A Device-to-Device Service Sharing Middleware for Heterogeneous Wireless Networks

Mostafizur Rahman, Sandeep Mathew, Murat Yuksel, and Shamik Sengupta
Dept. of Computer Science and Engineering, University of Nevada, Reno, USA
mostafizurrahmantopu@nevada.unr.edu, sandeep.sandos@gmail.com, yuksem@unr.edu, ssengupta@unr.edu

***Abstract*—Wireless devices with diverse capabilities are ubiquitous and will continue to flourish for the next foreseeable future. With cellular connectivity, each device is capable to utilize more necessary services. However, in some cases (e.g., public safety and disaster recovery) cellular connectivity may become unavailable. In these situations, device-to-device (D2D) communication can contribute to maintain connectivity as well as providing other services in terms of service sharing (e.g., SMS, Internet, camera) available in individual devices. This paper describes an approach where users can share wireless services with other users in a seamless manner. We develop a smart phone application, D2DMesh, for service sharing in a D2D manner. By using D2DMesh, users can get more work done via sharing, increase the spectrum utilization, create their own private networks without a centralized infrastructure and offload network traffic to a less congested network path. Our approach is platform-independent and is entirely implemented in user space. It allows sharing of services among neighboring devices over multiple hops without any help from the device vendors. We present results from initial experiments and show the effects of various important parameters on the performance of D2D service sharing.**

Keywords*—*D2D, Service-Sharing, Multihop, Infrastructureless, Heterogeneous, Wireless, Middleware.

I. Introduction

Wireless devices offer a variety of features and services that are unique to each and every device. Capability to reach a service wirelessly has become crucial to our lives in a lot of settings. Being able to check the latest news, to access financial information, to send message to colleagues are all considered to be given in our daily lives. Further, we rely heavily on these wireless devices for emergency situations like 911 calls or reporting of a threat for public safety. Yet, the spectrum band, the key resource behind the wireless services we enjoy, is under a crunch as wireless demand is growing exponentially [1]. According to recent reports [2], wireless traffic demand has already exceeded the cellular spectrum capacity. It has become clear that the cellular capacity itself will not be enough to accommodate the future wireless needs. Device-to-device (D2D) communication for service sharing can be a solution for such cases with or without the presence of regular cellular network.

However, D2D communications (and hence sharing) requires (i) *protocols* that enable seamless D2D networking and (ii) clear *incentives* for device owners to share their devices with others. The focus of this paper is to offer a solution to the former issue while indicating a few ideas for the latter. The main motivation of our work is to address the concern of enabling communication facilities in environments where a central communication infrastructure cannot be relied upon. Complete dependence on infrastructure can be a major bottleneck during disasters or emergency situations. In disaster scenarios, the central infrastructure facilities may be unavailable [3], 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. D2D sharing of wireless resources is a great way of circumventing this dependence on infrastructure. In this paper, we have developed an approach that extends the notion of D2D sharing to "service sharing" and performs the sharing at the user-level over multiple hops involving a heterogeneous set of wireless links. Our approach may pave the way for a model in which end users can sell their own services to other end users and establish a D2D service sharing market beneficial to the entire user set. Our approach also makes use of heterogeneous radios further improving the spectrum utilization by offloading network traffic to less congested wireless paths.

To attain seamless D2D networking, the community has employed tethering in devices. Sharing of the cellular Internet connectivity is already available in most mobile devices via tethering, but such sharing is essentially single hop and typically works only with homogeneous links. More importantly, current tethering techniques are limited by the carrier's permission and some impose an additional fee for tethering. Such D2D sharing is device-specific due to *kernel-level* implementation and is not agile to multiple carriers. Further, existing tethering solutions only offer a WiFi hotspot and do not use different wireless interfaces (e.g., Bluetooth, WiFi and 3G together) to extend the reach of D2D sharing. Our, smartphone app, D2DMesh enables seamless sharing of device services such as SMS text messaging, camera and 3G data. It is a user-level middleware framework for sharing services of devices in a neighborhood. The middleware abstracts devices' capabilities as service marketable to other devices in a neighborhood and allows dynamic gluing of heterogeneous wireless interfaces for D2D sharing.

The rest of the paper is organized as follows: We first survey other D2D wireless sharing techniques in Section II. We then detail the D2D sharing middleware and its implementation in the user space in Section III. We present a particular smartphone app, D2DMesh, using the user-level middleware concepts in Section IV. Section V presents results from an initial experimental evaluation of D2DMesh. Finally, Section VI summarizes our work.

