase the system performance and further research directions.

CHAPTER 2

LITERATURE REVIEW

This chapter summarizes the literature related to our work. We cover several papers to serve the purpose. The related work in the literature can be categorized into following:

- Peer-to-peer resource sharing
- Current service sharing technologies

2.1 Peer-to-peer resource sharing

MeshChord is a peer-to-peer resource sharing protocol described in [11]. It considers effects of MAC-cross layering and location awareness to improve the efficiency of the comparable Chord [49] protocol in wired systems. Chord is a scalable peer-to-peer lookup protocol for internet applications. It can be used for looking up a node in a network given the resource key. The authors of MeshChord propose a two tier architecture consisting of mobile mesh clients and stationary router nodes so as to make it suitable for ad hoc networks. They exploit location awareness by encoding nearby nodes so that their identification number does not vary by much. The make use of cross layering by caching packets in intermediary nodes and sending them to application layer to see if the lookup can be performed before it reaches the destination node. It significantly differs from our architecture where nodes are completely peer-to-peer and heterogeneous. The focus of our work is primarily on service sharing as opposed to lookup.

The authors of Friend Relay [10] describe the architecture of a resource sharing framework for mobile wireless devices that provides automatic publishing, discovery and configuration, monitoring and control. They also incorporate power awareness and security awareness into application so that resources of a node is not heavily used by its neighbors. The architecture they describe is built on existing protocols ZeroConf [24] for automatic configuration and BEEP [47] for session management and authentication. It however, is essentially meant for singly hop communication using same kind of wireless links. Our architecture and implementation supports multi-hop communication over heterogeneous wireless links.

Authors of [52] explore the benefits and challenges associated with sharing resources on mobile operating systems. A critical benefit described in the paper is the opportunistic use of idle resources in co-located devices for better network utilization. Another benefit described is the savings in power obtained by using neighbors resources instead of local resources. This is due to the fact that most resources have a starting phase which consumes more power. They conclude that enabling resource sharing on operating system level may lead to crowd-sourcing and greater collaboration. The main challenges described are adaptive resource discovery, selection of the right node, collaboration and efficient aggregation of resources and challenges associated with selecting the right IPC mechanism.

Authors of [37] describe a crowd sourced mobile Internet access framework and describe a distributed incentive mechanism based on Nash bargaining solution. The Internet service model described takes into account power consumption, users data requirements and financial costs. This is restricted to internet sharing while our design and implementation allows general sharing of services.

Routing protocols for peer-to-peer file sharing is evaluated in [15]. They compare the performance of simple broadcast based schemes and schemes using a distributed hash table (DHT). In one of the routing schemes they augment the hash table with information from the network layer thus achieving cross layering. They conclude that this cross layered scheme performs better and provide more scalability. We implement our service completely in user space, hence cross layering is not possible. The focus of our work is primarily on service sharing than routing.

The architecture of a resource aware middleware over mobile ad hoc networks for rescue and emergency applications is described in [16]. They make use of the AODV [46] for routing and routing information is used to predict which resources will be available in the near future. Their architecture is component based as opposed to layered as in our framework. The work, however is primarily targeted toward resource identification. The problems of heterogeneity and multi-hop access is not considered. But our work not only performs resource identification but also provides a mechanism to conveniently access services.

The authors of [22] describe DICHOTOMY which is a framework for resource discovery, scheduling and for picking up the most resourceful node in multi-hop ad hoc grids so that messaging overhead can be minimized. This solves only a part of the problem, which is, finding the right node in the wireless grid. In our framework we describe for methods for service discovery and as well as usage. They also do not consider the effect of heterogeneity of wireless links. They propose the use of MAC ID for node identification which may not be easily obtained in the case of commodity mobile devices. We make use of user defined device names for

identification.

The architecture of an efficient service-oriented framework over MANETS for agile environments is presented in [50]. The main components of the framework are Group Manager which is responsible for resource and service discovery and AgServe which provides dynamic instantiation, definition, calling and closure of services. The architecture is more suited for scenarios mobile nodes are reporting to a central node for publishing and sharing their services. The architecture does not consider multi-hop scenarios and heterogeneous wireless links.

An architecture and application for information resource sharing in MANETS is presented in [43]. The implementation is however is meant for commodity computer hardware instead of smart phone mobile devices. The focus of the work is mainly information resource sharing (document transfer) instead of general service sharing. The architecture is suited toward single hop communication than multi-hop transfer. The architecture does not take heterogeneity of wireless links into account.

2.2 Current service sharing technologies

IEEE 802.11 wireless modes and tethering

IEEE 802.11 is set of specifications for implementing Media Access Layer (MAC, Datalink) and Physical Layer (PHY) for wireless local area networks. IEEE 802.11 specifies two operating modes and they are (a) Infrastructure mode (b) Ad hoc mode.

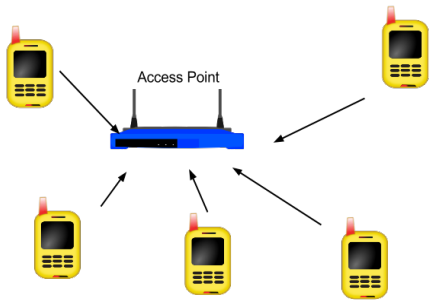

Figure 2.1: WLAN operating in Infrastructure mode

In infrastructure mode, all the wireless nodes are connected to a single access point (AP). Typically mobile phones are configured to operate in infrastructure mode and ad hoc mode is typically hidden from the user. A wireless router is an example for an access point. The Figure 2.1 illustrates infrastructure mode operation. In ad hoc mode which is also called infrastructure-less mode each node is connected to another in a peer to peer fashion. This mode needs to be activated in most mobile devices by executing configuration commands at an elevated privilege level. The way to reach the elevated privilege level and putting a device in ad hoc mode is heavily dependent on the type of mobile device used. At the 802.11 protocol layer level only single hop communication is supported and it is up to higher layers to provide for multi hop communication.

Ad hoc mode is important for our implementation because ad hoc mode facilitates tethering. Tethering involves forwarding network packets from one interface to another thereby each mobile device has the capability to act as a portable hotphot. This allows each mobile node to share the its own 4G, 3G, 2G data connection with other devices. The mobile devices within range connect to the hotspot and

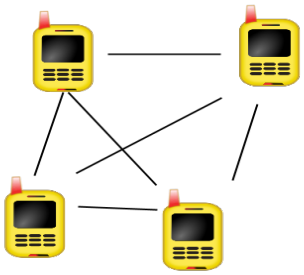

Figure 2.2: WLAN operating in Ad hoc mode

the tethered device acts a Wi-Fi access point. In places of missing infrastructure, this may be the only way to access the Internet. Also, tethered devices can be made to act as femto cells [3] [4].

