2017-06

Secure Infrastructure-less Network (SINET)

Akin, Micah P.

Monterey, California: Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

SECURE INFRASTRUCTURE-LESS NETWORK (SINET)

by

Micah P. Akin

June 2017

Thesis Advisor: Gurminder Singh
Co-Advisor: John H. Gibson

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2017	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE SECURE INFRASTRUCTURE-LESS NETWORK (SINET)	**5. FUNDING NUMBERS**
6. AUTHOR(S) Micah P. Akin	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Military leaders and first responders desire the familiarity of commercial-off-the-shelf lightweight mobile devices while operating in the environments of the modern battlefield and disaster sites. Both environments pose a significant challenge for command and control since they lack reliable or secure communication infrastructure. Routine and simple mobile information-sharing tasks become a challenge over the cumbersome and expensive radios currently available to first responders and the military. To fill this gap, there is a need for secure, well-connected, lightweight, and mobile handheld computing devices with simple and familiar interfaces.

This research explores the current Department of Defense requirements for security, existing secure tactical radios, and mobile device technologies. Furthermore, we investigate if Android devices might provide a solution. Specifically, we investigate the promising technology of Wi-Fi Direct on Android devices to build a secure network using a homogeneous Wifi mesh. We find that mobile devices running Android 6.0 API 23 cannot build a multi-hop homogeneous Wifi mesh without obtaining root permission. We recommend two methods for overcoming this limitation. The most promising method involves embedded devices providing a secure, lightweight, and mobile infrastructure through a homogenous Wifi mesh.

14. SUBJECT TERMS manet, mesh, infrastructure-less networks, denied or degraded network environment, secure	**15. NUMBER OF PAGES** 99
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

SECURE INFRASTRUCTURE-LESS NETWORK (SINET)

Micah P. Akin
Captain, United States Marine Corps
B.S., United States Naval Academy, 2007

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 2017

Approved by: Gurminder Singh, Ph.D.
Thesis Advisor

John H. Gibson
Co-Advisor

Peter J. Denning, Ph.D.
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Military leaders and first responders desire the familiarity of commercial-off-the-shelf lightweight mobile devices while operating in the environments of the modern battlefield and disaster sites. Both environments pose a significant challenge for command and control since they lack reliable or secure communication infrastructure. Routine and simple mobile information-sharing tasks become a challenge over the cumbersome and expensive radios currently available to first responders and the military. To fill this gap, there is a need for secure, well-connected, lightweight, and mobile handheld computing devices with simple and familiar interfaces.

This research explores the current Department of Defense requirements for security, existing secure tactical radios, and mobile device technologies. Furthermore, we investigate if Android devices might provide a solution. Specifically, we investigate the promising technology of Wi-Fi Direct on Android devices to build a secure network using a homogeneous Wifi mesh. We find that mobile devices running Android 6.0 API 23 cannot build a multi-hop homogeneous Wifi mesh without obtaining root permission. We recommend two methods for overcoming this limitation. The most promising method involves embedded devices providing a secure, lightweight, and mobile infrastructure through a homogenous Wifi mesh.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AES	Advanced Encryption Standard
AES-128-CMAC	Advanced Encryption Standard (with 128-bit key) Cipher-based Message Authentication Code
AN/PRC	Army Navy / Portable Radio Communications
ANW2	Adaptive Networking Wideband Waveform
AODV	Ad Hoc On-demand Distance Vector
API	Application Program Interface
C2	Command and Control
CBC	Cipher Block Chaining
CCI	Controlled Cryptographic Items
CCMP	Counter Mode Cipher Block Chaining Message Authentication Code Protocol, Counter Mode CBC-MAC Protocol
CNSA	Commercial National Security Algorithm
COMSEC	Communications Security
COTS	Commercial off the Shelf
CSfC	Commercial Solutions for Classified
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DHCP	Dynamic Host Configuration Protocol
DHT	Distributed Hash Table
DNS	Domain Name Service
DOD	Department of Defense
DSDV	Dynamic Destination-Sequenced Distance-Vector
DSR	Routing and Dynamic Source Routing Protocol
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EUD	End User Device
FIPS	Federal Information Processing Standards
GO	Group Owner
HAIPE	High Assurance Internet Protocol Encryptor

HMAC-SHA	Keyed-Hash Message Authentication Code - Secure Hash Algorithm
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPSEC	Internet Protocol Security
IW	Integrated Waveform
JTRS	Joint Tactical Radio System
LOS	Line of Sight
LTE	Long-Term Evolution
MANET	Mobile Ad-hoc Network
MCP	Mobility Capabilities Package
MPR	Multipoint Relays
NFC	Near Field Communications
NIAP	National Information Assurance Partnership
NIPRNET	Non-secure Internet Protocol Routing Network
NSA	National Security Agency
NSS	National Security Systems
OLSR	Optimized Link State Routing
P2P	Point-to-Point
POP	Post Office Protocol
RADIUS	Remote Authentication Dial-In User Service
RFC	Request for Comment
RREP	Route Response
RREQ	Route Requests
SATCOM	Satellite Communication
SHA	Secure Hash Algorithm
SINET	Secure Infrastructure-less Network
SIPRNET	Secret Internet Protocol Routing Network
SLAAC	Stateless Address Auto Configuration
SQL	Structured Query Language
SSH	Secure Shell
TC	Topology Control

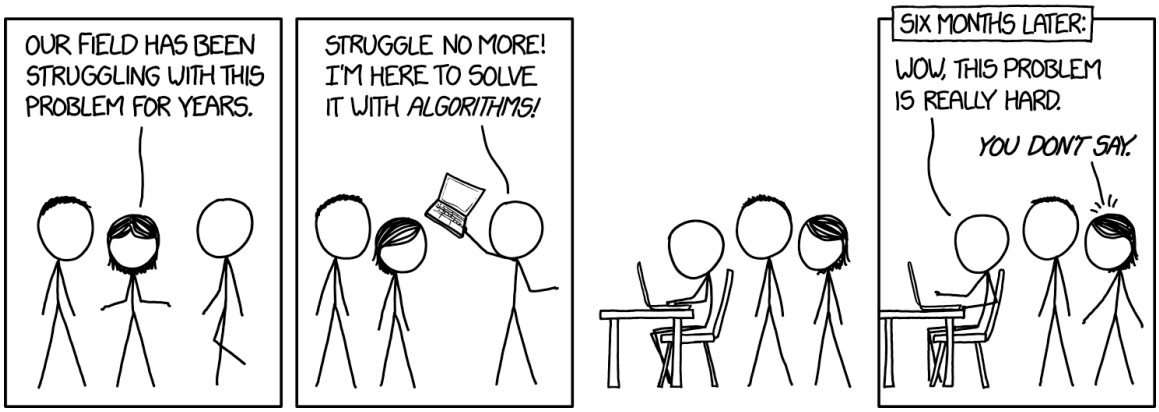
TDD	Time Division Duplexing
TDMA	Time Division Multiple Access
TRANSEC	Transmission Security
TTL	Time-to-live
UHF	Ultra High Frequency
UPNP	Universal Plug and Play
USSOCOM	United States Special Operations Command
VHF	Very High Frequency
VPN	Virtual Private Network

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my advisors, Dr. Gurminder Singh and John H. Gibson, for their support and patience through this entire process. Their guidance and experience enabled this research. I also would like to thank Robin Akin for all of her support.

The below XKCD webcomic describes the evolution of this research effort.



Source: https://imgs.xkcd.com/comics/here_to_help_2x.png.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Military and first responders desire to leverage the familiar lightweight commercial off the shelf (COTS) mobile devices while operating on the modern battlefield or disaster areas. These environments can sometimes lack reliable or secure communication infrastructure. In such situations, military and first responders must provide their communications infrastructure. Military radios are expensive and awkward in comparison to COTS mobile devices, and are limited in numbers because of high costs. This research explores how Android devices might meet the current Department of Defense (DOD) requirements for security and provide military and first responders a familiar and simple-to-use-device to communicate over which enables Command and Control (C2). In this chapter, we define the problem, state our objective, outline the scope of our research, explain the impact to the DOD, and describe the structure of our research.

A. PROBLEM DEFINITION

A substantial amount of study has been done to exploit the advanced capabilities of the latest mobile devices for a variety of purposes; the domain of mobile device secure communications in infrastructure-less environments lacks a comparable level of research. We define the infrastructure-less environment found on modern battlefields and disaster sites and describe the need for mobile devices in those environments. Further, we highlight related work that we plan to build upon for our research.

1. The Infrastructure-less Environment

Infrastructure-less environments arise from either an absence of communications infrastructure or the degradation or destruction of such infrastructure. For instance, in battlefield situations, an enemy may deny the use of the infrastructure resulting in a lack of trustable infrastructure. This harsh environment poses a significant challenge for C2. Military and first responders must provide their communications infrastructure as described in the next two examples.

The military executes long-range aerial insertion over 100 miles from friendly forces and must arrive at their destination with all needed resources. On many of these missions, it is prohibitive to bring heavy or bulky communications equipment to enable bandwidth-intensive communications. The cellular infrastructure either may not exist or cannot be trusted. Every member of the mission still must carry on with the task at hand and needs to effectively communicate with each other. They need to send messages to each other, share location information of friendly forces, mark enemy forces, mark points of interest, share images, and coordinate supporting forces. These everyday tasks prove challenging over expensive and restrictive tactical radios which primarily support voice communications.

A similar example exists when first responders operate in disaster sites where any surviving communications architecture may be congested, unreliable, or insecure. Coordinating the rescue and relief team's efforts would be significantly improved by robust communications capabilities. Each member of the response team should be able to send messages to others in the team, share location information, mark dangerous areas or points of interest, share images, and manage support. Similar to the military, these tasks prove challenging over handheld radios.

As we see in these two examples, simple and routine information sharing tasks become a challenge over handheld radio technology currently available to first responders and military personnel. Modern battlefields and disaster sites share the same underlying problem of a lack of trustable high bandwidth communications infrastructure. Austerity in communications networks surprise personnel who are accustomed to working with light weight and portable mobile handheld computing devices in well-connected environments.

2. Mobile Devices in Infrastructure-less Environments

Current secure mobile solutions require extensive manual configuration of multiple secure tunnels over tenuous links or must be tethered to an expensive tactical radio to provide the secure communications link. Neither is practical on the battlefield or disaster sites where only rugged devices survive the harsh environment. Complex to

configure, fragile, and expensive solutions are not practical. Any solution to this problem must provide a *durable or inexpensive to replace, well-connected, lightweight,* and *mobile handheld computing device* with a *simple and familiar interface*.

B. MOTIVATION AND RELEVANCE TO THE DEPARTMENT OF DEFENSE

Military leaders desire to leverage the familiar lightweight COTS mobile devices while operating on the modern battlefield in which computing and communications infrastructure are either lacking or impaired. Significant efforts have been made toward realizing that goal, but secure connectivity remains a weakness for these devices. Without secure connectivity, these mobile devices are severely underutilized. This research aspires to support secure connectivity through providing a less cumbersome and more fault-tolerant method of encrypting mobile device communications.

1. Data Needed in Austere Environments

C2 involves sharing information both up and down the chain of command. Commands include written directions, pictures of targets or areas of interest, diagrams to help explain the written directions, locations of key terrain. Control consists of a constant feedback between the commanding to the commanded. Control might involve voice communications, tables of information such as troop numbers and locations, a video feed, and damage assessments (containing pictures, videos, and text). A commander's staff performs the critical process of distilling all of the data and information into useful knowledge to inform the Commander's decisions. Rich media helps convey that information in easily digestible formats. Too much information turns into information overload and delays the Commander's decisions, too little information and the Commander makes uninformed decisions, making timely, accurate information critical.

2. Apps Needed in Austere Environments

There are mobile applications being developed to make devices more useful for military and disaster relief missions. These applications assume a simple, fast, secure, and reliable communications link to operate. In an austere environment, those

communications links must be established by the friendly forces, in a manner that sustains the required tempo of operations.

For this research, we distil the above into the following media types: pre-formatted text messages (small), free text (small), formatted text (small), pictures (large), mixed media documents (large), and video (huge). Our solution needs to handle all of these formats and allow collaboration apps to operate seamlessly.

C. SCOPE AND BOUNDARY

Our research focuses on how to meet the needs of the DOD through mobile devices that self-organize into a mesh network that securely forwards traffic utilizing Internet Protocol (IP) communications without relying on external wireless or cellular infrastructure. This research specifically investigates COTS Android mobile devices (phones and or tablets) and their ability to form self-healing routes over a Wifi mesh using NSA-approved encryption techniques.

The following, while closely related to this research, will not be covered:

- The distribution of digital keys and decision about which keys to trust—for this research we assume each device contains its private digital key and the public keys of all other trusted devices.
- How to protect data at rest on a mobile device that meets the NSA's encryption requirements.
- How to provide a way for this mesh network to easily join or leave an existing infrastructure from that infrastructure's perspective.

D. THESIS ORGANIZATION AND OVERVIEW

Chapter I presents the subject of the research and provides scope to the problem and the research. Chapter II, examines the encryption requirements for secure communication, then assess the current military radios that meet the security requirements, and finally we survey wireless technologies found in COTs devices and the mobile ad hoc network (MANET) concepts needed to interconnect them. Chapter III outlines our proposed design of an App that provides a secure MANET with embedded DNS. Chapter IV, we explain the results of our work and specifically the limitations of

Android 6.0 (Marshmallow) API 23 impose on implementing a homogeneous Wifi mesh network of mobile devices. We then suggest alternative methods for building a secure mesh to overcome the limitations. Chapter V, summarizes and concludes our findings.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

To leverage commercial-off-the-shelf (COTS) products to overcome the austere communications of an infrastructure-less environment, we need to understand what factors impact the solution space. This chapter explores current security requirements, government contracted solutions, promising current COTS technologies, methods to build mesh networks, and current research into building mesh networks with mobile devices.

A. SECURITY REQUIREMENTS

Military and first responders need their communications equipment to provide *confidentiality*, *integrity*, and *availability* (Hammond, Davis, & Gibson, 2015). The Department of Defense (DOD) requires all classified National Security Systems (NSS) conform to the National Security Agency (NSA) Type 1 Certification (Department of Defense, 2009). To simplify the process of integrating modern commercial security solutions, the NSA created the Commercial Solutions for Classified (CSfC) program (Hammond et al., 2015; National Security Agency, 2016c). To protect those commercial systems, the NSA mandates use of the Commercial National Security Algorithm Suite (CNSA) (National Security Agency, 2016b). In the next three subsections, we look into each of these requirement sets.

1. Type 1 Encryption

Type 1 certified encryption equipment utilizes approved NSA algorithms. The NSA endorses it for securing classified and sensitive NSS information (Department of Defense, 2009). Type 1 devices must be handled as classified or as Controlled Cryptographic Items (CCI) (National Security Agency, 2003). CCI, while unclassified, still requires strict physical control measures to protect against loss or compromise (National Security Agency, 2003).

2. Commercial National Security Algorithms (CNSA)

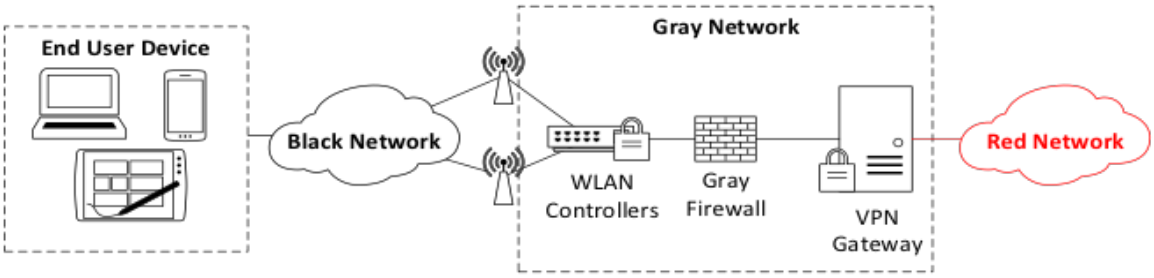
Per the NSA's Information Assurance website, Commercial National Security Algorithm Suite (CNSA) now replaces Suite B for mandating commercial cryptographic algorithms for protecting the NSS (National Security Agency, 2016b). The main reason for this change comes from the advancement in quantum computing that rapidly breaks all traditional cryptographic keys. The NSA built the program to transition to quantum resistant algorithms, which are in the process of development. Until the new quantum resistant algorithms are available, the program continues to rely on the Suite B algorithms: Advanced Encryption Standard (AES) for information protection. Elliptic Curve Diffie-Hellman (ECDH) for key establishment. Elliptic Curve Digital Signature Algorithm (ECDSA) for digitally signing. Secure Hash Algorithm (SHA) for creating an information fingerprint or hash (Hammond et al., 2015; National Security Agency, 2015). Table 1 provides an overview of the requirements to protect Secret and Top Secret information.

Table 1. CNSA Requirements for Classified Information. Adapted from National Security Agency (2015).