978-1-4673-9882-4/16/$31.00

II. Related Work

Device-to-device (D2D) communication is a new paradigm for next generation mobile networks. User-space application service sharing between devices has the potential to play a vital role in future means of communication. Earlier implementations of D2D service sharing over heterogeneous links require bypassing restrictions enforced by vendors and device configuration needs to be modified by executing commands at an elevated privilege level. Many different cross-layer designs were explored with the assumption of a privileged access. Effects of cross-layer medium access control (MAC) designs and location awareness are considered to improve efficiency of corresponding wired protocols in [4]. Automatic publishing, discovery, configuration, monitoring and control also have been considered for single hop communication in [5] and [6]. The benefits and challenges associated with sharing resources on mobile operating systems have been discussed in [7]. The architecture of a resource-aware middleware over mobile ad-hoc networks for rescue and emergency application has been described in [8]. Resource discovery, scheduling and picking up the most resourceful node in multi-hop ad-hoc grids so that messaging overhead between devices can be minimized has been described in [9].

Vendors of Android smartphones and tablets have implemented tethering facilities in their firmware [10]. Open Garden developed an app FireChat [11], which offers Wi-Fi hot-spots and Bluetooth tethering and allows sharing 3G/4G Internet connection between devices running different operating systems and assumes no tether fees. FireChat only provides the chat option to the users in multi-hop level as D2D manner when there is no Internet connectivity. Serval Mesh [12] is a telephony platform that uses ad-hoc Wi-Fi capability of the devices to form a wireless mesh network and allows the phone to perform voice calls, short messages and other modes of communication without a central carrier facility. However, these prior researches mainly focus on single hop communication.

Heterogeneity of devices and resources has been explored as well. Resource sharing such as document transfer has been described in [13] and makes use of commodity laptops instead of smartphones. Multinet [14] facilities simultaneous connections to multiple networks, but works only for the Windows platform and does not support multi-hop communication. Our approach is an improvement of these scenarios, where it can share services between devices over heterogeneous wireless links via multiple hops. It works entirely in user space, which eliminates the necessity of rooting devices so that end user license agreement (EULA) is not affected.

III. User-Level Service Sharing Middleware

We consider two major scenarios for as motivating cases for our design of user-level service sharing:

Scenario 1: Simple sharing of services. This scenario is very similar to a barter system in which two cooperating nodes with dissimilar services are able to utilize both the services. From Fig. 1, Joe has unlimited SMS service and Sando has unlimited Internet access via 3G service. After the sharing of services, both Joe and Sando gets unlimited Internet access and unlimited SMS respectively.

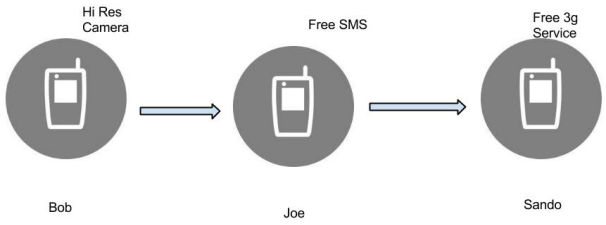

Fig. 1: Simple sharing of services

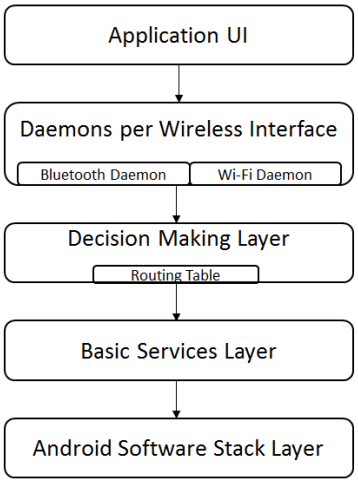

Fig. 2: Layered architecture for D2D service sharing middleware

Scenario 2: Increase in the overall network range by co-operating. In this scenario, a node is able to access a service from a node, outside of its range by using a second node as a relay. From Fig. 1, Sando is not in range of Bob. But he is able to access the high resolution camera of Bob by using Joe as relay. This increases the overall network range for Sando.

A. Application Stack

Our application design is a layered architecture. Fig. 2 shows the layers of the application. The base service layer is responsible for atomic actions. Atomic actions involve those services which can be done locally by the mobile device, e.g., fetching web pages. We make use of the services provided by the Android SDK to accomplish these actions. The decision making module decides to forward, discard or process by the current node for any given request. The decision making module therefore implements routing. We have a service running for each of the wireless mediums under consideration. For example, if we have Wi-Fi and Bluetooth wireless mediums, we would have two service threads running. The function of this module is to enable connectivity with other nodes in the network and gets requests from neighboring nodes.