Some of the Andorid smart-phones and tablets have implemented tethering facilities in their firmware. If it is not present it needs to be enabled by a special application which has elevated privileges. The Android system is a Linux system under the hood and root account has the maximum privileges. Therefore the term ”rooting” is used to mean gaining elevated access on an Android system. Rooting allows the user to bypass some of the software-enforced restrictions meant for typical mobile users. A rooted phone can be used to install custom operating system kernel and kernel modules typically known as Roms. Cyanogen-Mod [51] is a popular replacement for the stock Android kernel. Cyanogen-Mod enables easy support for Wi-Fi, Bluetooth and USB tethering. Tethering in Android devices is accomplished by the netfilter [42] kernel module provided by the Linux kernel. `iptables` command is the command line interface to the netfilter kernel module. Most tethering implementations take advantage of this utility.

Applications that facilitate sharing.

Android Wi-Fi-Tether, Wireless Tether for Root Users [27] allows tethering of Wi-Fi interface for rooted handsets running Android. Clients can connect to the tethered device and and avail the data connection using 4G, 3G, 2G mobile connection which is presently available in the handset. The application has provisions for more fine grained access control, network encryption, changing Wi-Fi channels and options for power levels.

Open Garden [18] is an application that offers Wi-Fi hotspot tether and Bluetooth tethering. The application allows sharing 3G/4G Internet connection with different operating systems and assumes no tether fees. It also needs root access on the device. Open Garden allows mobile devices to form a wireless peer-to-peer mesh network that provides Internet access. Installation of the Open Garden is mandatory on all devices in the mesh. The created mesh is capable of opportunistic local connections. This enables better bandwidth and coverage while reducing transmission power.

Tethering does have issues that users may be concerning to users. These issues include quality of service due to bandwidth reduction, power impact, security and privacy considerations. Tethering involves sharing own bandwidth with other users which leads to reduction of bandwidth and reduction in quality of the service. Tethering depending on the network traffic depletes the battery of the mobile phone. Some carriers impose a fee to enable tethering, while others forbid tethering. Also allowing other users to connect to your tethered mobile device is a cause of privacy concern for some mobile users.

Several Mesh [19] is a telephony platform that uses ad hoc Wi-Fi capability of the device to form a wireless mesh network. It allows the phone to perform voice calls, short messages and other modes of communication without a central carrier facility. This also requires installation of patched version of Android called Cyanogen-Mod for better functioning.

Multinet [12] is a software-based approach where a single wireless card is virtualized by introducing an intermediate layer below IP. In most of the wireless LAN cards switching from infrastructure to the ad hoc mode needs a reset from the firmware which makes it difficult for wireless interface to be in infrastructure and ad hoc mode. Multinet facilitates simultaneous connections to multiple networks. However it has not been extended to support multi hop scenarios. Multinet is implemented only for the Windows platform and makes extensive use of the services provided by the Windows platform, and hence is, difficult to port to other platforms.

BEDnet [20] is a J2ME [45] based mobile application. Its implementation uses a scatter-net forming algorithm and makes use of the DSDV algorithm for routing. The choice of J2ME makes its use restrictive in modern smart phone since that API is not supported anymore. Another limitation of BEDnet is the poor transfer speed. Beddernet [21] is the successor of BEDnet [20] which aimed to address the drawbacks of BEDnet. It was built on using the RFCOMM [7] protocol and implements the DSDV [25] routing protocol. It provided better throughput and multiple applications were able to make use of the framework. The framework facilitated the applications to use it in uni-cast as well as multicast mode. Major drawback of

Beddernet project is the cost of setting up RFCOMM connection between devices. ORWAR [48] project on Android makes use of Wi-Fi and implements DTN routing [9]. ORWAR [48] implementation relied on GPS information for building the routing table therefore it can be used only for devices capable of GPS functionality and is best suited to outdoor usage. Another drawback is that it is developed as a standalone application for Android and does not provide any API that can be used by other applications. [6] describes a middleware for mobile peer-to-peer computing using the older J2ME platform. The middleware is more centered around single hop communication by the formation of peer groups. Ad hoc on Android [39] implements a library that can be included in Android applications that needs to run on a mobile ad-hoc network, this implementation uses reactive routing based on AODV [46] but the library cannot be used by multiple applications at the same time. BATMAN [38] and OLSR [14] are routing protocols and library implementations used by many open source wireless mesh networking projects. They however do require root access to the network nodes for installation.

Crowdshare [2] describes an implementation of tethering with security and privacy improvements. The byzantium project [26] is a Linux distribution that helps to set up ad hoc mesh networks quickly. The pinwheel messenger [30] is a wireless mesh based messaging system that works by changing the device name and uses the inbuilt discovery functionality in both Bluetooth and Wi-Fi to communicate status messages. Village Telco [36] provides a do it yourself telephony platform using open source components. Project SPAN [31] provides an ad hoc manger for certain implementations of Android. A middleware for managing adhoc networks on Android is described in [53].

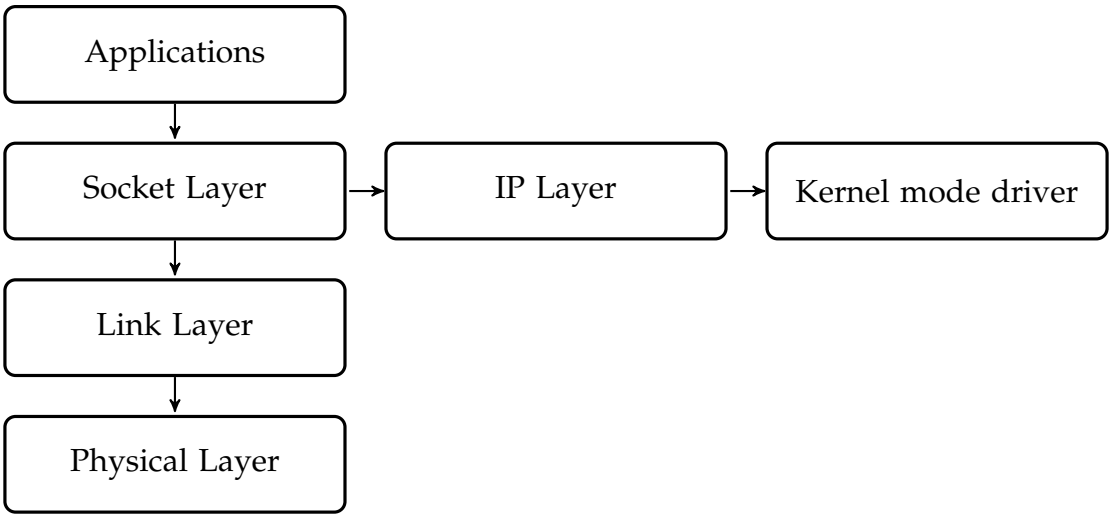

Figure 2.3: Framework implemented in kernel mode

2.2.1 Survey of implementation techniques

The following section describes various implementation techniques used for creating a multihop ad hoc mesh infrastructure.