Classification Level	AES	ECDH	ECDSA	SHA
Secret	AES-128	256-bit ECDH	256-bit ECDSA	SHA-256
Top Secret	AES-256	384-bit ECDH	384-bit ECDSA	SHA-384

3. Commercial Solutions for Classified Program

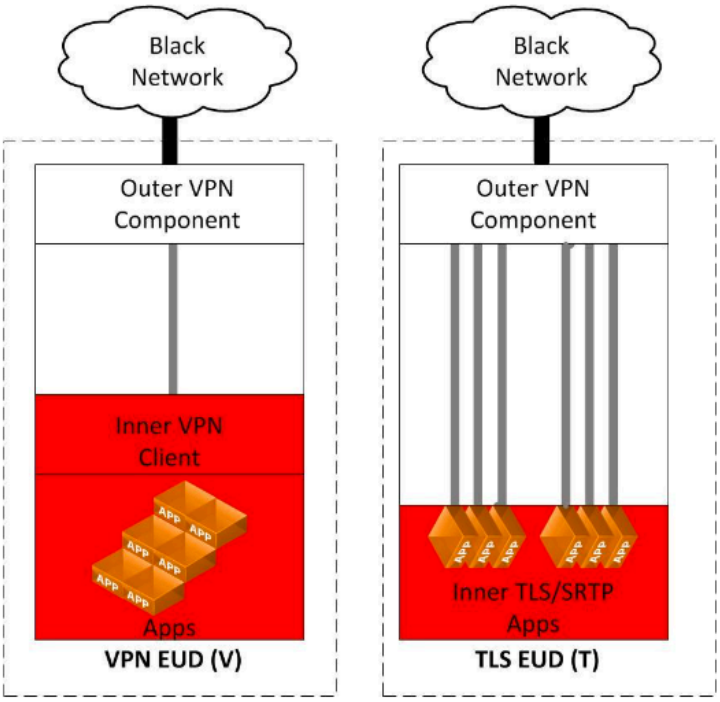
To simplify the labyrinth of regulations and meet mobile mission objectives, the NSA created the Commercial Solutions for Classified (CSfC) Program (National Security Agency, 2016c). The program offers capability packages each describing different ways to layer commercial solutions (chosen from a pre-approved Component List) to protect classified NSS information while also allowing remote sites, phones, laptops, and tablets to securely and wirelessly connect to a secured infrastructure (National Security Agency, 2016d). Figure 1 provides a simple example of how they might integrate.



Depicting Red, Gray, and Black Networks layered together to allow an End User Device to connect over a wireless connection securely.

Figure 1. Example of Layered Security Solution. Adapted from National Security Agency (2016a).

These packages describe Red, Gray, and Black Networks. The Red network refers to networks that contain the Enterprise Services and carry unencrypted classified or sensitive information. Red typically represents the unencrypted side of a sensitive data carrying system. The Grey network carries classified data encrypted in a single layer of commercial encryption (IPSEC, TLS, or SSH2). The Gray network also contains the Enterprise Mobility Infrastructure consisting of security mechanisms such as Intrusion Detection Systems (IDS) and Firewalls. The Black network carries the classified data twice-encrypted and consists of the wireless portion of the network. These networks are layered within each other (like an onion) with the Black network outside, the Gray network in the middle, and the Red network inside with the most protection (National Security Agency, 2016a). These colors do not indicate the classification levels; they merely indicate the state of encryption. Devices that access sensitive data must also provide at-rest encryption for stored data, anti-tamper mechanisms, strong authentication, and automatic-erasing mechanisms to stop brute-force attacks. Figure 2 provides two methods for handling security at the mobile device level. For the purpose of this study, we limit our focus to the data-in-transit aspect of the security requirements.



These two diagrams show how an EUD may implement the layered security to protect sensitive data. Both depict double wrapped sensitive data. The left diagram depicts two VPN tunnels bulk encrypting the applications' data. The right diagram depicts an outer VPN encrypting the data which the applications already once encrypted.

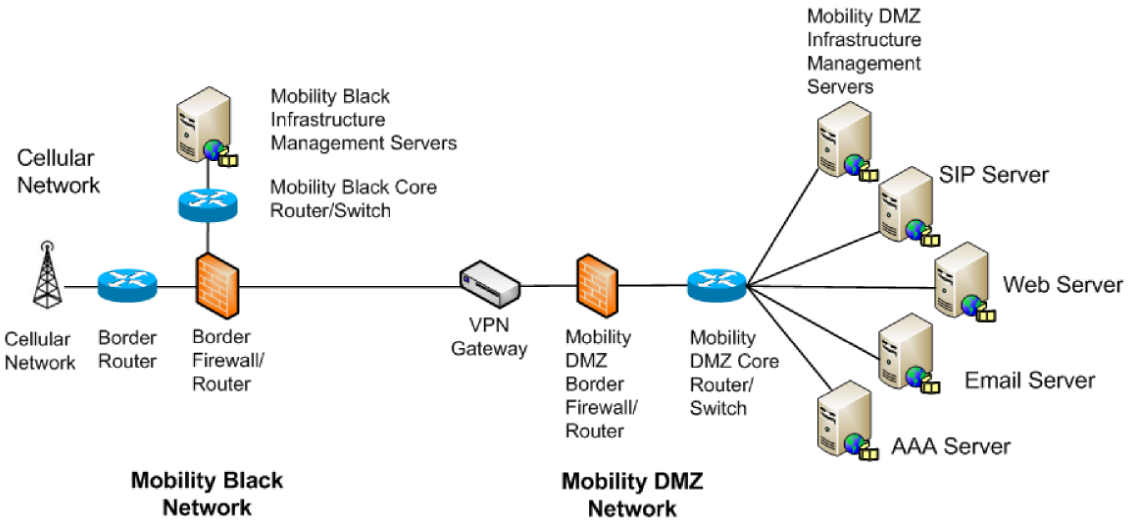
Figure 2. End User Device Solution Design. Source: National Security Agency (2013, p. 19).

Of the available Capability Packages, the Mobile Access Capability Package and the Campus WLAN Capability Package provide the closest architectures applicable to infrastructure-less environments. In the following two subsections, we look at those two capability packages.

a. Mobile Access Capability Package

The Mobility Capabilities Package allows mobile devices to connect to an insecure public cellular network through double-layered security solutions to existing secured enterprise services (National Security Agency, 2013). This Package grew from the desire for Blackberry phones to connect to the classified enterprise and reflects the limited capabilities found in older mobile phones. It primarily focuses on providing traditional Blackberry phone features such as secure voice calling, secure email, and

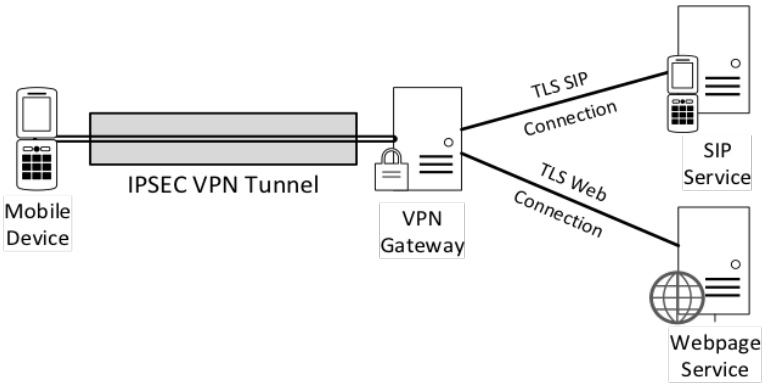
access to secure websites. It focuses on how to link a Blackberry type phone to an enterprise by providing limiting controls to what the device can access within the Secure Enterprise. Figure 3 shows how these services might be configured to provide phones with access to the Secure Enterprise. A key part of this Capability Package's security lies in the limitations it places on the services accessible to the mobile device. The purpose of this Package revolves around providing secure services to mobile devices.



An example architecture that would allow an End User Device to connect to Secure Enterprise Services such as a Web Server, Email Server, or Secure Voice over IP (SVOIP or SIP) Server.

Figure 3. Example Enterprise Mobility Infrastructure. Source: National Security Agency (2013).

To create the required two layers of encryption, this Capability Package requires each Enterprise Service to provide encryption for its data and a bulk VPN tunnel to encrypt all of the traffic as depicted in Figure 4. If the End User Device (EUD) accesses classified data, then the encryption scheme must utilize CNSA for encryption. Additionally, each layer must operate independently, and implement Public Key Infrastructure (PKI) authentication of each layer to reduce the potential for compromise of sensitive data (National Security Agency, 2013).



This diagram depicts a mobile device connecting to government enterprise networks over double encrypted tunnels. Each service (voice and email) encrypts its traffic before the VPN tunnel also encrypts it.

Figure 4. Two Layers of Encryption in the Enterprise Mobility Solution. Adapted from National Security Agency (2013).

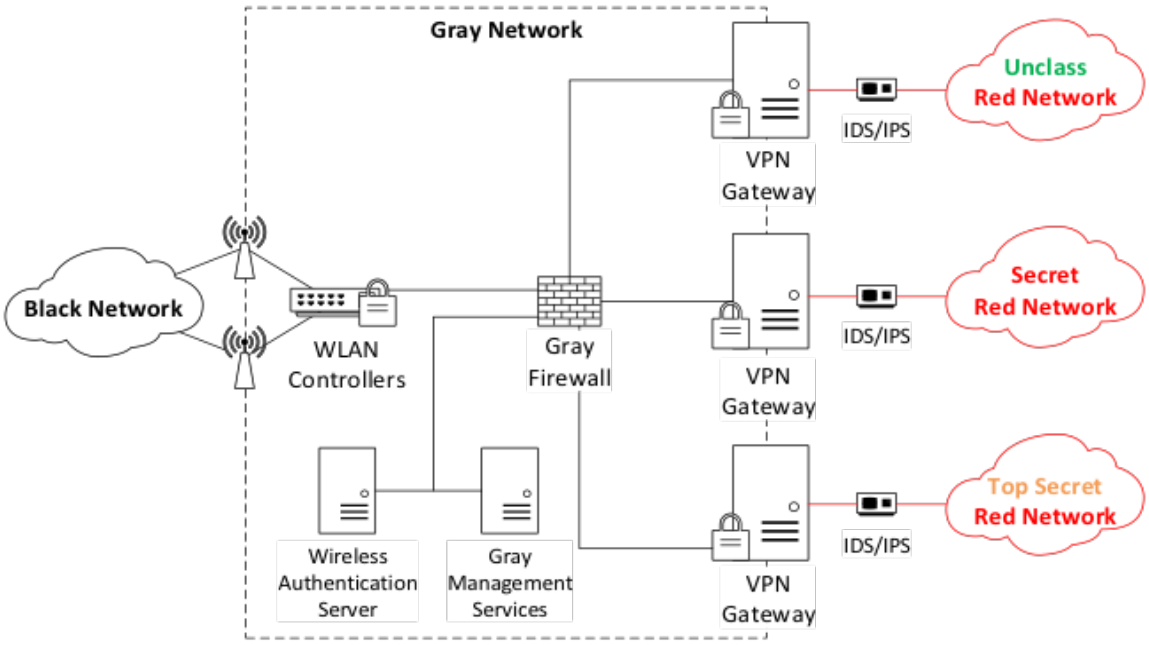
The Mobility Capabilities Package focuses on connecting traditional cellular phones to an enterprise network that contains sensitive data. The Capability Package provides security through limiting access to only specific services, hardening those services, and centralizing authentication, provisioning, and security mechanisms in the enterprise system.

b. Campus WLAN Capability Package

As its name implies, the Campus Wireless Local Area Network (WLAN) Package provides approved techniques for layering commercial security systems (from the approved list) to allow laptops, tablets, and smartphones to connect to an organization's network wirelessly. Instead of a Blackberry-type device accessing limited enterprise services, this Capability Package provides a way to replace conventional stationary workstations with their mobile versions.

This solution reaches the NSA's required double encryption through Wifi's WPA2 AES 128 encryption and enterprise Wifi management solutions as depicted in Figure 5. The inner layer of encryption comes from an IPSEC VPN using AES 256 encryption. While this package provides LAN capabilities to mobile computing devices,

it still requires the network to protect sensitive services with firewalls and access control lists.



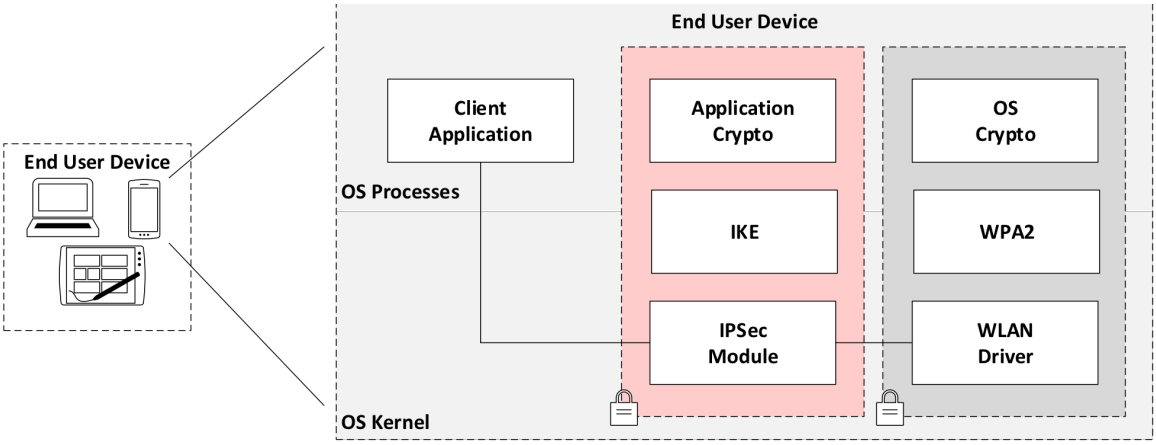
Mobile device connecting to government enterprise networks over double encryption provided by wireless security and IPSEC tunnel.

Figure 5. Campus WLAN Capability Package Example. Adapted from National Security Agency (2016a).

The Campus WLAN Package requires the following algorithms protect all NSS up to Top Secret: IPSEC Encryption: AES 256; authentication and signatures: RSA 3072 or ECDSA P-384; Key exchange: RSA 3072, DH 3072 or ECDH P-384; and hashing and integrity: SHA-384. Multiple Red networks of differing security levels may be part of this solution, but mobile devices should only connect to the enclave with corresponding classification level—Unclassified devices connecting to Unclassified networks, Secret devices with Secret networks, and Top Secret (TS) devices with TS networks.

To protect the End User Devices (EUD), this Capability Package requires physical controls for protecting the devices commensurate with their classification level. For example, a Secret device must either be encrypted when at-rest or it must be

physically controlled as a Secret device. Figure 6 depicts how an EUD might implement these security measures. When powered on, it must be treated at the level of the material it is processing and on the screen. When powered down, its physical controls must protect it at the level of its at-rest encryption capability.



How to set up two layers of encryption on the End User Device in the Campus WLAN Capability Package.

Figure 6. Campus WLAN End User Device Diagram. Adapted from National Security Agency (2016a).

This package focuses on connecting full-featured mobile devices to a network with two layers of security consisting of Wifi WPA2 and IPSEC VPN tunnels, as depicted in Figure 5. The security of this Package relies mainly on the enterprise protection capabilities. This Package provides a technology roadmap for the security requirements needed in creating a secure mobile COTS mesh.

c. Commercial Solutions for Classified Summary

The CSfC packages provide methods for securely connecting COTS devices to secure networks. The Mobility Capability Package focuses on how to connect the more traditional Blackberry-type mobile phone to secure enterprise services. The Campus WLAN Capability Package focuses on more full-featured mobile devices, such as laptops, tablets, and smartphones, allowing them to connect to classified networks wirelessly in a secure manner. To build an entirely COTs solution to overcome austere

communication environments, we need to leverage these CSfC packages. Specifically, the security requirements guide our choices in technologies with which to work.

4. Summary of Security Requirements

To effectively provide modern C2 systems in an infrastructure-less environment, we must understand the security requirements. NSA and the DIA together mandate Type 1 encryption accreditation, the CNSA algorithms, and CSfC packages as methods for protecting the NSS. The DOD currently requires Type 1 encryption devices to protect sensitive data. By de-facto the DOD also authorizes the use of CNSA and CSfC packages. Type 1 devices require rigorous testing which drives up their cost. COTS devices implementing CSNA and organized according to the CSfC package guidelines provide a significantly less expensive solution. Armed with a brief survey of Type 1, CNSA algorithms, and CSfC packages, we next examine a few of the communications solutions that match these accreditations and then a few technologies that might offer the promise of an alternate and less expensive solution.