1) Android Software Stack and Base Services: The Android software stack is also a layered architecture. At the base it uses Linux 2.6 kernel. On top of the base UNIX like system

NODE ID	SERVICE ID
PATH	PRICE
PADDING	

Fig. 3: Service Discovery Packet Format

NODE ID	SERVICE ID
PATH	TYPE
DATA	

Fig. 4: Service Request Packet Format

Android provides native application libraries. The next layer is the dalvik virtual machine layer [15], which is custom JVM developed for Android, specifically optimized for mobile application. The higher layer libraries are application layer frameworks. Our application uses the services provided by the application layer frameworks to perform basic actions, e.g., SMS service.

2) Decision Making Module: The decision making module is responsible for all the decision making and forwarding. The algorithm fetches packets from the wireless daemons, if the packet is a response to a service query then at first it looks up the current routing table for a better path. If it does not contain such path then it updates the routing table. 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. This module mainly handles packet forwarding and access of heterogeneous wireless interfaces.

3) Daemons per Wireless Medium: Our service is responsible for starting both Wi-Fi and Bluetooth daemons. These daemons are service processes that respond to each application layer packet sent using a wireless medium. They do minor processing and pass it to the relevant decision making module in case of service request. The Android API provides the concurrency mechanism in the form of Async Tasks, Services and Threads. We have used all of them in our application. An Android 'Service' is suitable for long running tasks.

B. Application Service

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

1) Service Discovery: The application must be able to see services 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, we have considered to pair the neighboring nodes either using Bluetooth or Wi-Fi Direct before starting any service discovery from any device. 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 Fig. 3.

When a node wants to get a service (i.e. SMS) it will fill the Node Id field and Service Id field and send it to its neighbors. If the neighbor is able to serve, it will send the packet back with cost(price) or it will forward to its immediate neighbors after updating the path. The requesting node then picks up a path based on the best price.

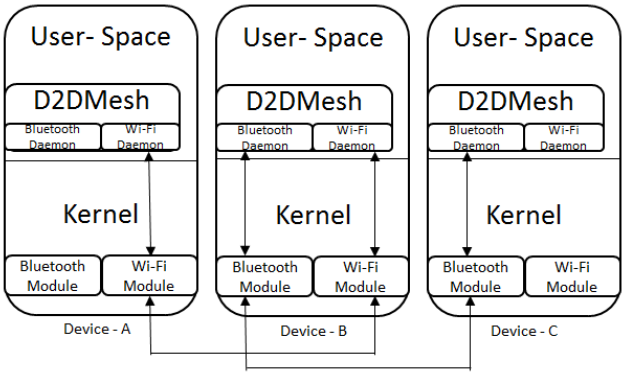

Fig. 5: High Level Design

2) Service Request: After the service is discovered and the best path for the node is calculated, then each node makes a service request by sending a packet to the best node providing that service. The service request packet format is described in Fig. 4. For example, if a node wants to use the SMS service available at node X, it will fill up the path from the routing table to node X, set the price, set the service ID as SMS along with the destination number and text message to be sent in the data section.

From high level view, the application design can be summarized using the Fig. 5, where a service available in device C can be served to device A via the intermediate device B. The application uses the Android services to interact with wireless radios. When a packet gets sent to the peer node, it is collected by the device driver and sent to the application layer. The application then routes the packet appropriately using the services provided by the operating system.

IV. D2DMesh

Our implementation can be found in [16]. The package ui implements UI, base service implements base services, radio implements daemons and routing implements routing. Fig. 6 illustrates the high level interaction between working modules of our application. Unlike other approaches, in-kernel routing tables are not modified and additional configurations are not done, the application works completely in user space using the services provided by the Android Operating System. This eliminates the necessity of rooting devices. The Fig. 7 shows the main application screen where the user can use the menu bar to choose the appropriate services. Once the user selects a service (i.e. SMS), the appropriate interface for using the service is invoked. The Android API provides methods for finding the peers using Wi-Fi as well as Bluetooth. Initially the routing table is empty and two service threads are running on the application for Wi-Fi as well as Bluetooth. The Android