Framework implemented as a kernel module

This is one of the common approaches for wireless device-to-device sharing [1] [41]. Modern operating systems are not entirely monolithic and have support for loadable kernel modules. As shown in Figure 2.3, in this kind of approach, the framework is implemented as a kernel module that changes the IP layer of the operating system. The module is responsible routing and additional packet processing. The operating systems routing table is used to perform packet forwarding. This is the most efficient implementation technique since there is no need to copy packets between kernel and user space. However it is more difficult to debug and is tied to a specific operating system implementation. Since it is completely imple-

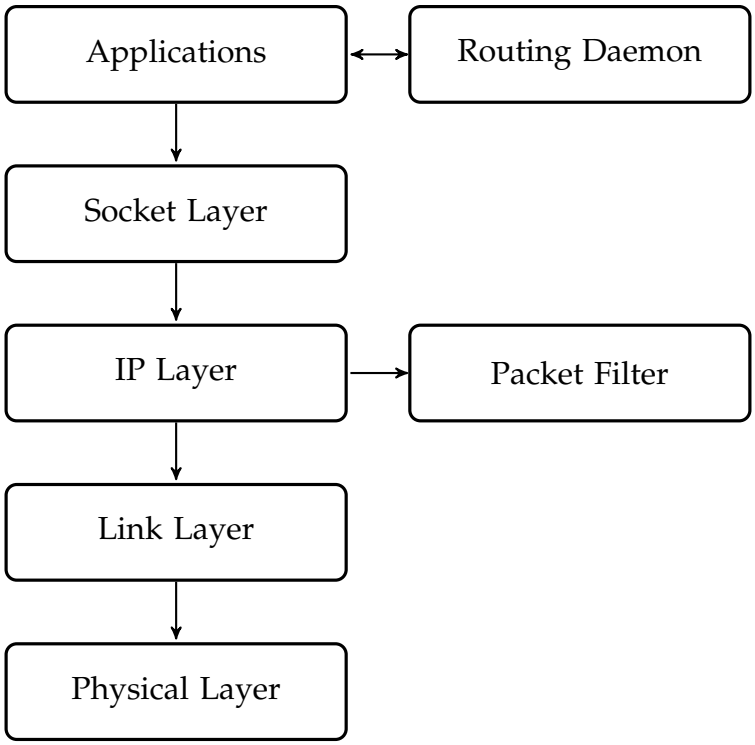

Figure 2.4: Framework implemented partly in kernel mode and user mode

mented in kernel mode a bug in the framework can bring the entire system down.

Framework implemented partly in kernel and user mode

In order to minimize the amount of kernel mode changes required and to make the application easier to port, some implementations [40] [44] make use of a small kernel mode driver and a user space driver to drive the kernel module. This reduces complexity and memory requirements but at the same time provide a great deal of control. Typically a daemon process in user space takes care of packet forwarding while packet sniffing is done by the kernel module. Packet sniffing tools such as Netfilter and Snoop [23] are used to intercept all the incoming and outgoing IP packets in the kernel space. These tools are typically implemented as a kernel

module. They identify many events that trigger a routing action. The packet sniffing tools callback functions are used to notify the routing daemon in user space about the occurred events. The routing daemon maintains routing table through IPC calls. The overall architecture for such hybrid implementations is shown in Figure 2.4.

Framework implemented completely in user space

The above approaches require a change in the host operating system. This is not always desirable and is tied to a specific implementation. In Android based systems the user must have elevated permissions to install a kernel mode driver and this facility is typically not provided to the user. However most of the functionality is can still be implemented in user space using the Bluetooth and Wi-Fi Direct services included in the Android SDK. Therefore the framework can be implemented as a user space application or a user space library as showin in Figure 2.5. This approach of implementation is more portable and easier to debug. The drawbacks of this approach is the inefficiency introduced due to copying data between kernel mode and user mode and the fact that applications may have to use a specific library dependency to accomplish various purposes. Another major drawback is the necessity of managing services using the shared wireless networking capability. Kernel-level sharing allows the sharing module(s) to share at the interface level, but user-level operation only allows application-level access to the packet contents and access to interface identifiers is not possible. Thus the middleware has to maintain the list services and devices within the neighborhood and perform encapsulations as needed to implement multi-hop forwarding and relaying.

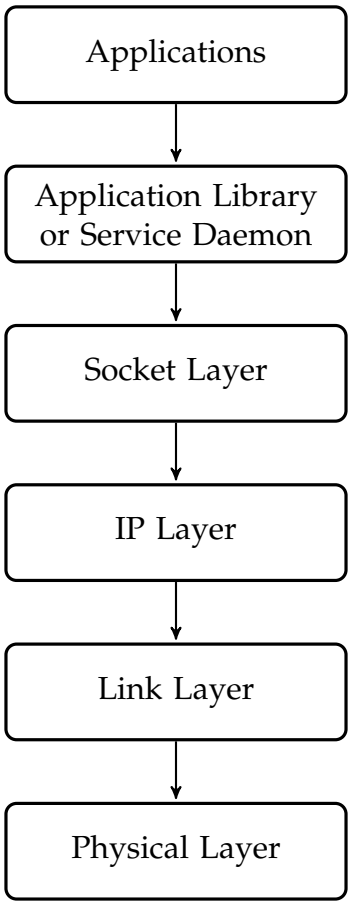

Figure 2.5: Framework implemented in user mode

CHAPTER 3
DESIGN AND PROTOTYPE IMPLEMENTATION OF USER-SPACE SERVICE SHARING

We implement the framework in a service oriented manner which involves service discovery and negotiation.

3.1 Service discovery

The application must be able to see what all services are available at a given point of time. This part of the application must gather the information about what neighboring nodes are willing to provide. In order to setup the initial topology the neighboring nodes must be paired either using Bluetooth or Wi-Fi Direct. When a node wants to avail a service it sends a service request packet to its neighbors (paired devices). The entire framework is implemented in the application layer so these packets are application layer packets. The discovery packet format is shown in Figure 3.1.

When a node wants to make use of service (say SMS) it will fill the Node Id field and Service Id field and send it to its neighbors. If the neighbor is able to service it, it will send the packet back to the requester with price or else it will forward to its immediate neighbors after updating the path. The requester then picks up a path based on the best price.