B. EXISTING TYPE 1 SOLUTIONS

The majority of the current mobile infrastructure-less communications solutions center around military radios tethered to portable computing devices (a broader category than mobile devices). These radios excel in security, resilience, and reliability but falter in weight, bulk, and cost. Leveraging the Ultra High Frequency (UHF) and Very High Frequency (VHF) bands, they offer voice and low bandwidth data. To provide these links, they must either form direct radio links or indirectly connect through a repeater (Satellite Communications (SATCOM) repeaters, terrestrial repeater site, or airborne repeater). Voice communications require a wired handset or headset. Data communications require a laptop or mobile device to tether to the radio with an Ethernet cable, as wireless tethering is not currently approved (Figure 7.) The three radios below lead their class in this capability.

Connecting to SIPRNet via AN/PRC-117G. Source: U.S. Army, https://www.army.mil/article/68498/Army_networking_radios_improve_communications_at_tactical_edge.

Figure 7. Laptop Tethered to AN/PRC-117G

1. AN/PRC-117G

The Harris AN/PRC-117g provides secure voice and data communications over traditional UHF, VHF, and SATCOM channels. This radio contains the newest technology of the man-pack sized radios currently fielded (VHF, UHF, SATCOM channels using IW and ANW2 modes). One of its salient features is its ability to operate a secure Mobile Adhoc Network (MANET) using the proprietary Adaptive Networking Wideband Waveform (ANW2). Unfortunately, the AN/PRC-117G's limitations are significant: Heavy (12lbs), bulky (3.7 H x 7.4 W x 8.8 D inches), and expensive (over $30k per radio) (Harris Corporation, n.d.-a). Additionally, this radio must physically tether with a cable to whatever device requires its data capability. Another major limitation stems from ANW2's utilization of Time Division Multiple Access (TDMA) to share the frequency band with all other radios in the MANET. Each additional radio in the MANET reduces the usable bandwidth, especially when establishing multi-hop networks. While the weight, size, and required cables detract from its appeal, the cost impacts its availability within the military forces, and even more so for first responders.

2. AN/PRC-152A

The Harris AN/PRC-152A incorporates the advanced technologies found in the AN/PRC-117G into a much smaller form factor of 10.25 H x 3.0 W x 2.5 D inches with a reduced weight of 2.7lbs (Harris Corporation, n.d.-b). Its reduced size and weight result in a limited power output of 5 watts (half that of the AN/PRC-117G), which limits its transmission range. It also suffers from the limitations of ANW2. At $13k per radio, it still requires a significant capital outlay to adequately equip relevant force structures (Department of Defense, 2014).

3. RF-335M-STC

The Harris RF-335M-STC (Figure 8) extends the existing capabilities of the AN/PRC-152A. It adds the ability to operate on two separate channels simultaneously, a modular expansion pack, and a host of other powerful capabilities (such as the 16-Mbps MANET TrellisWare TSM-X protocol), all within in a smaller and lighter package (Harris Corporation, n.d.-c). Even with its impressive capabilities, this radio still requires a cable to connect a device to it. The United States Special Operations Command (USSOCOM) contracted Harris Corporation for these radios pending receipt of accreditation from NSA.

Figure 8. The Harris Falcon III RF-335M-STC. Source: Harris Corporation (n.d.-c).

4. Summary of Modern Type 1 Devices

The above devices provide secure and rugged capabilities to communicate in austere environments. Their cost prohibits issuing a device per first responder or military member. Additionally, the requirement to physically tether mobile devices to the radio adds undesirable weight and fragility to the solution. The next section explores the capabilities found in inexpensive COTS devices that might be leveraged to provide an alternate access architecture.

C. EXISTING WIRELESS TECHNOLOGY IN MOBILE DEVICES

This research seeks to explore whether or not the expensive military radios can be eliminated by directly connecting mobile devices together using their organic radio interfaces. For this to work, we must select approved mobile devices, with radio interfaces with enough range to enable dispersed devices to create links, and place these devices in a mesh network. Most mobile devices, on the CSfC Component List, support (in descending range order): 4G Long-Term Evolution (LTE), 802.11 Wifi, Bluetooth, Ant+, and NFC. The following subsections explore which transmitters most approved devices provide.

1. 4G LTE

While providing enough bandwidth (300 Mbps) and range (limited by line of sight) for connecting devices together in our desired mesh, 4G LTE requires cellular infrastructure to connect to other devices (Parkvall et al., 2008). 4G LTE operates with an optional AES 128-bit or 256-bit encryption mode, which the operating network can disable (Bartock, Cichonski, & Franklin, 2015). It operates on Time Division Duplexing (TDD) and receives its timing from the cellular towers (Parkvall et al., 2008). Since our task mandates an infrastructure-less environment, we do not further explore this technology. Future versions of LTE promise a technology called LTE-Direct. Unfortunately, LTE-Direct's specification currently requires cellular infrastructure to provide timing for device interaction (Qualcomm, n.d.).

2. IEEE 802.11 Wifi

IEEE developed the 802.11 Wifi standard to enable devices traditionally found on a local area network (LAN) to connect wirelessly to form the LAN. It utilizes Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) to operate in either the 2.4 GHz or 5 GHz ranges and provide up to 300 Mbps in version 802.11n (Cisco, 2015). Version 802.11AC bonds multiple channels together to allow up to 1.3-Gbps (Cisco, 2015). The useable range of all modes depends heavily on the device's power output, antenna, 802.11 protocol version, and environmental conditions. Higher frequencies (5GHz) provide higher bandwidth but exhibit less range and greater attenuation by walls (Cisco, 2015). Lower frequencies (2.4 GHz) provide larger range and better penetration power through walls and other solids but at the cost of lower bandwidth (Cisco, 2015). Consumer reviews on relevant websites report an average indoor range of 150ft and outdoor range of 300ft.

Wifi's security modes are Open, Wired Equivalent Privacy (WEP), Wifi Protected Access (WPA), and WPA2. The Open mode provides no security and is not considered secure beyond a minimal level. WEP utilizes the Rivest Cipher 4 (RC4) stream cipher (IEEE Computer Society, 2012; Lackner, 2013). WPA utilizes the Temporal Key Integrity Protocol (TKIP), which shores up some vulnerabilities in WEP but still utilized the RC4 stream cipher (Lackner, 2013). WPA2 mandates the Counter Mode Cipher Block Chaining Message Authentication Code Protocol (CCMP) utilizing 128-bit AES standards (IEEE Computer Society, 2012; Lackner, 2013) and provides more robust security than other modes.

Android devices provide two Wifi interfaces: wlan0 and p2p0. The wlan0 interface is used for standard Wifi operations and allows Access Control (Hotspot) and Infrastructure Mobile modes. The p2p0 interface is used for Wi-Fi Direct, detailed in the next subsection, and only operates when wlan0 is in Infrastructure Mobile mode. Figure 9 provides an overview of how Wifi and Wi-Fi Direct work together.

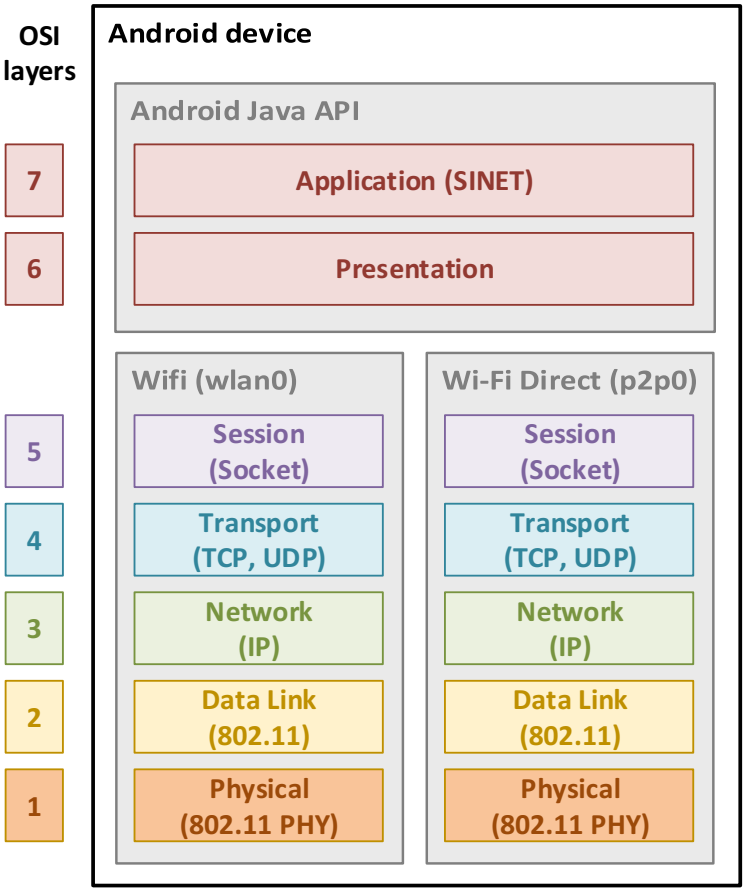


Figure 9. Android Devices Implementing Wifi and Wi-Fi Direct

Wifi's current edition (802.11AC) defines one of four possible operating modes for nodes: Access Control, Infrastructure Mobile, Ad Hoc Mobile, and Mesh modes (IEEE Computer Society, 2012).

a. Access Control

Access Control mode, utilizing an access point (AP), allows other devices to connect to each other but only through the AP. Depending on the setup, the AP may provide network services such as Dynamic Host Configuration Protocol (DHCP), routing, and Internet gateway. This capability is typical of home wireless routers or commercial access point systems.

b. Infrastructure Mobile

The Infrastructure Mobile mode allows devices to connect directly to a device that provides a Wifi AP through Access Control mode. This mode allows consumer devices to access Wifi networks and the Internet through wireless routers or commercial access point systems.

c. Ad Hoc Mobile

The Ad Hoc Mobile mode (Peer-to-Peer) enables devices to connect with each other without requiring any infrastructure directly. While the most basic mode available (IEEE Computer Society, 2012), it is rarely used due to poor standardization between devices implementing it (Griffith, 2009). Some early distributions of Android supported Ad Hoc mode with their devices.

d. Mesh

The Mesh mode creates a layer-two mesh where all joined devices operate in the same Internet Protocol (IP) subnet. Mesh mode provides both secure and nonsecure versions. Defined in 802.11s, Mesh mode has recently grown in popularity with Google Home Wifi (Google, n.d.), Plume Wifi (Plume Wifi, n.d.), and AmpliFi (AmpliFi Wi-Fi, n.d.) products.

e. Wifi Summary

Most Android and Apple IOS mobile devices support 802.11 Wifi in either Infrastructure Mobile (client device) or Access Control ("mobile hotspot") modes. Since the advent of Wi-Fi Direct, described in the next section, those few Android devices dropped Ad Hoc for Wi-Fi Direct. Currently, only rooted Android devices with custom Wifi kernel modules support Wifi Ad Hoc or mesh modes (Wakelin, 2014).

3. Wi-Fi Direct

The Wi-Fi Alliance introduced Wi-Fi Direct in 2009 as an alternative to the poorly standardized implementations of 802.11 Ad Hoc (peer to peer) mode (Griffith, 2009). Wi-Fi Direct builds on the advanced technologies of 802.11AC to provide a

simplified process of discovering available peers or services and securely and automatically connecting to them (Monarrez, Singh, & Buettner, 2015a; Monarrez, Singth, & Buettner, 2015b; Wi-Fi Alliance, 2014). Wi-Fi Direct was designed specifically to simplify Ad-Hoc peripheral connections, not multi-hop architectures, as depicted in Figure 10. It utilizes Wifi Access Control and Infrastructure Modes to build networks of devices.

Figure 10. Adding a Device to P2P Group by Invitation. Source: Wi-Fi Alliance (n.d.).

a. Operation Modes

Wi-Fi Direct operates in three modes: Standard, Autonomous, and Persistent. These modes govern how Peer-to-Peer (P2P) groups are formed (Camps-Mur, Garcia-Saavedra, & Serrano, 2013). Only Wi-Fi Direct devices may act as Group Owners (GO), but either legacy 802.11 devices or Wi-Fi Direct enabled devices may join an existing group. The GO sets up an 802.11 Access Control mode AP for legacy devices to join.

b. Security

Wi-Fi Direct uses Wifi Protected Setup (WPS), seen mainly as push-button setup on modern routers (Wi-Fi Alliance, n.d.). Even though the push-button method of WPS shows significant security flaws (The H. Open, 2011), the Wi-Fi Direct software version omits the portions causing the security flaws. Since WPS utilizes WPA2-based

encryption for Transport Layer Security (TLS), the resulting network is as secure as other WiFi WPA2 networks.

c. Addressing

When Android version 6.0 acts as a GO, it utilizes the same IP subnet of 192.168.49.0/24 for each network it establishes. The GO assumes the address 192.168.49.1 and issues IPs via DHCP to the clients that join. Since all Wi-Fi Direct networks contain the same address space, *a device can only associate to one Android-hosted Wi-Fi Direct network at a time* (Funai, Tapparello, & Heinzelman., 2017).

d. Routing

On the Android versions of the implementation, Wi-Fi Direct does not add any routing statements to the routing table. Traffic originating from within the Wi-Fi Direct group must remain within the group. This design protects mobile devices from other Wi-Fi Direct connected devices exploiting their cellular data plans or connections to other networks but limits its ability to support multi-hop network architectures.

e. Service Discovery

Wi-Fi Direct utilizes service discovery messages built on the Universal Plug and Play (UPnP) protocol to broadcast beacon messages containing announcements about the services provided by its Wi-Fi Direct network. Wi-Fi Direct's Service Discovery might allow a printer, television, or coffee maker to advertise its network and let nearby devices know the services it provides (Camps-Mur et al., 2013). This capability shows an interesting departure from IEEE's Wifi, which traditionally views itself purely as a means to connect devices to the Internet.

f. Wi-Fi Direct Advantages

Wi-Fi Direct provides a boon for home users and provides a couple of useful tools for our efforts in building mesh networks. Leveraging the power management features of Wifi Infrastructure and Access Control modes, it outperforms Wifi Ad Hoc networks in

power conservation (Wirtz, Heer, Backhaus, & Wehrle, 2011). Service Discovery also provides a useful method to identify other members of our mesh.

4. Bluetooth

Ericsson, a mobile phone manufacturer, originally designed Bluetooth to allow devices to connect to each other in a Personal Area Network (PAN) (Beachy, Gibson, & Singh, 2015). IEEE standardized Bluetooth as 802.15. Today, Bluetooth Special Interest Group (SIG) maintains the Bluetooth standard (Bluetooth Special Interest Group, 2014; Decuir, 2011). Bluetooth utilizes the 2.4 GHz band and creates channels using FHSS. Each channel supports seven slave devices controlled by a master device (eight devices per channel). The channel's master device utilizes TDD and polling to synchronize communication with slave devices, as depicted in Figure 11. Each slave may send data directly to the master (when polled) but not directly to each other, whereas the master may send data to all of the slaves at once (Beachy et al., 2015; Bluetooth Special Interest Group, 2014).

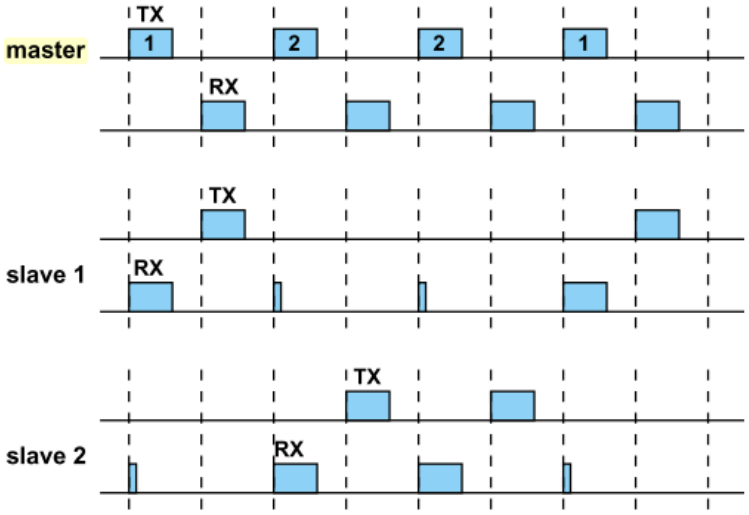


Diagram depicts how a master controls the timing of a channel and how two slaves listen on that channel.

Figure 11. Bluetooth Channel Time Division Duplexing Diagram. Source: Bluetooth Special Interest Group (2014).

Bluetooth version 2.0 introduced an Enhanced Data Rate (EDR), boasting a 2.1 Mbps transfer rate. It also introduced FIPS-approved algorithms for encryption, including SHA-256, HMAC-SHA-256, and P-192 elliptic curve (Bluetooth Special Interest Group, 2014). Bluetooth version 3.0 introduced high-speed (Bluetooth+HS) capabilities which allowed devices to negotiate with each other across Bluetooth and connect to each other using 802.11 for data transfer, providing a theoretical transfer speed of 24 Mbps. Bluetooth version 4.0 introduced the Low Energy (LE) specification for Internet of Things (IoT) devices operating on "coin" batteries (Bluetooth Special Interest Group, 2014). Field testing of mobile devices using Bluetooth suggests an average throughput of almost 700 Kbps at a range up to 180 feet (Hammond et al., 2015). Android devices currently support Bluetooth 4.0 (Samsung Corporation, n.d.).