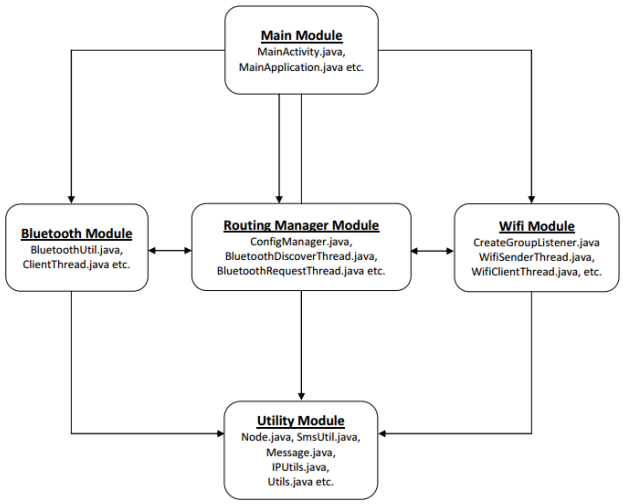

Fig. 6: High Level Interactive Diagram

platform forces I/O activities to be on threads that are separate from the UI thread. Therefore, we have threads for transferring data packets to peer nodes. Lets, Consider a scenario as in Fig. 1 where Device#1 has unlimited 3G service, Device#2 has high resolution camera and Device#3 has unlimited SMS. The routing table from each devices perspective has been shown in Fig. 8. We use a hash map for implementing the routing table. The service id, path and cost corresponds to the service being used, path taken to reach the capable node for specific request and the number of hops to reach the services. The routing table is initially empty. The discovery packets are sent when the application is initially started and when a configurable timer expires upon which the routing table is cleared. The routing table is updated only when a response to a service discovery arrives at the source node which has lesser cost. The routing table is used for populating the service request packets.

V. Experimental Evaluation

We used Google Nexus Phone with Android operating system(version 5.0.1) for our experimental purposes. We focus our experiments on scenarios where it can find applications in real world. One such scenario is the usage 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. We performed our experiments 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. It was an indoor experiment with the 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 to measure duration and time of method invocation.

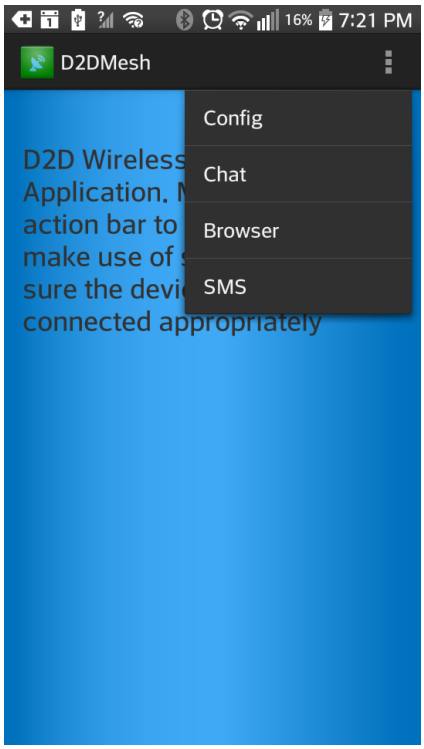

Fig. 7: Application UI

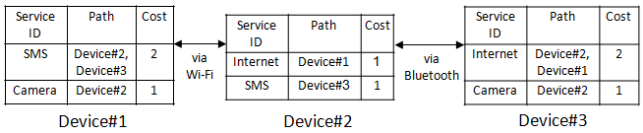

Fig. 8: Routing tables

A. Service ratio

A critical issue emerges in multi-hop D2D service sharing at the user-level is the service ratio, because of involvement of multiple hops. We defined service ratio as the ratio of the number of requests served to the total number of requests received. We performed 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 to compute the ratio. 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 ratio with the increase of hops is shown in Fig. 9.

We noticed that, the ratio is closer to 90% even when the number of hops is five. This indicates that our application provides reliable multi-hop communication. This was due to the use of reliable application layer protocols for transferring data. The ratio in star topology is slightly lower compared to the linear topology. Because, multiple nodes are sending packets to the same destination simultaneously which leads to congestion.