NODE ID	SERVICE ID
PATH	PRICE
PADDING	

Figure 3.1: Service Discovery Packet Format

NODE ID	SERVICE ID
PATH	TYPE
DATA	

Figure 3.2: Service Request Packet Format

3.2 Service request

After the service is discovered and the best path for the node is calculated, then node makes a service request by sending a packet to the best node providing that service. It makes use of the below packet format to make a service request. The service request packet format is described in Figure 3.2. For example if a node wants to make use of SMS service available at node *i*, it will fill up the path from the routing table to node *i*, set the price, set the service ID as SMS, and will have the destination number and text message to be sent in the data section.

3.3 Application design

The purpose of this application is to build a cooperative network of mobile users that share services in heterogeneous wireless radios. We make use of many of the open source libraries in our project. We implement our smart phone application on Android operating system which is popular on wide variety of platforms. We decided to implement the application entirely in user space due to following ben-

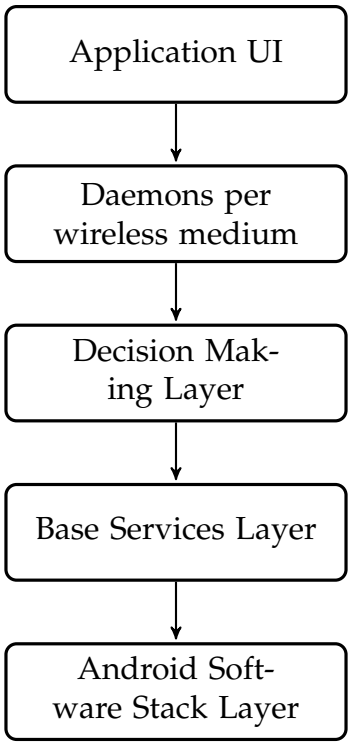

Figure 3.3: Layered architecture

efits.

- Ease of debugging and testing
- Portability
- Clean networking stack
- System less prone to failure as system components are not modified
- There is no need to root the phone

We designed the application as a layered application. The base service layers is responsible to atomic actions. Atomic actions involve those which can be done locally by the mobile device. Fetching a web page is such an action. We make use of the services provided by the Android SDK to accomplish these actions. The

decision making module decides if the given request needs to be forwarded, discarded or if the action needs to be done by current device. The decision making module therefore implements routing. We have a service running for each of the wireless medium under consideration. For example, if we have Wi-Fi and Bluetooth wireless mediums, we would have 2 service threads running. The function of this module is to enable connectivity with other nodes in the network and gets requests from neighboring nodes.

Android software stack and base services

The Android software stack is also a layered architecture. At the base level it makes use of 2.6 version of the Linux kernel. On top of the base UNIX like system Android provides native application libraries. The next layer is the dalvik virtual machine layer [8] which is custom JVM developed for Android specifically optimized for mobile application. The higher layer libraries are application layer frameworks. Our application makes use of the services provided by the application layer frameworks. The code listing below shows the implementation of SMS sending class. We have similar classes for fetching web pages, recording audio messages and for using the Android camera. The implementation can be found in [34].

```
public class SmsUtil
{
        public static void SendSms(String phoneNum, String msgText)
        {
                SmsManager smsManager = SmsManager.getDefault();
                smsManager.sendTextMessage(phoneNum, null, msgText,
                null, null);
        }
}
```

Figure 3.4: SMS sending implementation

Decision making module

The decision making module is responsible for all the decision making and forwarding. The algorithm of the decision making module is as depicted in the Figure 3.5. The algorithm fetches packets from the wireless daemons, if the packet is a response to a service query then it checks if the current routing table has a better path and if that is not the case the routing table is updated. If the packet is a service request and the packet is meant for the current node then it is serviced by the node otherwise the packet is forwarded after updating the path.

Daemons per wireless medium

These are service processes that responds to each application layer packet sent using a wireless medium. They do minor processing and pass it to the relevant decision making module if it is a service request. The Android API provides the concurrency mechanism in the form of Async Taks, Services and Threads. Async Tasks makes it easier to update the UI components once background processing is done. Async tasks are used in our application to update the ui components once processing is done. However, threads are better for longer running tasks. Since

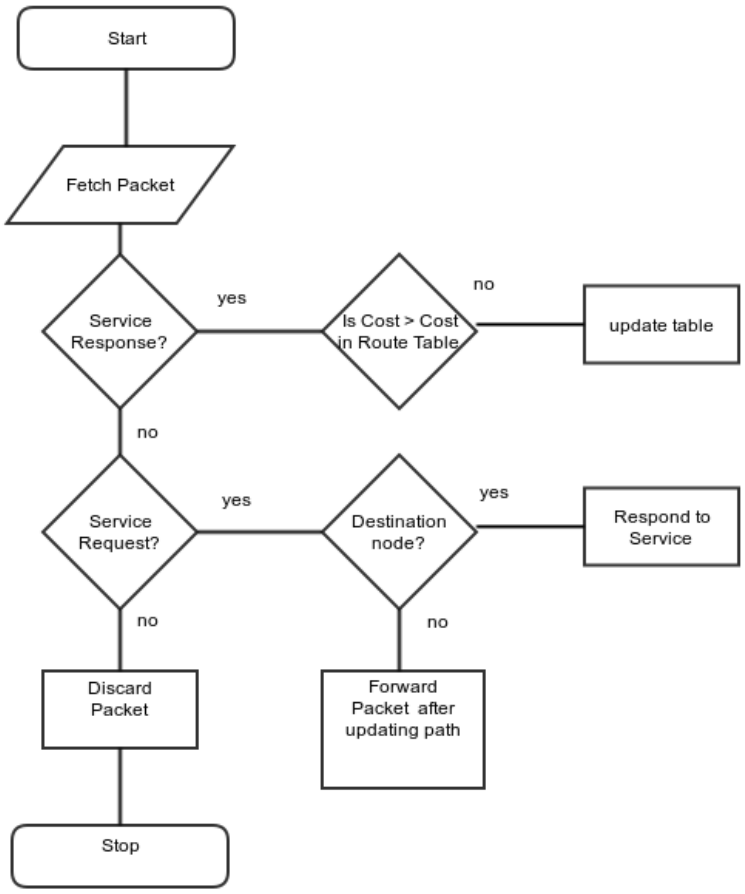

Figure 3.5: Decision making algorithm

Bluetooth has lower transfer rate, we make use of threads for data transfer. Services are also used for background processing but they provide an calling interface so that even other applications may use the service. We make use of all three of these mechanisms to get our job done. An Android 'Service' is suitable for long running tasks. Our Service is responsible for starting both Wi-Fi and Bluetooth daemons.