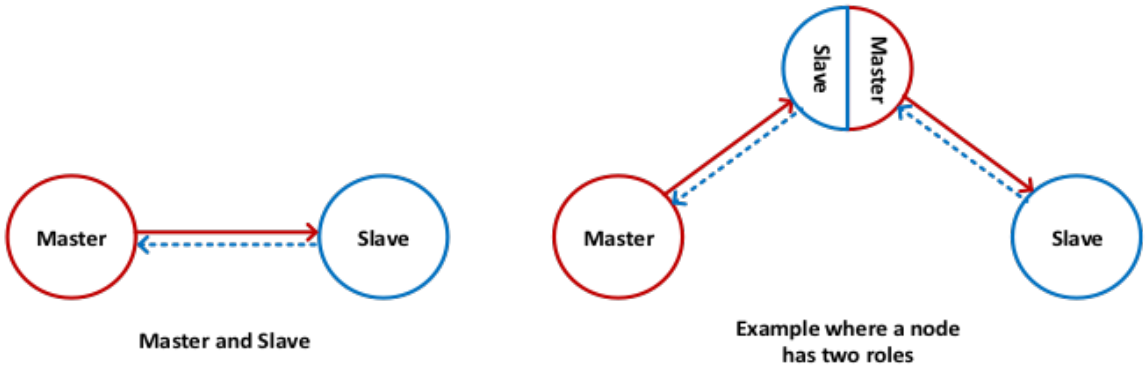
Bluetooth's limited throughput and range limit its usefulness in building media-rich mesh networks. As mentioned by Hammond, Bluetooth may provide an excellent control channel or means to transfer text, but not an optimal means to transfer video or pictures (Hammond et al., 2015).

5. ANT+

Dynastream, a subsidiary of Garmin Ltd, created the proprietary, but open, ANT+ protocol to allow personal fitness sensors (master nodes) to transfer information to displays (slave nodes) with ultra-low power usage. An ANT+ slave node can capture multiple master nodes' data. Multiple slave nodes can capture the same master node's data (Dynastream, n.d.-b). The master nodes and slave nodes can be configured in any mix of many-to-many architectures, as seen in Figure 12 (Khssibi, Idoudi, Van Den Bossche, Val, & Azzouz Saidane, 2013).

Currently only supported by Android mobile devices, ANT+ operates up to 90 feet on one of eight channels in the 2.4GHz frequency range, with one of three types of 8-byte messages: Broadcast, Acknowledgement, and Burst. Broadcast messages provide a stream of unacknowledged data. Acknowledgment messages provide critical data transmission that requires acknowledgment but uses more energy. Burst messages push 20 kbps from master to slave, require acknowledgment, and consume even more energy

(Khssibi et al., 2013). ANT+ provides "8-byte network key and 128-bit AES encryption" (Dynastream, n.d.-a).



(a) ANT+ Master node sending acknowledged messages to a Slave. (b) An ANT+ Master node pushing acknowledged messages to a Master/Slave node which also pushes acknowledged messages to another Slave node.

Figure 12. ANT+ Example Topologies. Adapted from Khssibi et al. (2013, Figure 2).

As a protocol designed for IoT devices, ANT+ provides excellent power conservation modes when in Broadcast mode but it fails to meet our needs in range, throughput, and device interoperability. So, for the purpose of our research, ANT+ is interesting but not usable in our mesh.

6. Summary of Existing Technologies

As we see, modern COTS mobile devices come equipped with a diverse range of communication technologies. In Table 2, we compare these technologies. To build an effective mesh, we need a technology that provides enough range and throughput to handle rich media. 4G LTE offers the best throughput and range except it requires cellular towers to control the timing of the protocol. ANT+ and Bluetooth offer excellent low power modes but do not provide the range or throughput needed. Thus, we focus our efforts on Wifi and Wi-Fi Direct. In the next section, we discuss how we might leverage these technologies to build different mesh networks.

Table 2. Comparison of Existing Technologies in COTS Mobile Devices

Technology	Outdoor Range	Through-put	Protocol	Intended Use	Security
4G LTE	20 miles	300 Mbps	OFDMA	Cellular voice and data	Optional 128-bit or 256-bit AES
Wifi	300 feet	300 Mbps	FDMA Channels with CSMA/CA in channel	Wireless LAN	128-bit AES-based CCMP
Bluetooth	180 feet	700 Kbps	FHSS channels with Master-Slave TDMA in channel	Wireless PAN	SHA-256, HMAC-SHA-256, and P-192 elliptic curve
ANT+	90 feet	20 Kbps	Master-Slave: FDMA channels with TDMA in channel	PAN IoT sensors	8-byte network key and 128-bit AES

Adapted from Bluetooth Special Interest Group (2014), Decuir (2011), Dynastream (n.d.-a, n.d.-b), IEEE Computer Society (2012), Khssibi et al. (2013), Parkvall (2008), Qualcomm (n.d.).

D. TYPES OF MESH NETWORKS

Meshes come in many different shapes and sizes. In this section, we present the difference between fully and partially connected mesh networks. We also present the difference between heterogeneous and homogeneous meshes and review the teeter-totter technique and how that might apply to our problem.

1. Fully vs. Partially Connected Mesh Networks

Meshes are classified into partially or fully connected meshes as depicted in Figure 13. In the context of networking, each has different strengths and weaknesses that we will now examine.

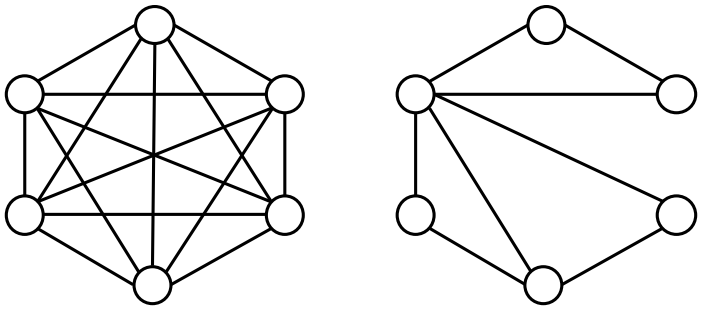


Figure 13. Fully vs. Partially Connected Meshes

A fully connected mesh implies that every node has a link to every other node. In a fully connected computer network, each computer can directly communicate to every other computer; the process of one node communicating with another node is simple. In a fully meshed Ethernet network, each computer would connect to all other computers with a dedicated Ethernet cable—requiring N*(N-1)/2 cables for N computers. In practice, fully connected meshes are uncommon since they do not scale well. As fully connected computer meshes grow, they quickly exhaust their access medium. In a resource-constrained environment, fully connected meshes are rare and impractical.

A partially connected mesh contains at least some nodes not fully connected to every other node. In a partially connected mesh computer network, devices not directly connected send traffic to intermediary devices, which in turn forward the traffic to the desired recipient, thus forming a multi-hop architecture. Most modern computer networks are a form of partially connected meshes, where multiple paths between nodes allow for some redundancy and survivability. Partially connected meshes trade the complexity of their routing algorithms for the conservation of the transport medium and scalability.

The resource-constrained environment of the radio frequency (RF) spectrum generally, as well as the range constraints of line-of-sight radios, restricts wireless data networks to only partially connected meshes. For this reason, we only attempt to implement a partially connected mesh for our research.

2. Heterogeneous versus Homogeneous Mesh Networks

Not only do we see the differences between fully connected meshes and partially connected meshes, but we also must consider utilizing different mediums in creating the mesh. Thus, we need to focus on homogeneous and heterogeneous meshes.

A homogeneous mesh consists of similar nodes and similar links. In a mesh computer network, this might mean that the computers are all laptops and the links are all Bluetooth. The benefit of a homogenous mesh comes from the simplicity found in the similarities. The devices do not need to maintain multiple interface types. On the other hand, it has the disadvantage of the entire mesh suffering if that medium suffers.

On a heterogeneous mesh, the links might be similar while the nodes may be dissimilar or the nodes are similar while the links may be dissimilar, or both may be the case. A good example of this comes from the domain of the Internet of Things (IoT) with sensors and gateways (dissimilar devices) connecting into a heterogeneous mesh. A heterogeneous mesh also might consist of dissimilar links. For example, a computer mesh network might use Wifi and Bluetooth (Hammond et al., 2015). A benefit of a heterogeneous mesh comes from its diversity. A weakness in one part does not necessarily affect all. This diversity also produces the disadvantage of higher complexity in managing and requiring nodes to support multiple connection types (Hammond et al., 2015).

For our research, we strive for the simplicity of homogeneous mesh networks.

3. Teeter-Totter Technique

Another novel approach to connecting devices is the teeter-totter technique. This method is used when a device or node can only connect to one other device at a time. To form a mesh, a device switches back and forth between connections with other devices to create the appearance of being connected to more than one device at a time. By analogy, this is like multithreading in a single processor system: it gives the illusion of multiple threads executing simultaneously. This method, while interesting, should be reserved for extreme situations since it proves very inefficient with large delays in the passage of information (Funai et al., 2017).

4. Mesh Summary

As we see in this section, fully connected meshes are impractical since they require links to all devices, which creates a significant overhead on the access medium. We also see that heterogeneous meshes are complex. For our problem, we attempt to build a homogeneous and partially connected mesh network.

E. MANET ROUTING PROTOCOLS

The two broad categories of MANET routing protocols are proactive and reactive. Proactive protocols provide quick answers to routing requests by actively maintaining an awareness of the entire network topology. Proactive protocols collect and maintain this information to continuously map the network which results in significant network traffic. Reactive protocols do not attempt to find routes in advance. Instead, they search on-demand. Searching on-demand reduces the maintenance overhead but increases the request response time. The following sections explore proactive and reactive routing protocols.

1. Proactive Routing

The proactive routing protocols actively attempt to populate their routing tables. This search could be accomplished by examining all nodes (using depth- or breadth-first search techniques) or by finding local nodes and exchanging neighbor information with them. A few of the most well-known proactive, table-based, MANET routing protocols are Dynamic Destination-Sequenced Distance-Vector (DSDV), Babel, and Optimized Link State Routing (OLSR) protocols.

a. DSDV

The DSDV routing protocol utilizes the Bellman-Ford algorithm to find the shortest path by measuring the hop count to a destination. Further, it keeps an updated table of all destinations in the network. Each node uses “even” sequence numbers to declare good routes and avoid loops. When a node receives a route from a neighbor, it only updates when the route’s advertised sequence number is higher than its version of that route. If the advertised sequence number is also odd, then it drops the route. When a

node detects a link loss, it increments that table entry by one, which ensures its neighbors drop the route. Signaling neighbors to drop the route avoids the count-to-infinite issue that plagues the Bellman-Ford algorithm (C. E. Perkins et al., 1994). DSDV implements a settling time requirement to dampen radical routing fluctuations.

b. Babel

The Babel routing protocol also utilizes the Bellman-Ford algorithm to build a distance-vector routing protocol. Per Internet Engineering Task Force (IETF) Request for Comment (RFC) 6126 (Experimental), Babel borrows the slow convergence of DSDV to limit the duration and frequency of routing loops and black holes in unstable networks. Its limitations come from its periodic routing table updates which "generate more traffic than protocols that only send updates when the network topology changes" and its hold times for retraced network prefixes (Chroboczek, 2011). The Babel routing protocol shows promise for mesh networks that exhibit significant mobility but still require proactively established routing tables.

c. OLSR

Unlike the DSDV and Babel, the Optimized Link State Routing (OLSR) Protocol does not use the Bellman-Ford algorithm. It regularly floods HELLO messages to discover links and floods topology control (TC) messages to broadcast link-state information throughout the network (Narra, Cheng, Çetinkaya, Rohrer, & Sterbenz, 2011). A core concept of OLSR comes from the use of multipoint relays (MPRs). "MPRs are selected nodes which forward broadcast messages during the flooding process" (Clausen & Jacquet, 2003). Link state information is generated only by MPR nodes, thus minimizing the number of flooded TC messages in the network (Clausen & Jacquet, 2003). OLSR works best in a densely-populated mesh network that can take advantage of the MPRs to reduce the traffic load.

2. Reactive Routing

Reactive routing protocols do not waste the resources of maintaining a route lookup table. Instead, they discover the route on-demand. By discovering routes on-

demand, the reactive routing protocols save on overhead but pass on the cost to the user who must wait for the route to resolve before traffic can flow. A few well know reactive routing protocols include Ad hoc On-Demand Distance Vector (AODV) Routing and Dynamic Source Routing Protocol (DSR).

a. AODV

The AODV protocol calculates the Distance Vector on-demand. It uses HELLO packets to discover and track neighbors. Upon an unknown route request, AODV floods route requests (RREQ) to its neighbors containing a time-to-live (TTL). Each neighbor rebroadcasts the RREQ. The RREQ continues outward until the TTL expires or a node responds with a route reply (RREP) message. The RREQ messages contain sequence numbers to protect intermediate nodes from creating routing loops and also determine the freshest path (highest sequence number). The node with knowledge of the destination sends a route reply (RREP) message back to the originator along the reverse path and forwards the RREQ to the destination so that it also knows the route. Every node along the reverse path records the new route for future use. If the RREQ does not find its destination, then the originator rebroadcasts it with a larger TTL and higher sequence number (C. Perkins, Belding-Royer, & Das, 2003). Note that the use of the reverse path implies that the protocol assumes bi-directional links throughout the entire network.

b. DSR

The Dynamic Source Routing (DSR) also uses a broadcast RREQ containing the source, destination, and sequence number to neighbors who forward the RREQ after appending themselves to the request. When the RREQ message reaches the destination, it contains the reverse path on which to send the RREP, also implying the demand for bi-directional links throughout the network. Maintaining a list of the nodes in the path protects against routing loops. The destination, intermediate, and source nodes cache the route and utilize it. When a link fails, each end of the failed link floods the failure to the network. Any node with a cached route that contains the link drops the route from its table (Johnson, Hu, & Maltz, 2007). DSR's strengths lie in its ability to handle node mobility and network segmentation.

3. MANET Routing Protocol Summary

In this section, we explored the common routing protocols utilized for MANETS. The two groupings of MANET routing protocols are proactive and reactive protocols. Proactive protocols provide a quick response at the expense of route maintenance traffic. Reactive protocols reduce the overhead of route maintenance but increase the time per request cost by adding the time for route discovery. Each protocol provides specific characteristics that make them strong in certain circumstances. For our highly mobile mesh, we utilize OLSR since it convergences quickly in response to routing requests.

F. MANET DATA LOOKUP

The infrastructure-less environment of MANETs removes the servers needed for centralized data lookups such as Domain Name Service (DNS), Structured Query Language (SQL), or Post Office Protocol (POP). These functions might reside in miniature form on mobile devices, but not in their centralized simplicity or power. By distributing these capabilities, it is possible to provide a fault-tolerant and aggregated implementation. Once distributed, each device must still have access to the disbursed capability. Finding data in this distributed system shares the same trade-off as we have seen in the proceeding section—proactive lookup or reactive search for content's location. In data lookup, a proactive system creates a Structured Overlay and a reactive search is called an Unstructured Overlay.

1. Structured Overlays

Structured Overlays build on the idea of distributing a data structure across multiple nodes. Hash Tables are a popular data structure for distribution since they provide key->value lookups. Hash tables allow efficient data indexing for retrieval in O(log(n)) computational complexity. The challenge of this scheme comes from how to distribute the entries symmetrically, how to synchronize the overlay with the physical topology of the MANET, and how to provide resilience against node loss or network segmentations. To this end, the MADPastry protocol integrates a Distributed Hash Table (DHT) with AODV (Zahn & Schiller, 2005). Tightly grouped nodes (by hop-count) hold similar items. Each area of the MANET has a Landmark node which helps coordinate the

other local nodes and provides a touch point for nodes looking into that section of the DHT. A DHT provides a reliable method for overcoming the lack of centralized servers in an infrastructure-less network such as a MANET.

2. Unstructured Overlays

Setting up an Unstructured Overlay requires a shared Application Program Interface (API) to ensure all nodes share a common communication language to enable flood requests to continue to other neighboring nodes. Requests flood until an answering node responds with a result. Then, the answering node either sends the resulting data or a pointer to the data (address) back to the requestor. This system excels in small networks at finding common data (like hay in a haystack). Unstructured Overlays do not guarantee that something is found and do not perform well in large-scale ad-hoc networks where the volume of flooding requests can overwhelm the system's capacity (Peterson & Davie, 2011).

3. MANET Data Lookup Summary

Both structured and unstructured data lookup have their place in distributed data systems. Structured overlays provide quick lookups of unique data (the needle in the haystack). Unstructured overlays provide low overhead and a simple way to find common or highly redundant data (the hay in the haystack). A distributed DNS server requires a structured overlay to ensure unique names are quickly available. We use a simple version of a DHT to provide DNS in our MANET implementation.