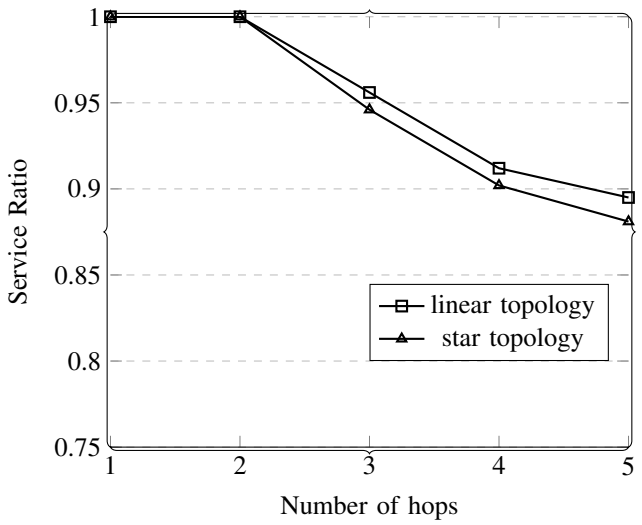

Fig. 9: Service ratio over a linear topology

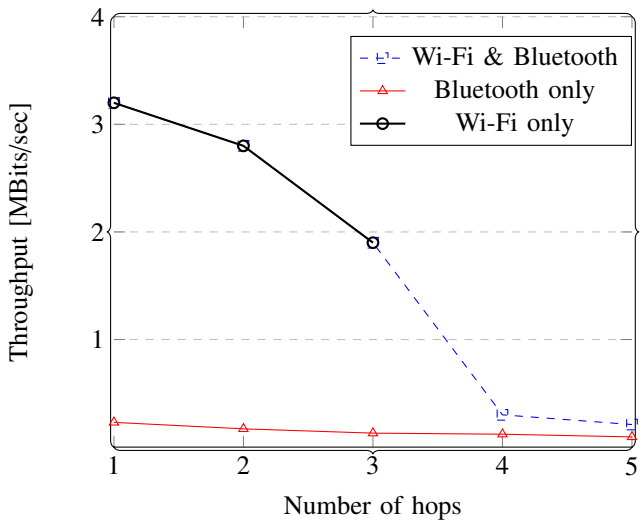

Fig. 10: End-to-end throughput

B. System throughput

We define throughput as the number of bits transferred per second. In our experiments, we set up nodes in linear topology and 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 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 over homogeneous links (i.e., Bluetooth only or Wi-Fi only) in Figure 10. 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. [17].

We also experimented with heterogeneous links where first three nodes were connected by Wi-Fi using linear topology and other nodes were connected with them via Bluetooth to the end while maintaining the topology. Then we calculated end to end system throughput over heterogeneous links. The throughput variation is shown in Fig. 10. It has been seen that, added Bluetooth nodes adversely effect on the throughput since Bluetooth link has lower throughput compared to Wi-Fi. Commodity mobile phones generally come with class 2 radios [18] where Bluetooth connection has low transfer rate.

C. Increase in wireless range

In this experiment we measured the wireless range by chatting between devices located at different distances. Implementation of Wi-Fi Direct in stock Android restricts the number of hops to two, so we used Bluetooth connection for third node to improve the range. This experiment was performed in open space with 100% battery capacity and the power saving features of Android were disabled in each device.

We observed that Wi-Fi radio in commodity hardware provides a range of 52 meters while the Bluetooth radio

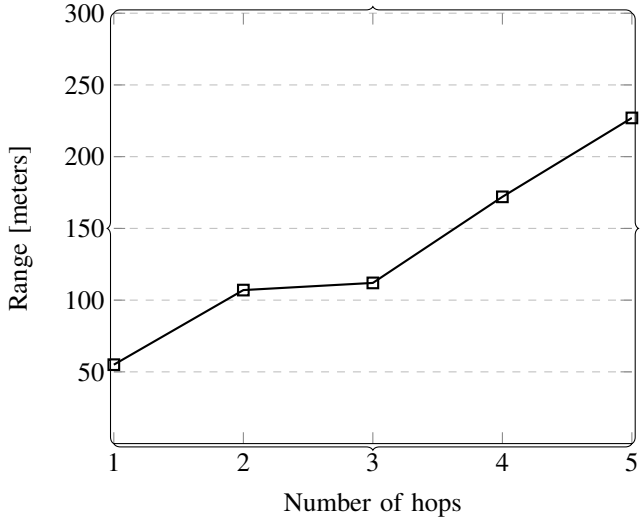

Fig. 11: Range measurements in heterogeneous radios

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. The range measurements are shown in Fig. 11.

VI. Summary and Future Work

In this paper, we have implemented a heterogeneous multi-hop ad-hoc networking framework in user space for Android operating system which is easy to deploy and can be extended for development of other collaborative MANET applications. By taking advantage of idle network resources and services, our framework offers higher network utilization by making use of device-to-device (D2D) communication.