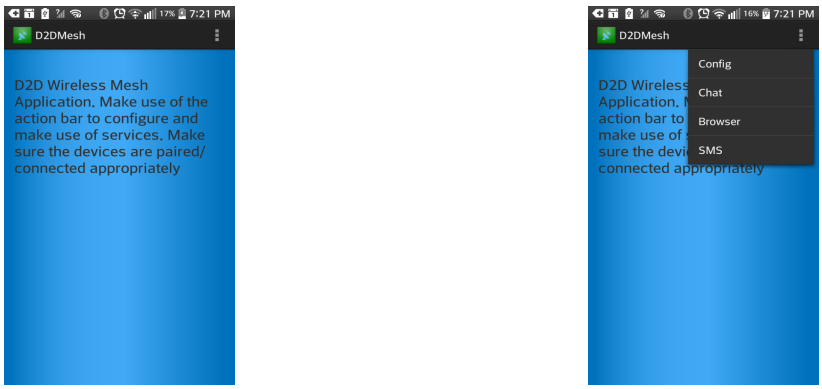

Figure 3.6: Screenshot showing the main screen and configuration screen

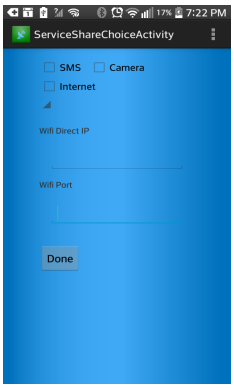

Figure 3.7: Screen shot showing configuration items

3.4 Application workflow

The main application window as depicted in Figure 3.4 shows basic help information on how to use the application and the action bar contains links to necessary sub components of the application. Choosing the config option form the main window activates the configuration activity which is used to configure which services needs to be shared and the route path for a particular service. The application provides user interfaces for SMS sharing, voice calls, video calling and multi-hop file transfer.

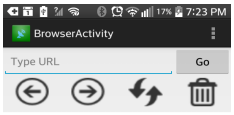

Figure 3.8: Screen shot showing Internet sharing using custom built browser

Sharing internet services

We have implemented a custom browser by using the WebView Android component. The inbuilt browser is not capable of using the proxy mechanism implemented in our framework, so we implemented our own browser capable of using the proxy mechanism in our framework. Figure 3.8 shows our browser in action.

3.5 Overview of the source code

Our implementation is centered on the previously presented layered design and is divided into java packages each of which corresponds to a layer in our architecture. We have incorporated libraries [28] for simplified IO operations, [33] for Bluetooth connection setup and transfer, [29] for easier Wi-Fi peer node identification and [32] for icons and UI design. The complete source to our implementation can be found in [34]. The packages we implemented are as follows:

- **com.d2dsq.baseservices**: This package corresponds to the base services layer. The classes in this package are responsible for atomic actions. The class *Sm-*

sUtil is responsible for SMS handling and *TinyHttp* is used for http handling.

- **com.d2dsq.models**: This package consists of shared data structures and network packet data structures used by the application. The service request and discovery packets are encapsulated using *Message* class in this package.
- **com.d2dsq.radio**: This package corresponds to daemons per wireless medium layer, and is responsible for creating threads for connection and data transfer. The classes *BluetoothUtil* , *ClientThread*, *ServerThread* corresponds to Bluetooth daemon while rest of the classes in the package deal with Wi-Fi daemon. The Wi-Fi daemon code is a modified version of [29] and Bluetooth daemon is a modified version of [33] so as to suit our purpose of working well in a hetrogenous environment better.
- **com.d2dsq.ui**: This package corresponds to the application UI layer in our architecture and provides the front end UI. The class names are self-explanatory and follow the naming convention that each class starts with the service being shared.
- **com.d2dsq.routing**: This package corresponds to the decision making layer in our architecture. The main class in this package is the *RoutingManager* which is responsible for maintaining the routing table. The *Node* class is responsible for holding network related details for a wireless node.
- **com.d2dsq.utils**: This package consists of utility classes that does not fit into any of the other categories. An example is the *SystemUIHider* class which is responsible for hiding some of the default Android UI elements when our application is running.

CHAPTER 4

PERFORMANCE BENCHMARKS

We made use of a variety of platforms to test our smart-phone application. We made use of the variety mobile phones running different versions of the Android operating systems (4.0 and above) for our testing purposes. The mobile phones used are Kyocera Rise, Google Nexus tablet, and LG Optimus L9.

4.1 Sample experiments

We base our experiments on scenarios where it could find applications in real world. One such scenario is the use of our application in remote infrastructure less locations where there are only few devices providing services to end users. This kind of scenario leads to a linear network topology. The other scenario where we performed our experiments is in a noisy environment where there are plenty of devices present. This generally leads to a star like network topology with a large number of inter connected nodes. The experiments were performed indoor in a room size of 10x10x5 meters. Inter node distance was 3 meters and there were no obstacles between them. The nodes were connected to a power source and power saving features provided by Android were disabled. These experiments were repeated fifty times and the average value was used. A typical user would use the middleware using the application UI but it is cumbersome to use the UI for measurement purposes. Therefore we used a modified application with traces added to measure duration and time of method invocation.

4.1.1 Service ratio variation

A critical issue that emerges from a multi-hop D2D sharing system at the user-level is the additional inefficiency because of multiple hops and the user-level design. We look at the service ratio that is defined as the ratio between the number of requests serviced to the totals number of requests sent to the application. We perform the experiment by creating a modified application that sends 100 dummy SMS requests. When the SMS request is received at the destination node, we record a trace log. In the first experiment we have nodes connected in a linear topology using Bluetooth. In the second experiment we have nodes connected using star topology. The variation of service ratio as the number of hops increases is shown in Figure 4.1.

We notice that service ratio is close to 90% even when the number of hops is five. This indicates that our application provides reliable multi-hop communication. This is due to the fact that we make use of reliable application layer protocols for transferring data. The service ratio in star topology is slightly lower than that of linear topology due to that fact that multiple nodes are sending packets at the same time to the same node which leads to congestion.