G. RELATED WORK

The field of using mobile devices to build a MANET has been subject of considerable and recent research. We intend to extend this body of work. This subsection provides recent and relevant research that we leverage.

Hammond (2015) offer an overview of the military's requirements for securing mobile devices. They survey the relevant regulations from the DOD, Defense Information Systems Agency (DISA), National Security Agency (NSA), and its subordinate the National Information Assurance Partnership (NIAP).

Beachy (2015) implement an Android Bluetooth messaging mobile application which utilizes a homogeneous mesh of Bluetooth connections to pass messages between Android devices.

Monarrez, Singh, and Buettner (2015b) explore how to extend the Wi-Fi Direct protocol to allow for a "persistent communications network that involves zero user interaction." Their extension provides an automated method for re-assigning the Wi-Fi Direct Group Owner (GO) without disrupting an existing Wi-Fi Direct group.

Camps-Mur, Garcia-Saavedra, and Serrano (2013) evaluate Wi-Fi Direct on Android version 4.0 in a variety of configurations as it forms groups and provides power saving. We focus specifically on their work on building multi-group connections.

Funai, Tapparello, and Heinzelman (2017) provide a method for building mesh networks on Android devices through obtaining root permissions and modifying how Android devices connects to other devices over Wi-Fi Direct.

H. SUMMARY

In this chapter, we reviewed the encryption requirements for secure communication, existing military solutions for infrastructure-less environments, wireless technologies found in COTs devices, types of meshes, MANET routing protocols, methods for storing data in a MANET we might employ, and existing work on the topic. The security requirements imposed by the military on their communication links establish a high bar. The solutions that meet that requirement are expensive and inconvenient. The device price must support large volume purchases for fielding to each first responder or military member. With this impetus, we investigate the built-in technologies of COTs devices for a means to connect them together in an ad hoc network with the potential of becoming a MANET. We also examined the different kinds of meshes that might be employed to build this MANET. Wifi with WPA2 encryption provides the needed security, bandwidth, and range to support this MANET. Out-of-the-box Android does not support mesh networks and Wi-Fi Direct's restrictive addressing scheme limits devices to connecting to only one Android-hosted Wi-Fi Direct network at a time. So, in the next chapter, we explore how we might overcome Android's limitations to build a Wifi mesh.

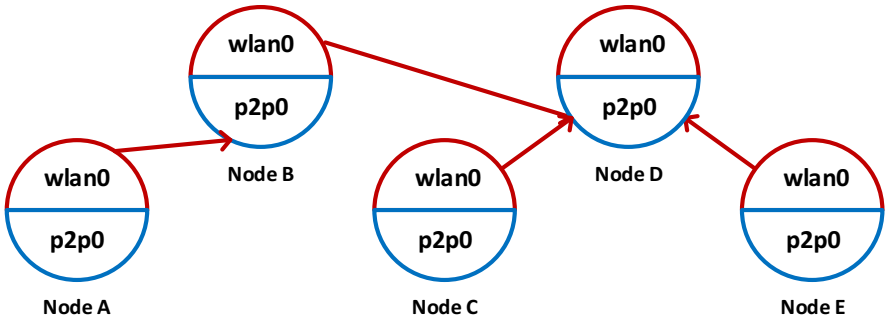
THIS PAGE INTENTIONALLY LEFT BLANK

III. DESIGN

In Chapter II we review the information requirements, security requirements, existing secure solutions, technologies modern mobile devices contain, different kinds of mesh networks, MANET routing protocols, and methods of storing information in MANETs without centralized servers. In this chapter, we explore how to build a homogeneous mesh network utilizing the technologies in an Android device.

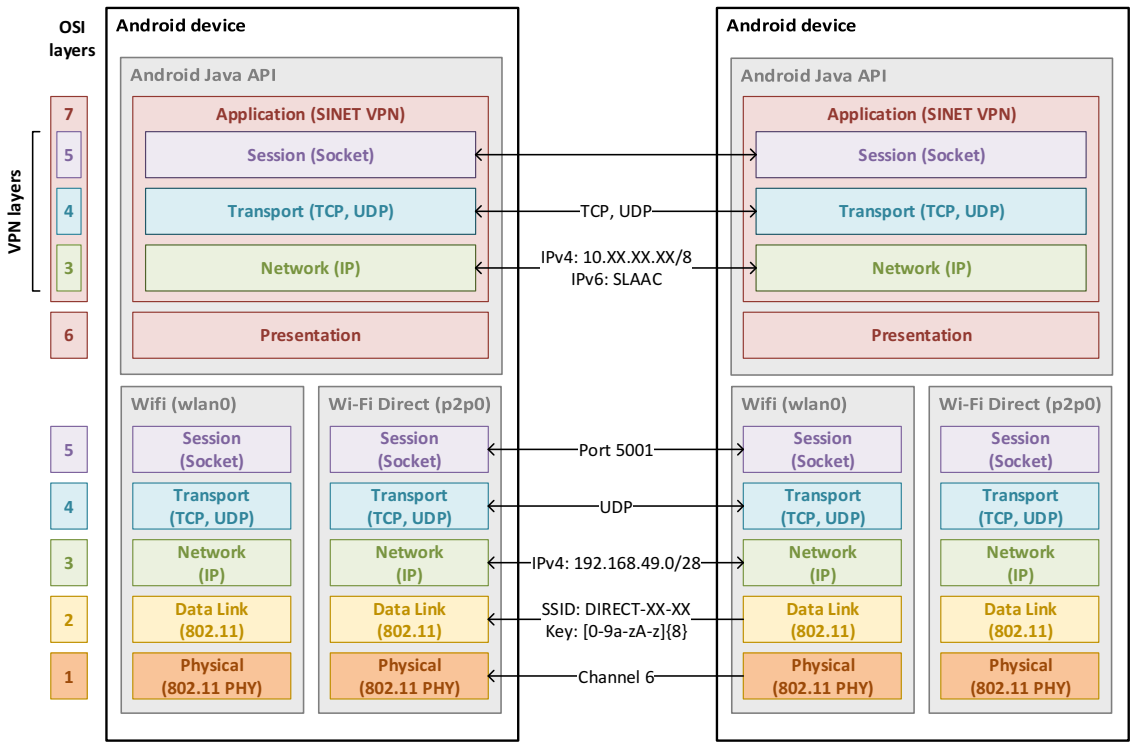
A. DESIGN OVERVIEW

The Wifi's well-established protocol, high bandwidth, and ubiquity within mobile devices make it a natural choice for building a mesh. Apple devices abstract Wi-Fi Direct under a multi-peer framework that makes forming a mesh nearly impossible. Since Android no longer supports Mesh or Ad Hoc Mobile modes, we look to Wi-Fi Direct as a method for building a mesh. A benefit of Wi-Fi Direct is its support for advanced power management and security features provided by Wifi Access Control mode. Figure 14 and Figure 15 depict how we might connect Android devices together using their Wi-Fi Direct interface, p2p0, and Wifi Infrastructure Mobile interface, wlan0, to build a homogeneous Wifi mesh. The next section outlines in general terms our design for implementing this mesh.



This figure shows an Android homogeneous wifi mesh network built using the Wi-Fi Direct interface and Wifi Infrastructure Mobile interface. Each line represents a connection from a wlan0 interface to a p2p0 interface. The wlan0 interface initiates the connection on the bi-directional communications link.

Figure 14. An Example Heterogeneous Wi-Fi Direct and Wifi Mesh



This figure shows how the SINETapp configures the network stack. It builds a VPN which inserts the second set of layers 3–5 within the Application layer. This VPN allows the changes of the mesh to be seamless to the user level Apps.

Figure 15. SINETapp Network Stack Flow and OSI Layer Connections

B. DESIGN OF A SOLUTION

Android devices contain all the fundamental technologies needed to create a meshed network. This section explores how to integrate these functions into an application. For simplicity, we refer to this application as the SINETapp and provide a generalized overview of the architecture in Figure 16. The SINETapp needs an "advertise and discovery" mechanism for finding other devices running the SINETapp. It requires an automated method of securely connecting to those devices. Once the SINETapp connects to other devices, it needs to act as a mesh router—discovering multi-hop routes to other non-neighbor devices and also forwarding traffic. To meet the security requirements outlined above in the Campus WLAN Capability Package, the SINETapp needs to encrypt all packets leaving the device. Either a proxy or a VPN interface

provides the second layer of encryption. Lastly, the SINETapp must provide distributed DNS capability. The following subsections describe each of these functions.

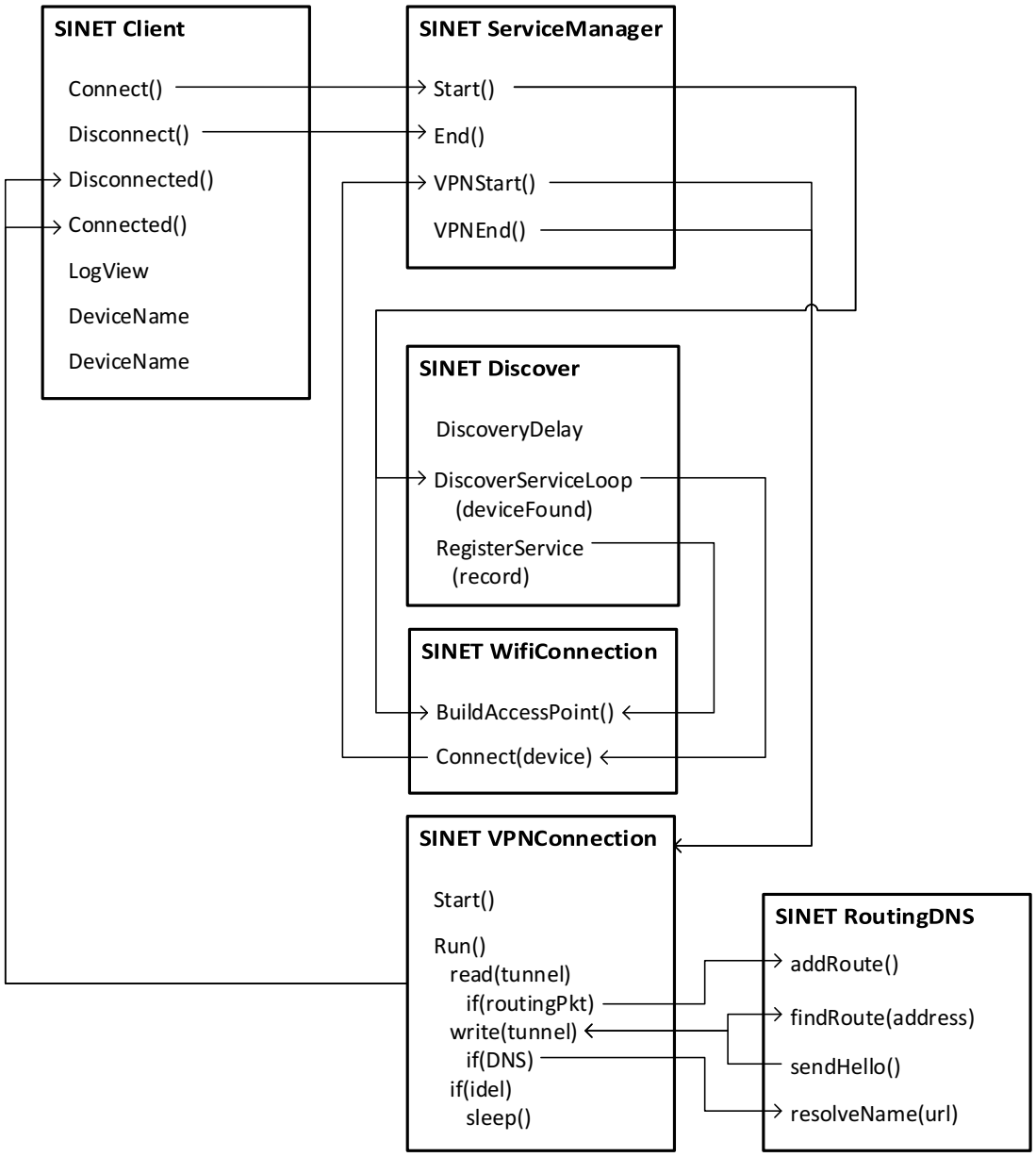


Figure 16. SINETapp Application Architecture

1. ServiceManager Class

The Service Manager class provides the switchboard from which the rest of the SINETapp functions. It runs in the background and maintains the application's configurations. It waits until the User Interface initiates a connection intent and then it runs its Start method. The Start method calls the DiscoverService method from the Discover Class and the BuildAccessPoint method from the WifiConnection class. Once a connection has been made, the Service Manager calls Start from the VPN Connect runnable class. If the User Interface sends the Disconnect intent, then everything receives a disconnect call.

2. Discover Class

For devices to find each other, the SINETapp needs to share its availability. For this, we build a Java class that extends the Wi-Fi Direct Service Discovery Library. Our discovery class contains a DiscoverServiceLoop method and a RegisterService method. The ServiceManager class calls the DiscoverServiceLoop method. Once the WifiConnection class finishes the BuildAccessPoint method, the ServiceManager calls the RegisterService method.

a. DiscoverServiceLoop Method

The DiscoverServiceLoop method must operate asynchronously to ensure that it does not wait for other functions to complete. We call it with a pointer to a method as its argument. This pointer allows us to assign actions to perform upon discovery of another device. The particular action is outlined below in the WifiConnection class. The Wi-Fi Direct Service Discovery Framework only searches when requested, so we start a delayed (calculated by equation 1) looping background thread that registers this Discover Service method with the framework.

$$\text{DiscoveryDelay} = \text{Constant} \times (\text{Connected Devices} + 1) + \text{Random Integer [1-30]} \quad (1)$$

The delay calculated in equation 1 sets how often a node should search for new networks with which to connect. The constant is 60 seconds. The randomized integer,

measured in seconds, helps reduce network flapping—avoiding all nodes becoming synchronized and making connection decisions at the same time. By multiplying the constant, also measured in seconds, by the number of inbound connections, a well-connected node evaluates its connectivity less frequently. For example, a node with four immediate neighbors will scan every 301 to 330 seconds. Since a lone or edge node has few connections, it assesses and attempts to connect more frequently. A lone node scans every 61 to 90 seconds. This weighted delay calculation stabilizes the topology by keeping central nodes more stable. The weight of the constant and span of the random integer need are chosen for the testing environment. Before application in production networks, they need research to provide optimization of energy consumption against detection delay.

b. RegisterService Method

The RegisterService method receives an object map containing free-text as an argument and registers it as a service with the Wi-Fi Direct Service Discovery Framework. This free-text contains the encrypted device name, availability number (calculated by equation 2), SSID and key for the local Wi-Fi Direct GO AP (from the Automated Connection defined below). The Wi-Fi Direct Service Discovery Framework waits until it receives a discovery request and then responds with the free-text.

$$\text{Availability} = \text{device power } [0\text{-}10] - \text{number of connected devices} \quad (2)$$

The Availability number comes from the device's battery state translated into ten levels minus the number of connected devices. Equation 2 attempts to balance the battery state against the number of devices connected. We set the step size of the battery level in a variable to allow easy adjustment and "tuning" of this parameter. More than 10 devices connected reduces the availability to zero with the step size of 10. A low availability pushes detecting devices to diversify the mesh by connecting to a different node.

c. Discover Class Summary

The Discover Service method operates as the client, and Register Service method acts as the server. Together they allow devices to detect each other.