Moreover, D2DMesh is a user-space application and users need not to be worried about violating the EULA. The service sharing approach of using D2D communication in user-space has great potential. Since it abstracts the device resources as

a service, it can trigger a D2D market where users share their device resources in return of a payment or reward. Although this approach is likely to conflict with carrier's data plan expectations, it is still applicable to device resources such as camera and other sensing capabilities. Particularly for larger good (e.g., public safety), users will have the incentive to share their device resources.

The implementation of our framework is still in early prototype stage, there are many issues yet to be solved. If the initial pairing could be done automatically then the end users do not require to perform any configuration manually. Further, the current implementation works in certain Android versions(e. g. ver. 5.0.1) and phones, and more work is needed to improve the prototype for a larger and better coverage of all Android versions and phones. We are working on the application to enhance it further and make it more stable.

Future work should focus on a scheme for identifying and punishing malicious nodes which drain the network. In a longer run, an incentive mechanism can be introduced to motivate carriers as well as the end users for commercial adoption. At lower levels, a patched version of Android can be made which has service sharing capabilities built into the operating system itself. Such an effort will obviously require rooting of the devices and may not be easy to realize. Yet, it may prove to be a fruitful approach in convincing device vendors and carriers if significant benefits in spectrum utilization are attained. Either at the user-space or the kernel-space, several technical challenges lie ahead: (i) Effective buffering mechanisms to handle speed imbalances among heterogeneous wireless interfaces of a device, (ii) Dynamic routing among devices so as to share over multi-hop and ad-hoc topologies of devices, (iii) congestion control among devices to avoid overloading a particular set of devices, and (iv) topology establishment to minimize interference.

Acknowledgements

This work was supported in part by United States NSF awards AST-1444077, CNS-1321069 and United States NSF CAREER award CNS-1346600.

References

[1] J. Chapin and W. Lehr, "Mobile broadband growth, spectrum scarcity, and sustainable competition." TPRC, 2011.

[2] Cisco, "Global Mobile Data Traffic Forecast Update, 2011-2016," pp. 1–29, Feb. 2012, White Paper.

[3] http://www.firefighterclosecalls.com/news/fullstory/newsid/216435, "Sources: Radios did not work in dc metro fatal fire," 2014.

[4] C. Canali, M. E. Renda, P. Santi, and S. Burresi, "Enabling efficient peer-to-peer resource sharing in wireless mesh networks," *Mobile Computing, IEEE Transactions on*, vol. 9, no. 3, pp. 333–347, 2010.

[5] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[6] N. Suri, M. Marcon, R. Quitadamo, M. Rebeschini, M. Arguedas, S. Stabellini, M. Tortonesi, and C. 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*. IEEE, 2008, pp. 364–371.

[7] N. Vallina-Rodriguez, C. Efstratiou, G. Xie, and J. 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*. ACM, 2011, pp. 29–32.

[8] G. Iosifidis, L. Gao, J. Huang, and L. Tassiulas, "Enabling crowd-sourced mobile internet access," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 451–459.

[9] C. Perkins, E. Belding-Royer, S. Das *et al.*, "Rfc 3561-ad hoc on-demand distance vector (aodv) routing," *Internet RFCs*, pp. 1–38, 2003.

[10] http://www.netfilter.org, "Netfilter."

[11] http://opengarden.com/firechat, "Firechat," 2014.

[12] P. Gardner-Stephen and S. Palaniswamy, "Serval mesh software-wifi multi model management," in *Proceedings of the 1st International Conference on Wireless Technologies for Humanitarian Relief*. ACM, 2011, pp. 71–77.

[13] A. Neyem, S. F. Ochoa, J. A. Pino, and L. A. Guerrero, "Sharing information resources in mobile ad-hoc networks," in *Groupware: Design, Implementation, and Use*. Springer, 2005, pp. 351–358.

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

[15] D. Bornstein, "Dalvik vm internals," in *Google I/O Developer Conference*, vol. 23, 2008, pp. 17–30.

[16] https://github.com/ThomasMadappattu/P2PSQ, "D2d service sharing."

[17] https://help.netflix.com/en/node/306, "Netflix minimum requirements."

[18] C. Bisdikian *et al.*, "An overview of the bluetooth wireless technology," *IEEE Commun Mag*, vol. 39, no. 12, pp. 86–94, 2001.