4.1.2 Delay measurements

We also measured the delay incurred due to overhead of the sharing framework. Delay here is the total time required for a small query application packet to reach the destination. Here also we make use of a dummy SMS service packet and trace the time using *System.currentTimeMillis* API call. When the dummy packet reaches

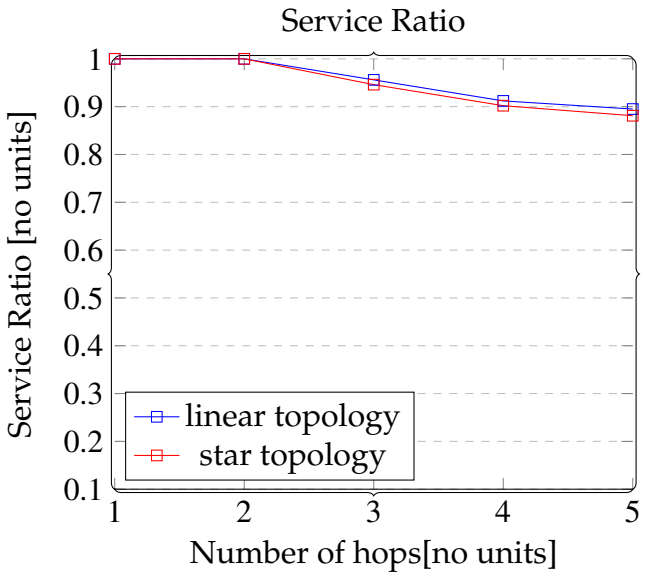

Figure 4.1: Service Ratio Variation in linear topology

the destination, we send it back again to the source, where the round-trip time is measured. The delay is measured as half of the difference in both time stamps. In this experiment all the nodes are connected via Bluetooth. The nodes were connected in a linear topology. The delay variation is shown in Figure 4.2.

We find that there is an average delay of 127 millisecond per node. The average inter node delay does not vary significantly as the number of hops increases. But the total end to end delay increases since these per node delays are cumulative.

4.1.3 Throughput in a single medium

We define throughput as the number of bits transferred per second. In our experiments we set up nodes in linear topology and make use of the file transfer interface to transfer a 100 MB file. We make use of traces and record the time taken to send the file and the time taken for the file to be completely downloaded. Throughput

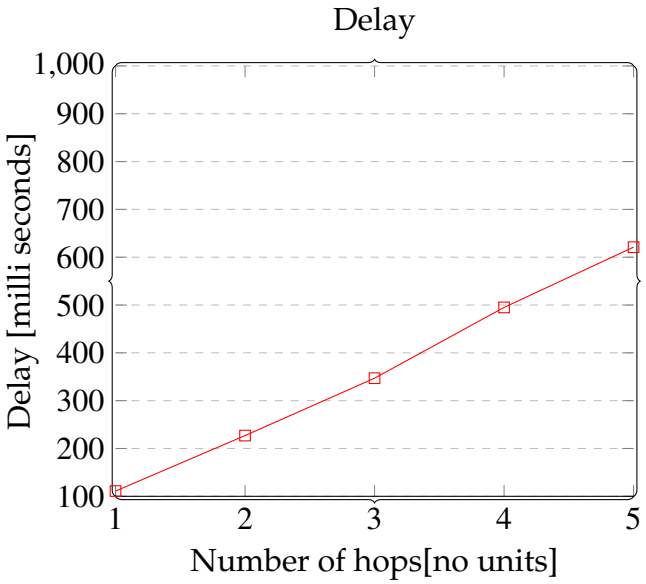

Figure 4.2: Delay Measurement

is then calculated as the total bytes transferred to total time taken. We perform the experiments using a single wireless medium as well as in a heterogeneous wireless medium. The throughput variation of Bluetooth medium is shown in Figure 4.3 and Wi-Fi medium in Figure 4.4.

We find that there is an average 30% reduction in the throughput as the number of hops increases. It can be observed that Wi-Fi offers much higher throughput than Bluetooth medium. However, even up to three hops, in the case of Wi-Fi medium, the quality of service is good enough for live video streaming [35].

4.1.4 Throughput over heterogeneous wireless medium

In this experiment we start with two Wi-Fi nodes connected to each other. We add nodes to the end of this network so as to maintain linear topology. The throughput