3. WifiConnection Class

The WifiConnection class provides a BuildAccessPoint method and a Connect method. The ServiceManager's Start method calls the BuildAccessPoint method first; then it calls the DiscoverServiceLoop method. The BuildAccessPoint method calls the RegisterService method and passes it the SSID and key. The DiscoverServiceLoop receives a pointer to the Connect method. Since we are using the Wifi interface, wlan0, for connections, each device may only initiate one connection to one other device. The Wi-Fi Direct interface, p2p0, only limits the number of inbound connections to the size of its DHCP pool of 253 addresses. This process builds layer 3 point-to-point links between each node. The relationship between nodes in the mesh is many-to-one.

a. BuildAccessPoint Method

The BuildAccessPoint method calls the Wi-Fi Direct createGroup() method. It then receives the Wi-Fi Direct GO AP SSID and key which it uses to call the RegisterService method of the Discover class. We configure the Wi-Fi Direct GO AP as a WPA2 pre-shared key (PSK) AP to provide the strongest encryption available. Wi-Fi Direct automatically assigns a pseudorandom SSID of the format "DIRECT-XX-XX" and a random eight-digit alphanumeric key. Wi-Fi Direct also automatically chooses the Wifi channel to use. These are encrypted using the OpenSSL library and passed as an argument to the Register Service method of the Discovery class which allows other devices to collect the encrypted SSID and key.

b. Connect Method

The Connect Method receives the neighbor's SSID and key in an encrypted free-text packet. It decrypts the packet using the OpenSSL library to provide the SSID and key to the WifiConfig. Once configured it enables it and reconnects to it. By enabling and re-connecting, it allows the device to connect to the neighboring device's Wi-Fi Direct GO AP. In the case of multiple neighbors, equation 3 defines which neighbor with whom to connect.

$$\text{Priority} = \text{stranger?}\ [0,100] + \text{critical link}\ [50,0] + \text{signal}\ [1\text{-}10] + \text{availability} \quad (3)$$

In equation 3, we check the routing table to verify whether the detected device exists on the network already and add 100 to the equation in the case of new or isolated devices. We choose the value of 100 to provide enough weight to overcome other factors and ensure stranger nodes receive immediate attention. Next, we analyze the routing table to determine if the link provides a single threaded connection to some part of the network. If a link supports a critical link in the mesh, then we protect that link against by providing it a heavy weight of 50. The signal strength comes from the local device's detection of the other device. Availability comes through from the detected device in the encrypted package. All of these components are considered together to ensure isolated nodes receive priority, then critical paths are preserved, and lastly, energy conserved. We set the values of each component of the equation in variables to allow easy adjustment and "tuning" of parameters.

c. WifiConnection Class Summary

With the above Discovery and Automated Connection working together we can build the underlying layer of the mesh. Any lone device joins existing meshes. Segmented meshes heal due to the high priority assigned to connecting unrouteable devices. Devices with better signal strength and more battery-charge naturally become central to the mesh.

4. RoutingDNS Class

Once the SINETapp connects to other devices, it needs to act as a mesh router. It must discover non-neighbor devices for multi-hop routes. It also must correctly forward traffic along the path to the traffic's destination. We implement OLSR, discussed in Chapter II, as our routing algorithm. In addition to OLSR, we built a method that takes a node pair as an argument and analyzes the network to determine if the link is critical. The routing table contains the DNS name and MAC address of each device. This modification provides essential functionality for the DNS service defined herein.

The DNS aspect of this service allows devices to register themselves with other devices through the routing scheme defined above. When a device desires to find another device by name, it requests that device through the DNS protocol which queries the local

DNS server that in turn matches the request against the routing table. Domain names (such as http://www.google.com) that do not exist in the network cause the local DNS server to refer the request to a public DNS off-mesh. If the off-mesh DNS server is not accessible, then the request fails.

SINET mesh's namespace is of the format: [service].[devicename].sinet.[organization]. The service identity depends on the services a device provides and registers with the interface. User input, from the user interface defined below, defines the device name and organization. For example: www.micah-phone.sinet.nps.edu.

5. VPNConnection Class

To meet the security requirements outlined above in the Campus WLAN Capability Package, the SINETapp must interface with a VPN application to provide the second layer of encryption. The Android Developer API site provides the steps (Figure 17) for using the VPNService class to create a VPN tunnel. We modify step 3 of the process by instead connecting to other neighbor nodes over multicast channels. For the VPN client's network parameters, we set the following values:

- Subnet: 10.0.0.0/8
- IPv4 address: last three hex-tets of the MAC address converted to decimal in the format of 10.XX.XX.XX
- IPv6 address: by Stateless Address Auto Configuration (SLAAC)
- DNS server: 10.0.0.1
- default route: 10.0.0.1
- For example, a device with the MAC address of C2:BD:D1:16:50:CD would result in the following settings:
- IPv4: 10.22.80.205
- IPv6: fe80::c2db:d1ff:fe16:50cd

The IPv4 scheme only ensures uniqueness with devices from the same manufacturer since we discard the vendor specific part of the MAC address.

All packets destined for 10.0.0.1 route to the closest device with an external network access; except packets sent to port 53 (DNS). Packets sent to 10.0.0.1:53 go to the method described in the RoutingDNS subsection which provides name resolution.

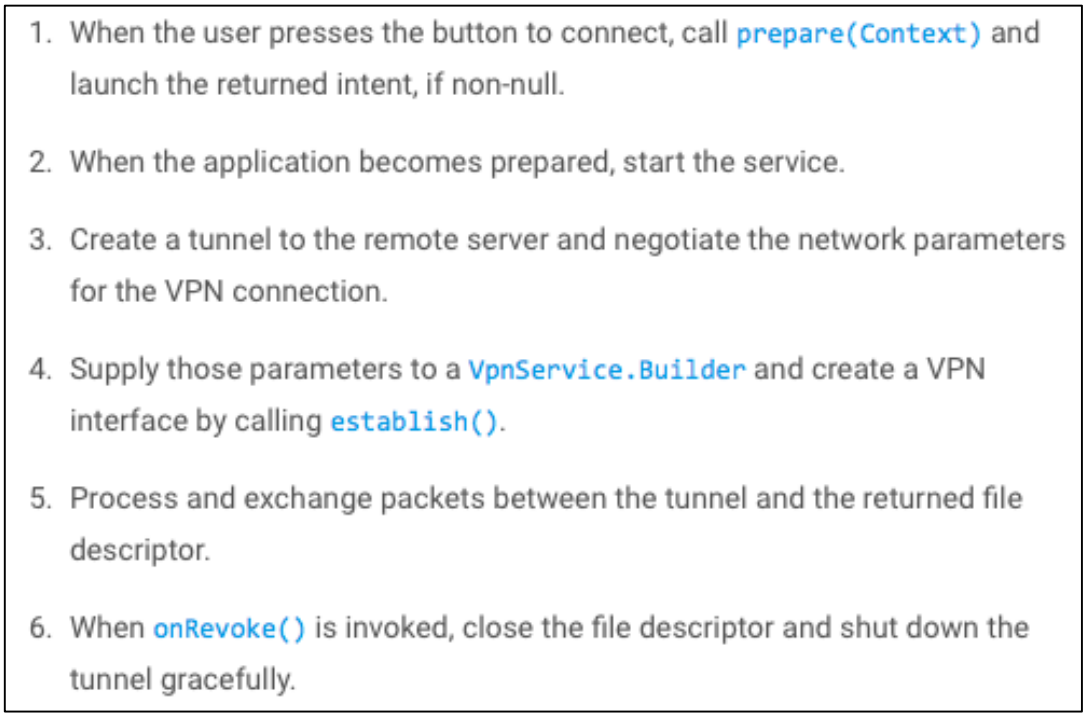


Figure 17. Steps for Creating an Android VPN Tunnel. Source: https://developer.android.com/reference/android/net/VpnService.html.

For encryption, we encrypt outbound packet payloads using a symmetric key build by encrypting a pre-shared secret with the destination's PKI-based Public asymmetric key. Inbound packet payloads are unencrypted using a symmetric key after decrypting the pre-shared secret with the recipient's PKI-based private asymmetric key and injected into the VPNService's interface.

6. User Interface

Since the purpose of this thesis is to provide a simple interface with intuitive configuration options, the SINETapp interface provides a simple interface consisting of text boxes to input the device's name and pre-shared key, and green "connect" button. Once connected, the input fields become locked and the button changes into a red

"disconnect" button. A log screen, available from the menu, displays a trace of the device's connection and routing activity to assist in troubleshooting.

7. Summary of Design Overview

The purpose of the SINETapp is to provide a simple-to-use application that allows mobile devices to connect to each other in a secure mesh that meets the DOD's requirements for transmission security. Utilizing the service discovery methods and Group Owner methods of Wi-Fi Direct and the traditional Wifi Infrastructure Mobile connection mode we build the basis of a mesh network. Once the mesh is in place, we provide a hybrid routing and DNS system that allows devices to find each other with user-friendly names. To secure this mesh, we overlay a secure layer by implementing a peer-to-peer VPN using the VPNService interfaces. The result is a simple-to-use application that enables first responders or military members to communicate in an infrastructure-less environment.

C. SUMMARY OF DESIGN

This chapter provided a high-level overview of the design by which we intend to implement the security requirements through technologies organic to modern Android devices. We design a homogeneous Wifi mesh that supports MANET routing protocols and methods of storing DNS information in MANETs without centralized servers. In the next chapter, we explore the implementation of this design and our lessons learned from testing it on Android devices.

IV. IMPLEMENTATION AND TESTING

In this chapter, we outline the fundamental and practical barriers obstructing our implementation of the SINETapp as laid out in Chapter III. We had hoped that Wi-Fi Direct would provide an energy efficient approach to building a mesh of Android devices. We find that the security and power features built into Android 6.0 (Marshmallow), API 23, make it impossible to create a routable homogeneous Wifi mesh with Samsung Galaxy Note 4 mobile devices. In the second half of this chapter, we suggest two approaches to overcoming these limitations. The first approach requires root permissions on the devices to modify the Wi-Fi Direct configurations. The second method takes an entirely different approach and builds the mesh on embedded devices, thus confounding the original intent of building the mesh entirely of homogenous COTS devices, that is, without non-organic communicating entities such as external radios or networking devices. Additionally, we evaluate the strengths of each approach.

A. IMPLEMENTATION OF PROTOTYPE

We follow the agile development model for developing up small pieces of code to validate capabilities before bringing the parts together into a larger construct. We develop modules of the SINETapp and concurrently test them by monitoring network traffic with wireless capture cards. This technique allows us to identify a few poorly documented or undocumented API "features." This agile develop-and-monitor technique yielded the following results. We separate these results into three categories: theoretical limitations, practical limitations, and positive results.

1. Theoretical Limitations of the SINETapp Design

Some of the limitations we found while implementing the SINETapp are fundamental to its design and not unique to the Android device's vendor or Android version. These limitations are common across the majority of mobile devices or limitations imposed by the nature of the technology. We highlight frequency utilization and power consumption as notable limitations introduced by the project's design.

a. Frequency Utilization

In our testing, we notice that Samsung Android devices (Galaxy Note 4 and Galaxy 6S) use Channel 6 when setting up a Wi-Fi Direct GO AP. Once the device configures and enables the GO AP, it blocks wlan0 from connecting to any AP on any channel other than Channel 6. This limitation derives from the constraint that mobile devices contain only one Wifi antenna. If the device tried to monitor two channels at once with a single antenna, it would have to switch back and forth very quickly causing it to miss some packets. This limitation means that every device on the mesh network will be operating exclusively on Channel 6. Figure 18 shows our Kali Linux penetration-testing box running Airodump with an Alfa Wifi card to capture the wireless network traffic generated by SINETapp.

BSSID	PWR	Beacons	#Data,	#/s	CH	MB	ENC	CIPHER	AUTH	ESSID
C2:BD:D1:77:39:C6	-31	316	1	0	6	54e	WPA2	CCMP	PSK	DIRECT-2f-C6
EE:1F:72:7A:63:B4	-33	439	2	0	6	54e	WPA2	CCMP	PSK	DIRECT-K8-2
C2:BD:D1:16:D0:CD	-36	343	0	0	6	54e	WPA2	CCMP	PSK	DIRECT-ik-CD

The Airodump result from three Android devices running SINETapp. The “CH” column indicates the channel each device is operating on. All three are using channel six.

Figure 18. Airodump Results of Scanning SINETapp

Since every link of the mesh runs on the same channel (a shared collision domain), as the network grows the quantities of collisions will also grow which drives down useful throughput. Hence, the network does not scale well. If the nodes of the mesh are tightly packed (geographically), the effect grows even worse as the added nodes compete for the limited resource of one Wifi channel. If geographically widely separated, the nodes might re-use the channel without colliding with each other.

This limitation comes from SINETapp’s design constraint of building a homogeneous Wifi mesh out of mobile devices which generally only contain a single Wifi antenna. Using devices with multiple Wifi antennas or by switching to heterogeneous mesh network constructed with various interface types (Bluetooth and Wifi for example) would solve this limitation.

b. Energy Consumed Operating in Access Control Mode

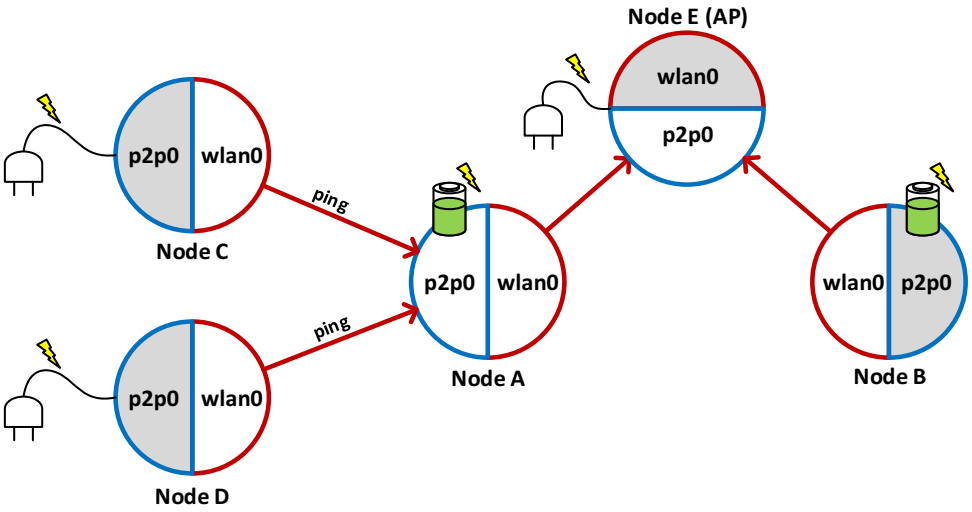
As we implemented and tested SINET we guessed that operating a Wi-Fi Direct GO AP would consume a significant amount of energy. We decided to informally test our hunch. Since it does not form a critical part of our research objective, we did not attempt to perform a rigorous energy consumption test. Specifically, this test lacks the controls and repetitions needed to provide scientifically reproducible results.

Disclaimers: The devices used are the same model and age, but the battery capacity of both phones may not be the same. Further, we measured the battery level by reading the percentage reported by the device, which may be limited in its accuracy. We also only ran the test once in the configuration depicted in Figure 19. This test should only be considered interesting and not rigorous.

With these disclaimers, we present a crude experiment to explore the general sense of energy consumption of Wi-Fi Direct. We hypothesize that operating an AP requires the hosting device to keep the Wifi card constantly "awake" to respond to BEACON messages and receive traffic from connected devices. Staying "awake" consumes more energy than "sleeping" the Wifi card between uses.

To measure the severity of Wi-Fi Direct's energy consumption, we set up a testbed according to Figure 19. Nodes A and B start side by side with a full charge. We turn off all other apps and restart the both nodes. Once we verify both nodes are in nearly identical states, we load SINETapp. They exchange information with the other nodes in the testbed and set up their Wi-Fi Direct GO APs. We then disable the Wi-Fi Direct GO AP on Node B. We then connect Node A and B to Node E's Wi-Fi Direct GO AP to simulate both devices connecting to the mesh. We connect Nodes C and D to Node A's Wi-Fi Direct GO AP. On Nodes C and D, we run a continuous ping to Node A. For the first 40 minutes, we monitor Node A and B with their screens on which forces an "awake" state. We note that the "awake" state and screen consume significant energy on both devices. After the first 40 minutes, we turn off Node A and B's display to further isolate the energy consumption of Node A's Wi-Fi Direct GO AP. For the next 12 hours, we monitor the testbed by periodically waking Nodes A and B to verify their connectivity

to Node E. Keeping Nodes A and B connected to Node E, simulates a realistic outbound connection. At the end of 12 hours, we took screen captures of both nodes' energy consumption history depicted in Figure 20. We combined the information into a single graph depicted in Figure 21.



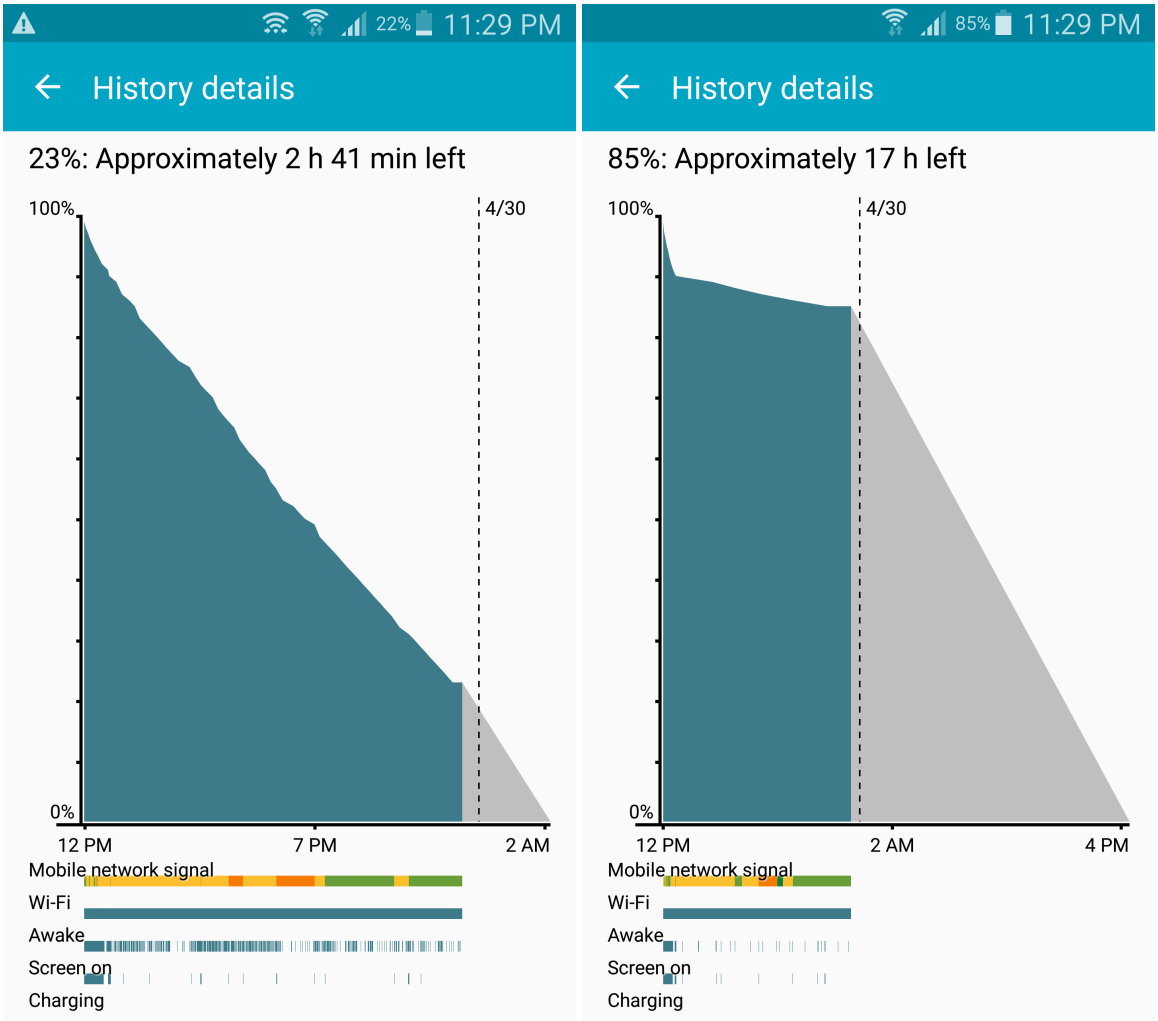
Interfaces filled gray are inactive. Battery and plug symbols indicate power supply for each device. We measure the energy consumption of Nodes A and B to determine the energy required to operate a Wi-Fi Direct GO AP. Nodes C, D, and E stay awake while plugged into wall power to ensure their constant activity.

Figure 19. Topology of Wi-Fi Direct GO AP's Energy Consumption Test

To calculate the results of energy consumed without the screen active, we focus only on the period of 40 minutes through 11.5 hours, shown in Figure 21. Node A, running the Wi-Fi Direct GO AP, consumed 68% of its battery energy in 10.83 hours equaling a rate of 6.28% per hour. Node B, during the same period, consumed only 5% of its battery energy equaling 0.46% per hour. This test suggests that an Android device running a Wi-Fi Direct GO AP, even with its screen off, consumes significantly more energy than a device not running the Wi-Fi Direct GO AP.

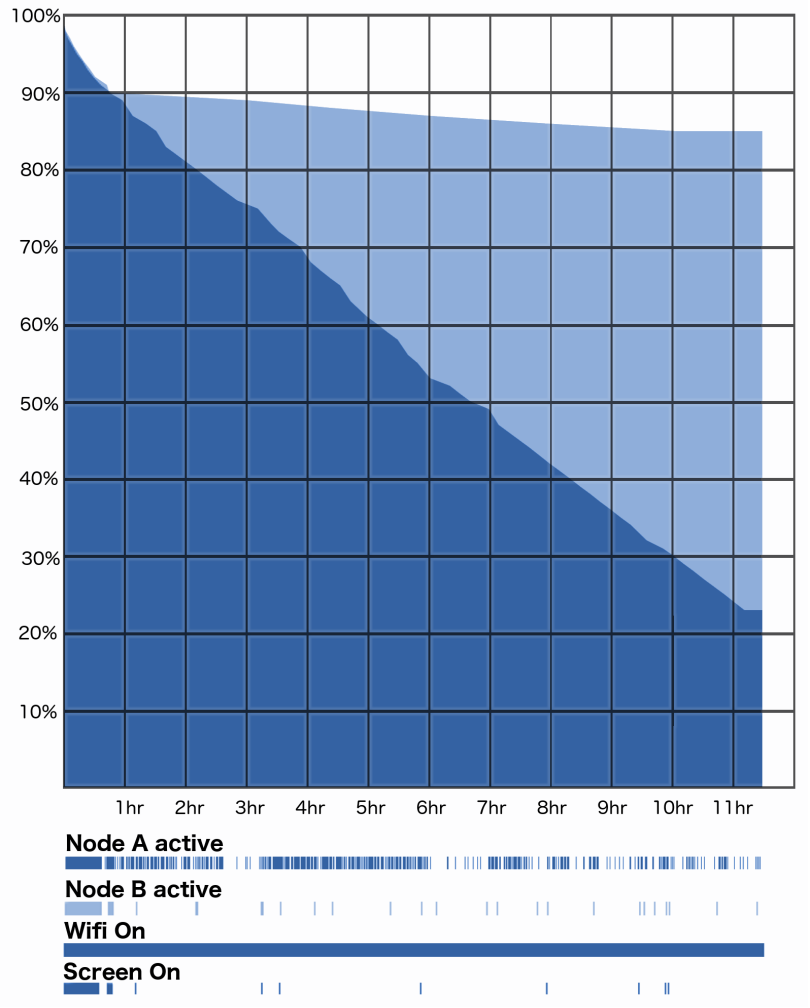
Building a homogeneous Wifi mesh introduces the inherent energy cost of running APs on each node. With our design model, we must accept this energy cost. To avoid the energy cost would require that we change the nature of the mesh to a heterogeneous mesh. In the second half of this chapter, we suggest an alternate method

which removes this energy cost from the end-user mobile devices freeing them to perform only tasks that directly support the user.



The left-hand screenshot depicts Node A's energy consumption history. The right-hand screenshot depicts Node B's energy consumption history.

Figure 20. Wi-Fi Direct GO AP's Energy Consumption Screenshots



The dark blue area represents the power remaining in Node A while providing a Wi-Fi Direct GO AP. The light blue area represents the energy remaining in Node B in normal operation.

Figure 21. Wi-Fi Direct GO AP's Energy Consumption

c. Summary of Theoretical Limitations

Our design of the SINETapp had to contend with the fundamental limitations of Wifi channel utilization and imposed a power consumption cost upon devices running APs. To overcome the issue of all devices operating on the same channel, we suggest multi-antenna devices or moving to a heterogeneous mesh. To overcome the power consumption issue, we suggest utilizing devices dedicated to operating APs. This approach, as with the use of a heterogeneous mesh, confounds the desire to utilize a single device per user-entity. These suggestions shape our alternate approaches later in this chapter.

2. Practical Limitations of SINET's Design

While implementing the SINETapp, we also discovered Android API version 23's specific limitations that may in the future be removed by Android software updates. These deficiencies include: how the Wi-Fi Direct DHCP server issues IP addresses, issues with IPv4 unicast on devices with redundant network routes, constraints with IPv4 multicast on Wi-Fi Direct interfaces, and limitations with IPv6 sockets using SLAAC addresses. Below, we describe these limitations in detail and suggest methods for overcoming each.

Since these restrictions exist as artifacts of the Android API version 23 implementation, modifying Android's source code might overcome the shortcomings. Unfortunately, this presents a challenging and time-consuming task that does not guarantee that Android's source code curators will incorporate the changes into the master repository. If the changes were not incorporated, the effort would be wasted, as it would require third-party oversight of the affected, essentially rooted, code.

It is our opinion that modifications to Android must support the business goals of the cellular service companies who sell Android phones. If the modification conflicts with that, the modification will never be incorporated into the master repository no matter how worthy. Since mesh networks would make metering cellular data challenging, the limitations may remain largely unchanged in the Android master repository.

a. Manual Intervention Required to Connection to Wifi Networks

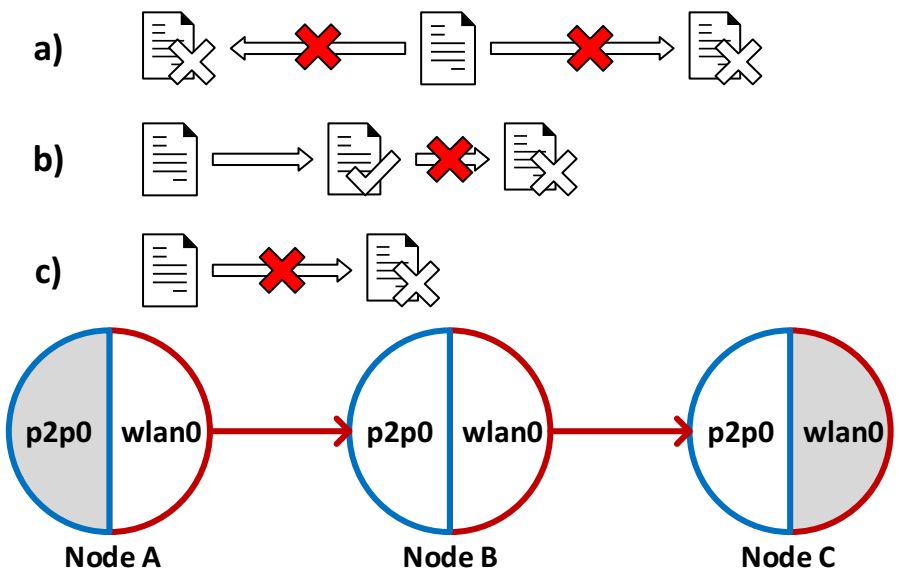
The first step in building this mesh requires Android devices to connect to another device programmatically without requiring user input. To implement this, we configure an AP and load its connection settings (SSID and key) into a WifiConfiguration variable. Then we tell the WifiManager to enable() and reconnect() to that AP. As a security precaution, Android will not connect to it until the user visits the Wifi settings dialog and sees the connection. Once connected, it will automatically reconnect without user input. This automatic reconnection holds true while the WifiConfiguration remains unchanged. If we change the WifiConfiguration, then the user must view the settings again before it reconnects to the Wifi AP.

This security feature forces the user to manually open the Wifi Settings dialog before the mesh can automatically connect. Requiring user interaction defeats the purpose of the automatic configuration. It also defeats one of the core purposes of this project, which is to provide an autonomous capability, that is, an automatically-connecting mesh.

To overcome this limitation, we need a stable WifiConfiguration for devices connecting to the mesh. A stable WifiConfiguration requires either a shared static pre-shared key or using device-specific keys.

b. Wi-Fi Direct Fixed IP Subnet and Lack of Routing

As we build out the Discover and WifiConnection classes specified in Chapter III, we find that Wi-Fi Direct uses the 192.168.49.0/28 subnet each time it builds a GO AP. *This static setting causes every link in the mesh to have the same subnet.* Also, Wi-Fi Direct does not add routing statements to bridge or forward packets between the p2p0 interface with the wlan0 interface as depicted in scenario b) of Figure 22. *This lack of routing modifications means that Android devices cannot act as mesh routers without a workaround.*



Scenario a) Node B cannot determine which interface (wlan0 or p2p0) to open a unicast socket on to connect to either Node A or Node C since both networks have the same subnet.

Scenario b): Node B will not forward packets between interface p2p0 and wlan0 because it lacks routes and forwarding rules.

Scenario c): Node B filters all inbound Multicast packets on its p2p0 interface.

Figure 22. Android Devices Failing to Bridge Groups

Funai, Tapparello, and Heinzelman note this same problem in their paper, titled "Supporting Multi-hop Device-to-Device Networks Through WiFi Direct Multi-group Networking" (Funai et al., 2017). They offer two solutions for overcoming this problem: relay devices and modifying the Android source code.

First, they add a relay device that connects to two Wi-Fi Direct groups simultaneously with its wlan0 and p2p0 interfaces depicted in Figure 23. This relay device ferries data between the two groups using a variety of the teeter-totter schemes for overcoming the redundant IP subnets (Funai et al., 2017). This technique creates a brittle mesh that heals poorly after insertion of nodes. After any change of topology, the mesh must re-converge to discover which nodes should perform the roles of Relay and GO. While the mesh re-converges, routes over any affected area fail. This technique may work for delay tolerant networks, but would cause our standard TCP/IP network to fail. We reject this technique because it requires devices to play different roles depending on

their context, which violates our premise of creating a homogenous mesh network out of homogeneous nodes. We also reject it because it introduces significant additional instability to the mesh at the link layer.

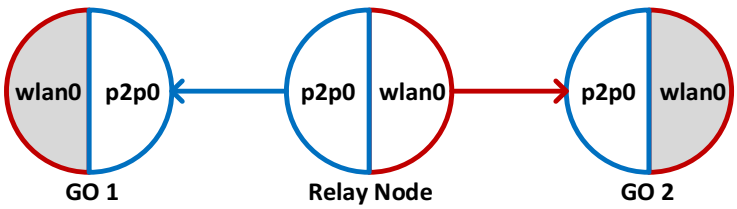


Figure 23. Relay Node Ferrying Messages between Two Groups

Second, they suggest modifying the Android source code to change how Wi-Fi Direct sets up the subnets and configures the DHCP service (Funai et al., 2017). They successfully modify the Android source code on a version of Android before version 5.0. We attempt the same changes on Android 6.0 (Lollipop, API 23) but Android source code no longer contains WifiP2pService. In fact, Google rewrote Android 5.0's Wifi networking package, and it no longer offers the same flexibility for modifying the WifiP2pService. We downloaded the most recent Android source and tried to replicate the changes to the default Wi-Fi Direct configurations without success. We discovered that Android, after version 4.4, baked the IP address into the libraries that interface between the Androids Java environment and Android kernel. This removed the relatively simple method of customizing the IP network settings that Funai et al utilized.

We find that the Android kernel still responds to ifconfig commands from the ADB shell, but Android does not provide an API to control it natively. Soares, Brandão, Prior, and Aguiar (2017) successfully get around this problem using reflection to internal APIs and root permissions on a Gigabyte Gsmart G1305, a Samsung Nexus S and a Samsung Galaxy Tab 10.1 to create 802.11 IBSS (Wifi Ad Hoc mobile) connections. They build an app called AdHocDroid which controls the device's interfaces and implements an OLSR daemon. They find that each Android device maker uses different drivers, which makes their app only work on certain devices. Considering the fragility of their approach we decide to not follow their method.

While neither of Funai, Tapparello, and Heinzelman's suggestions worked in our case, they did provide us the idea of using Broadcast or Multicast to bridge the groups. In the next two subsections, we describe our attempts at bridging the groups.

c. IPv4 Unicast for Bridging Two Groups

As we see from above, a homogeneous mesh built using Wi-Fi Direct and Legacy Wifi limits us to re-using the 192.168.49.0/28 subnet for every link. Since every link has the same IPv4 subnet, we found that Android will not open any Unicast sockets or Broadcast IPv4 sockets while the Wi-Fi Direct GO and Legacy Wifi connections are both active within the mesh (depicted in Figure 24.) If we disable the Wi-Fi Direct GO, then Unicast and Broadcast become available again. When both links are active, the routing table contains entries for both networks making it impossible for Android to determine on which interface to open the socket. Further, Android Unicast and Broadcast socket libraries do not support specifying the interface. Thus, IPv4 Unicast and Broadcast are not available for bridging groups, leaving us with either IPv4 Multicast or IPv6 for transport.