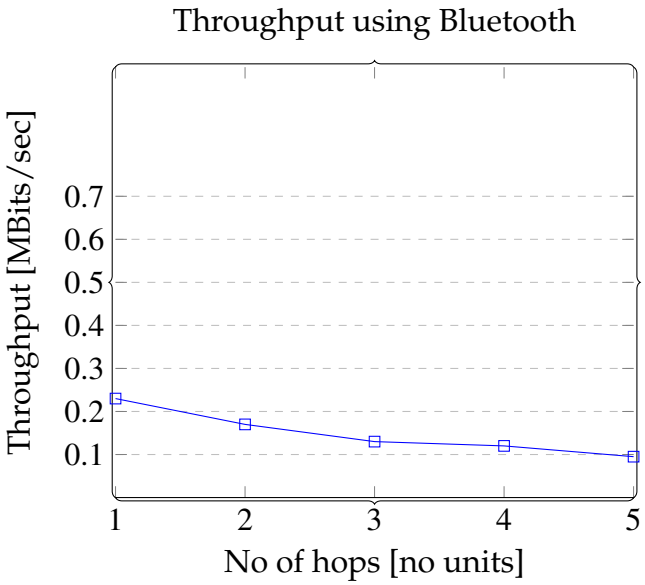

Figure 4.3: End to End throughput measurement using Bluetooth

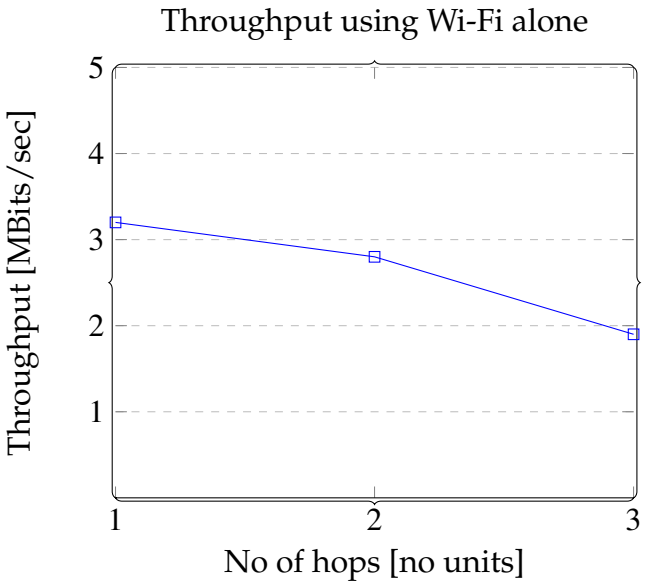

Figure 4.4: End to End throughput measurement using Wi-Fi

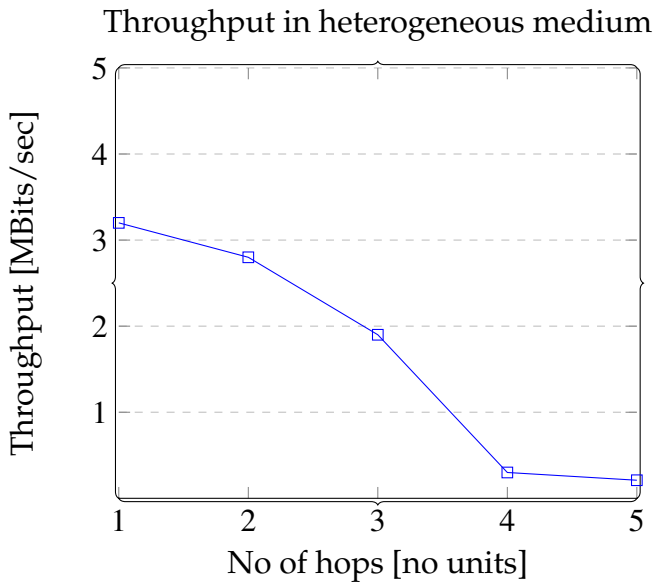

Figure 4.5: End to end throughput variation in heterogeneous medium

is measured end to end between the starting node and the recently added node. The first three nodes are connected using Wi-Fi, but rest of the links are Bluetooth. The throughput variation is shown Figure 4.5. It can be seen that adding Bluetooth nodes have an adverse effect on the throughput since Bluetooth throughput is lower and affects the total throughput. The lower throughput rate due to Bluetooth links is attributed to that fact that commodity mobile phones generally come with class 2 radios [5] whose transfer rate is not high.

4.1.5 Increase in wireless range

In this experiment we measure the wireless range by checking if we are able to perform the chatting at a particular distance range. Wi-Fi Direct implementation in stock Android restricts the number of hops to two so we make use of Bluetooth connections to improve the range. This experiment was performed outdoors

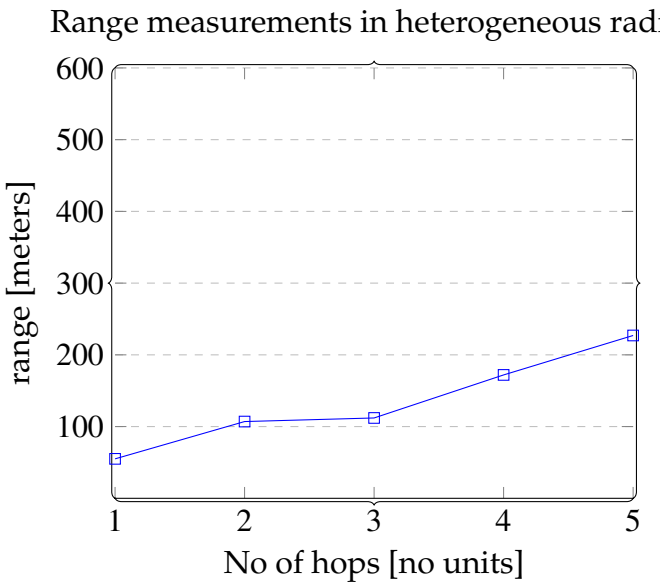

Figure 4.6: Range measurements in heterogeneous radios

such that the nodes have 100% battery capacity and the power saving features of Android were disabled. We made sure that there are no obstacles between neighboring nodes. We make use of a linear topology and the third node is connected to second node using Bluetooth.

We observe that Wi-Fi radio in commodity hardware provides a range of 52 meters while the Bluetooth radio provides a range of 5 meters on an average. Hence, adding a Wi-Fi node increases the range of the network by 52 meters while adding a Bluetooth node increases it by 5.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In this thesis, we implement a heterogeneous ad hoc multi-hop networking framework for Android operating system completely in user space. By completely implementing the sharing and negotiation logic in user space, we made it very portable and easy to deploy. The application interface is designed to be simple. Our application provides heterogeneous ad-hoc service sharing for Android and it can be extended for development of other collaborative MANET applications. Since mobile devices has constrained resources, we focused on reducing its complexity and improving its efficiency. To evaluate the performance and feasibility of our application, we have done experiments to measure the delay, service ratio and throughput.

By taking advantage of idle network resources and services, our framework offers higher utilization. The implementation of the framework is still at a prototype stage, there is still some issues that need to be solved and improvements could be done in the future as a continuation of our current work. The initial pairing of devices is manually chosen by the user which makes initial setup inconvenient. We need a way to make it transparent so that end user does not have to perform the initial pairing.

The current implementation does not identify and punish malicious and misbehaving nodes. A malicious node may perform a denial of service attack by flooding the network with unwanted service requests. A node may also act like a leach where it does not contribute anything to the network but utilizes all the network resources. A reputation based approach can be used to punish misbehaving nodes

so that they do not drain the network.

The application improves spectrum utilization, hence improves the overall network utilization. Therefore, there is a need for an incentive mechanism so that application may be more ready for commercial adoption by carrier providers. The application as such does not require anything more than a commodity smartphone. The middleware is completely implemented in user space, therefore the end user may install the application without worrying about breaking the terms of use agreement.

The current framework is mostly written in java. As mentioned before, JVM introduce some inefficiency. Therefore it is possible to further improve the efficiency of our system by rewriting some of the most performance-critical functions in C or C++ with Androids NDK. The native code runs faster hence that would help in reducing the higher processing overhead of the current implementation. It is also possible to implement the features of this framework as a kernel module so that performance and transparency can be improved greatly. A patched version of Android could be made so that it has service sharing capabilities built into the operating system itself.

BIBLIOGRAPHY

[1] Implementation of dynamic source routing. *Internet: http://monarch.cs.rice.edu/dsr-impl.html*, 2002.

[2] N Asokan, Alexandra Dmitrienko, Marcin Nagy, Elena Reshetova, Ahmad-Reza Sadeghi, Thomas Schneider, and Stanislaus Stelle. Crowdshare: Secure mobile resource sharing. In *Applied Cryptography and Network Security*, pages 432–440. Springer, 2013.

[3] Zubin Bharucha, Harald Haas, Andreas Saul, and Gunther Auer. Throughput enhancement through femto-cell deployment. *European Transactions on Telecommunications*, 21(5):469–477, 2010.

[4] Zubin Rustam Bharucha. Ad hoc wireless networks with femto-cell deployment: a study. 2010.

[5] Chatschik Bisdikian et al. An overview of the bluetooth wireless technology. *IEEE Commun Mag*, 39(12):86–94, 2001.

[6] Mario Bisignano, Giuseppe Di Modica, and Orazio Tomarchio. Jmobipeer: a middleware for mobile peer-to-peer computing in manets. In *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, pages 785–791. IEEE, 2005.

[7] S Bluetooth. Rfcomm with ts 07.10. *Bluetooth SIG*, 2003.

[8] Dan Bornstein. Dalvik vm internals. In *Google I/O Developer Conference*, volume 23, pages 17–30, 2008.

[9] Scott Burleigh, Adrian Hooke, Leigh Torgerson, Robert Durst, Keith Scott, Kevin Fall, and Howard Weiss. Rfc4838-delay-tolerant networking architecture, 2007.

[10] Hillary Caituiro-Monge, Kevin Almeroth, and Maria del Mar Alvarez-Rohena. Friend relay: a resource sharing framework for mobile wireless devices. In *Proceedings of the 4th international workshop on Wireless mobile applications and services on WLAN hotspots*, pages 20–29. ACM, 2006.

[11] Claudia Canali, Maria Elena Renda, Paolo Santi, and Simone Burresi. Enabling efficient peer-to-peer resource sharing in wireless mesh networks. *Mobile Computing, IEEE Transactions on*, 9(3):333–347, 2010.

[12] Ranveer Chandra. *MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card*. in IEEE INFOCOM, Hong Kong, 2004.

[13] John Chapin and William Lehr. Mobile broadband growth, spectrum scarcity, and sustainable competition. TPRC, 2011.

[14] Nasim Chowdhury. Olsr in android operating system. 2013.

[15] Gang Ding and Bharat Bhargava. Peer-to-peer file-sharing over mobile ad hoc networks. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 104–108. IEEE, 2004.

[16] Ovidiu Valentin Drugan, Thomas Plagemann, and Ellen Munthe-Kaas. Building resource aware middleware services over manet for rescue and emergency applications. In *Personal, Indoor and Mobile Radio Communications, 2005. PIMRC 2005. IEEE 16th International Symposium on*, volume 2, pages 816–820. IEEE, 2005.

[17] firefighterclosecalls.com. Sources: Radios did not work in dc metro fatal fire, 2014.

[18] Open Garden. Open garden web page, 2013.

[19] Paul Gardner-Stephen and Swapna Palaniswamy. Serval mesh software-wifi multi model management. In *Proceedings of the 1st International Conference on Wireless Technologies for Humanitarian Relief*, pages 71–77. ACM, 2011.

[20] Arne John Glenstrup, Michael Nielsen, Frederik Skytte, and Arnar Gunason. Real-world bluetooth manet java middleware. Technical report, IT-Universitetet i København, 2009.

[21] RS Gohs. *Beddernet–Bluetoth Scatternet Framework For Mobile Devices*. PhD thesis, Masters thesis, IT University of Copenhagen, 2010.

[22] Antônio Tadeu A Gomes, Artur Ziviani, Luciana S Lima, and Markus Endler. Dichotomy: A resource discovery and scheduling protocol for multihop ad hoc mobile grids. In *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, pages 719–724. IEEE, 2007.

[23] Ms Prinima Gupta and RK Tuteja. Design strategies for aodv implementation in linux. *IJACSA Editorial*, page 102, 2010.

[24] Myron Hattig. Zero-conf ip host requirements. *draft-ietf-zeroconf-reqts-09. txt, IETF MANET Working Group*, 2001.

[25] Guoyou He. Destination-sequenced distance vector (dsdv) protocol. *Networking Laboratory, Helsinki University of Technology*, pages 1–9, 2002.

[26] http://project byzantium.org/. Project byzantium.

[27] https://code.google.com/p/android-wifi tether/. Wifi tehter for root users.

[28] https://commons.apache.org/proper/commons io/download.cgi. Apache io library.

[29] https://github.com/albertcbraun/WifiDLite. Wifidlite.

[30] https://github.com/n8fr8/gilgamesh. pinwheel messenger.

[31] https://github.com/ProjectSPAN/android-manet manager. Project span.

[32] https://github.com/rubeus90/WonderCom. wondercom.

[33] https://github.com/simonguest/android btxfr. android-btxfr.

[34] https://github.com/ThomasMadappattu/P2PSQ. D2d service sharing.

[35] https://help.netflix.com/en/node/306. Netflix minimum requirements.

[36] http://villagetelco.org/. village telco.

[37] George Iosifidis, Lin Gao, Jianwei Huang, and Leandros Tassiulas. Enabling crowd-sourced mobile internet access. In *INFOCOM, 2014 Proceedings IEEE*, pages 451–459. IEEE, 2014.

[38] David Johnson, Ntsibane Ntlatlapa, and Corinna Aichele. Simple pragmatic approach to mesh routing using batman. 2008.

[39] Rabie Khodr Jradi and Lasse Seligmann Reedtz. Adhoc on android. *Internet: http://code. google. com/p/ad-hoc-on-android*, 2010.

[40] Vikas Kawadia, Yongguang Zhang, and Binita Gupta. System services for implementing ad-hoc routing protocols. In *Parallel Processing Workshops, 2002. Proceedings. International Conference on*, pages 135–142. IEEE, 2002.

[41] L Klein-Berndt. Kernel aodv from national institute of standards and technology (nist), 2010.

[42] netfilter.org. Netfilter.

[43] Andrés Neyem, Sergio F Ochoa, José A Pino, and Luis A Guerrero. Sharing information resources in mobile ad-hoc networks. In *Groupware: Design, Implementation, and Use*, pages 351–358. Springer, 2005.

[44] Erik Nordstrom and B Wiberg. Aodv-uu implementation.

[45] oracle. J2me.

[46] Charles Perkins, Elizabeth Belding-Royer, Samir Das, et al. Rfc 3561-ad hoc on-demand distance vector (aodv) routing. *Internet RFCs*, pages 1–38, 2003.

[47] Marshall Rose. The blocks extensible exchange protocol core. 2001.

[48] Gabriel Sandulescu and Simin Nadjm-Tehrani. Opportunistic dtn routing with window-aware adaptive replication. In *Proceedings of the 4th Asian Conference on Internet Engineering*, pages 103–112. ACM, 2008.

[49] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160, 2001.

[50] Niranjan Suri, Massimiliano Marcon, Raffaele Quitadamo, Matteo Rebeschini, Marco Arguedas, Stefano Stabellini, Mauro Tortonesi, and Cesare Stefanelli. An adaptive and efficient peer-to-peer service-oriented architecture for manet environments with agile computing. In *Network Operations and Management Symposium Workshops, 2008. NOMS Workshops 2008. IEEE*, pages 364–371. IEEE, 2008.

[51] CyanogenMod Team. Cyanogenmod.

[52] Narseo Vallina-Rodriguez, Christos Efstratiou, Geoffrey Xie, and Jon Crowcroft. Enabling opportunistic resources sharing on mobile operating systems: Benefits and challenges. In *Proceedings of the 3rd ACM workshop on Wireless of the students, by the students, for the students*, pages 29–32. ACM, 2011.

[53] Tiancheng Zhuang, Paul Baskett, and Yi Shang. Managing ad hoc networks

of smartphones. *International Journal of Information and Education Technology*, 3(5):540, 2013.