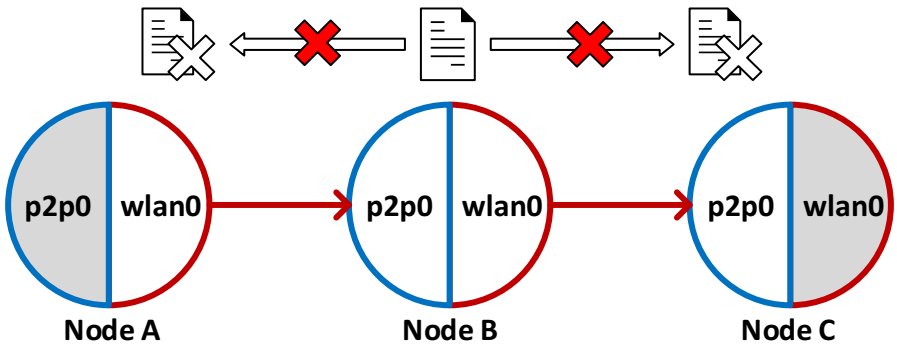
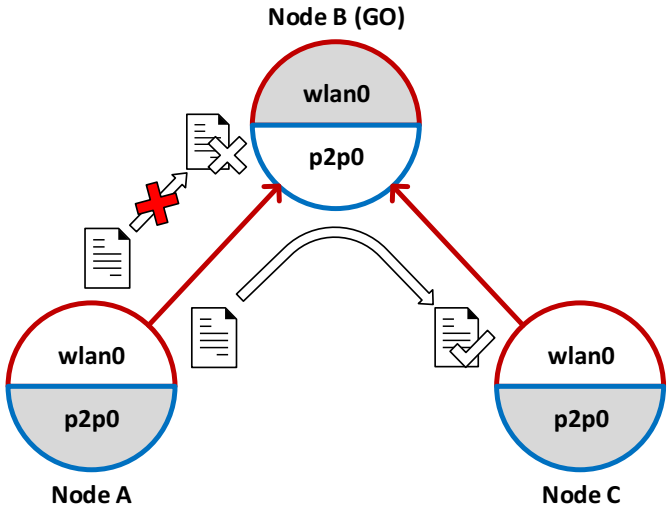


Figure 24. SINET Node Fails to Open IPv4 Unicast Sockets

d. IPv4 Multicast for Bridging Two Groups

After attempting to bridge the wlan0 and p2p0 interfaces using IPv4 Unicast and Broadcast sockets, we next try using IPv4 Multicast sockets. The IPv4 Multicast libraries support specifying the outbound interface. We can successfully send Multicast between

the Node A and Node C as depicted in Figure 25. Node B (the GO) does not allow multicast packets to come up the network stack but filters them at the IP level. Android API 23 provides the method WifiManager.MulticastLock() for removing the multicast filter from the wlan0 interface. Android does not provide a corresponding method for the WifiP2pManager class that manages the p2p0 interface. Since the p2p0 interface filters multicast packets, we cannot use multicast to bridge multiple groups in our mesh.



When Node A sends a multicast packet out on interface wlan0, Node B will forward it to Node C but will filter it from itself. Wi-Fi Direct filters out multicast traffic. Similarly, Wi-Fi Direct does not accept IPv6 connections but will allow IPv6 packets to travel from Node A to Node C.

Figure 25. Multicast Traffic and IPv6 Traffic with Wi-Fi Direct

e. IPv6 for Bridging Two Groups

In Android version 6.0, Wi-Fi Direct does not issue IPv6 addresses through its DHCP service. When a device connects to a Wi-Fi Direct AP, its wlan0 interface only shows an IPv6 SLAAC address. On the Wi-Fi Direct device, its p2p0 interface also only shows an IPv6 SLAAC address. When a host uses an IPv6 SLAAC address to connect to another host in Linux, the host must specify which outbound interface to use.

We first try running a "ping6 –I wlan0 fe80::c2bd:d1ff:fe77:b9c6" command from Node A to Node B using node B's p2p0 SLAAC address, with the same topology as

depicted in Figure 25. The ping6 command fails to receive any response from Node B's p2p0 interface. We next try a "ping6 –I wlan0 fe80::c0bd:d1ff:fe16:50cd" command from Node A to Node C's wlan0 SLAAC address. This time the ping6 command succeeds, thereby proving IPv6 works on the Android devices.

Next, we attempt to implement IPv6 socket connections in our SINETapp. We are unable to create unicast connections to any other device using IPv6 SLAAC addresses in Java on Android version 6.0. As we see with the IPv4 unicast, Android attempts to determine the outbound interface for sockets automatically. This automation conflicts with the Linux requirement of defining the interface for IPv6 SLAAC connections.

We next attempt to create IPv6 multicast connections with SLAAC addresses. We experience the same filtering as see with IPv4 multicast on the p2p0 interface. Having exhausted the possible methods for connecting over IPv6, we determine that IPv6 did not provide us any additional capability for bridging two groups in a homogeneous Wifi mesh network.

f. Summary of Practical Limitations

Our design objective of building a homogeneous Wifi mesh network using only Android mobile devices results in an impasse due to practical limitations. First, we are unable to connect to the mesh without user interaction for each new connection. Second, we find that Wi-Fi Direct utilizes the IP subnet of 192.168.49.0/28 each time it builds a new group. It also does not insert the needed routing statements to allow traffic to bridge between groups. We attempt to overcome these routing limitations by unwrapping each packet and manually forwarding them using an IPv4 unicast, IPv4 multicast, IPv6 unicast, and IPv6 multicast. None of these approaches allow us to bridge between two groups. Thus, we cannot build a homogeneous Wifi mesh network with mobile devices running Android version 6.0.

3. Positive Aspects of SINET's Design

In this next section, we explore the positive aspects of our research. While Wi-Fi Direct proves a poor choice in its current implementation from the aspect of IP

assignment, it does provide robust security and excellent service discovery mechanisms. We highlight these two areas in the following sections.

a. Testing SINET's WPA2 Security

Once Wi-Fi Direct sets up the GO AP it chooses a random eight-digit upper and lower case alphabetic pre-shared key (PSK). The security of the Wifi resides in the $(26 + 26)^8 = 5.3 \text{ x } 10^{13}$ possible combinations of PSKs. Using Kali Linux, in a Virtual Machine, provisioned with 1x 2.8 GHz processor core and 1024 MB of random access memory, we brute-force guess PKSs at a rate of 1230.88 PSKs per second. At this rate, our Kali Linux Virtual Machine would take 684.5 years on average to guess the WPA2 PSK. Wi-Fi Direct's scheme appears sufficiently robust and meets the requirements of the Capability Package.

When we discuss alternate methods later in this chapter, we are no longer constrained to Android's Wi-Fi Direct implementation and can increase the security by defining a longer and more complex PSK.

b. Wi-Fi Service Discovery for Automated Connections

The Wi-Fi Direct Service Discovery mechanism proves robust and very handy in building the automated device detection. We appreciate its reliability and simplicity. Passing the PSK in an encrypted packet significantly reduces the requirement for humans to type in complex passwords. It also allows devices to detect each other effectively.

c. Summary of Positive Aspects

Wi-Fi Direct's WPA2 security and Service Discovery mechanisms impress us in their simplicity, security, and robustness. We highly suggest follow-on non-Android centric mesh projects leverage these capabilities. Together they provide a secure way to automate device connection configuration.

4. Summary of Implementation Results

Through the process of implementing the design prescribed in Chapter III, we discovered significant limitations in Wi-Fi Direct with respect to Android version 6.0 that

prevented the construction of a homogeneous Wifi mesh network on that platform. We discovered fundamental limitations, which included channel utilization and power consumption, while running an AP on a mobile device. We also found Wi-Fi Direct reuses the same IP subnet every time it constructs an AP on this platform. The Android version 5.0 and 6.0 re-write of the Wifi module blocks one of the previous workarounds for this limitation. Also, we explored four other approaches to bridge Wi-Fi Direct groups that proved unsuccessful. These limitations in Android version 6.0 stymied our efforts to build a homogeneous Wifi mesh network over Android mobile devices. In the process of this research, we found Wi-Fi Direct's choice of WPA2 PSKs robust, and we found the Wi-Fi Direct's Service Discovery mechanism provides a significant capability. In the next section, we suggest two approaches for overcoming infrastructure-less environments with secure MANETs.

B. ALTERNATE APPROACHES TO SECURE MANETS

Since the COTS Android 6.0's implementation of Wi-Fi Direct makes it impossible to create a routable homogeneous Wifi mesh, we next discuss how future projects might overcome these limitations. The first method involves creating custom kernel modules for Android. The second method alters our constraints and builds a homogeneous mesh of "helper devices" to which mobile devices connect; that is, the structure of the mesh is implemented separately from the Android smartphone devices that connect to the provisioned network.

1. Rooted Devices

Before delving into this section, we believe it is critical to highlight the numerous hacked-together Android modifications that litter the Internet. Our naivetés would incline us to think that our modification will end up as a glorious revision added into the master Android repository. In reality, only the patches that support the business goals of the cellular service providers will find traction. If Google does not include a kernel modification or patch into the master repository, it will not receive updates. If not included, the patch will require significant effort, by a third-party, to keep current as a branch, or it will die.

With that disclaimer, we think there are two ways to approach this building of a secure homogeneous Wifi mesh network with rooted Android devices, that is, Android devices where the core kernel has been modified outside Google control. First, implement an 802.11s Mesh networking kernel module for Android. Second, leverage the prior work by Soares, Brandão, Prior, and Aguiar (2017).

a. 802.11s Mesh Kernel Module

Building an Android kernel module for 802.11s Mesh would require a significant amount of work. Our brief exploration of the topic shows very little previous work. Not only would the kernel module need to be developed, but a means of controlling it from the Android Java environment would be necessary. Soares, Brandão, Prior, and Aguiar provide a method for executing shell commands that might help in this effort (Soares et al., 2017). The kernel module would need to incorporate a mechanism for running a RADIUS server to satisfy the NSA's requirement for unique authentication and encryption per connection. Such a service would also need to be distributed, as described for the DNS service earlier, such that it is readily available throughout the established mesh regardless of link failures.

Building an 802.11s Mesh kernel module for Android would abstract the problems of building and maintaining the mesh from the DNS and VPN security layers mentioned above in Chapter III. IEEE 802.11s Mesh specifications require hardware layer routing within the mesh to provide two virtual LANs (VLAN) to the IP layer. This abstraction would increase the modularity of the project. It would also require the VPN and DNS to map the mesh themselves, which would duplicate the network discovery and mapping messages. The trade-off between modularity and duplication of effort may require careful consideration.

b. Leverage Soares, Brandão, Prior, and Aguiar's Work

Soares, Brandão, Prior, and Aguiar's work provides a mesh over Ad Hoc links as well as a mesh-oriented routing protocol (Soares et al., 2017). A follow-on project would need to modify their work to control the Wi-Fi Direct interface. Wifi Ad Hoc mode does not meet the NSA's requirement for unique authentication on each link. A follow-on

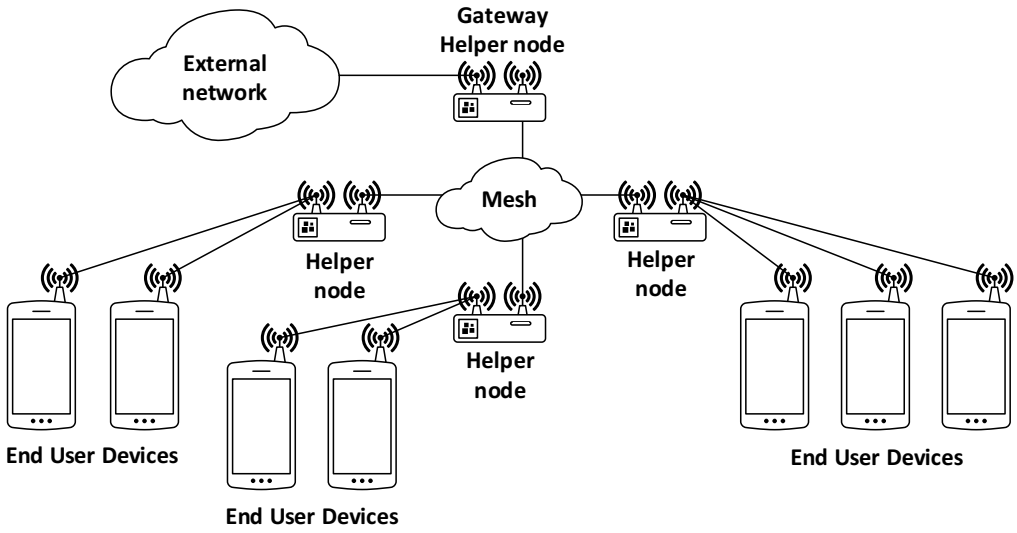
project would need to also implement a secure peer-to-peer VPN and DNS on top of their protocol's mesh.

c. Rooted Devices Summary

Rooting Android allows a developer to access functionality that otherwise is inaccessible at the risk of having a device no longer supportable by the Google support base. By rooting Android, a future project could successfully build a secure homogeneous Wifi mesh. We propose two paths forward but caveat those ways with a strong warning about the fragility of developing on a rooted device. The next subsection offers a different approach that would avoid these concerns.

2. Helper Devices

Since Android version 6.0 blocks multi-hop routing over a homogeneous Wifi mesh network, we need to change a core constraint for this project. We suggest changing the homogeneous mesh to a heterogeneous network. The core of this network would consist of a homogeneous mesh of helper devices connecting over a meshed Wifi backhaul as depicted in Figure 26. The mesh helper network would control access as seen in Figure 27, provide IP routing, and act as a distributed DNS service. The helper devices would need at least two antennas to allow the backhaul mesh to run on a separate channel from the APs for the end-user devices (EUDs). The EUDs would connect to the helper devices and build a p2p VPN on top of it, as depicted in Figure 30.



Helper nodes need to have two antennas to allow one to provide an EUD AP and the other for connecting to the backhaul mesh network.

Figure 26. Example Helper Device Mesh Network

For access control, the helper mesh will need a RADIUS server, as depicted in Figure 27 to satisfy the NSA's requirement for uniquely keyed WPA2 connections. The RADIUS server would validate devices using either a replicated or distributed key-store containing all authorized devices' public keys. Use of the PSK would allow the operator to rapidly re-key the mesh. WPA2's challenge and response mechanism would protect against playback attacks.

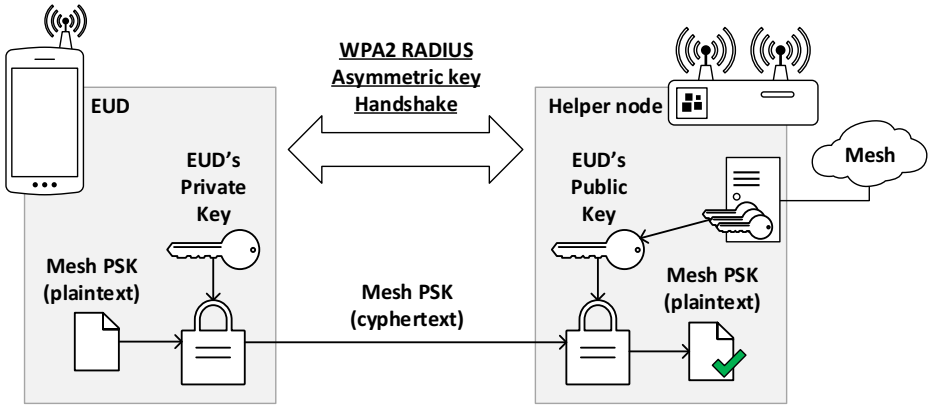


Figure 27. WPA2 RADIUS Asymmetric Key Handshake

For IP routing, we suggest OLSR routing implemented at the application level. Further, we suggest augmenting OLSR to carry the hostname with the IP address of end nodes in its routing maintenance messages. Carrying the hostname of the devices would allow the helper devices to easily provide a distributed DNS service for the mesh.

The EUD would run a p2p VPN client that would pull DNS information from the helper node's DNS service, as depicted in Figure 28. This VPN client would provide the second layer of the NSA-required dual layers of encryption. Since each layer must provide EUD specific encryption and authentication, we propose the hybrid encryption technique as depicted in Figure 29 as a means of authenticating and encrypting the tunneled traffic for the p2p VPN. Our hybrid technique uses the strength of asymmetric keys to share the symmetric stream cipher key used for encrypting packets. The symmetric stream cipher key provides the efficiency of stream ciphers for bulk encryption of the packets. This technique works well for unicast traffic. This concept might be modified to enable multicast and broadcast encryption by removing the destination's key from the scheme.

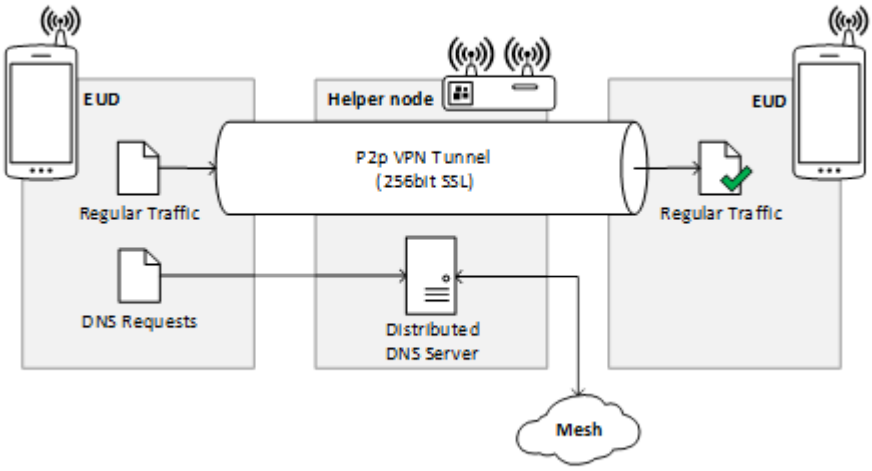


Figure 28. Helper Device Providing OLSR Routing and DNS Services

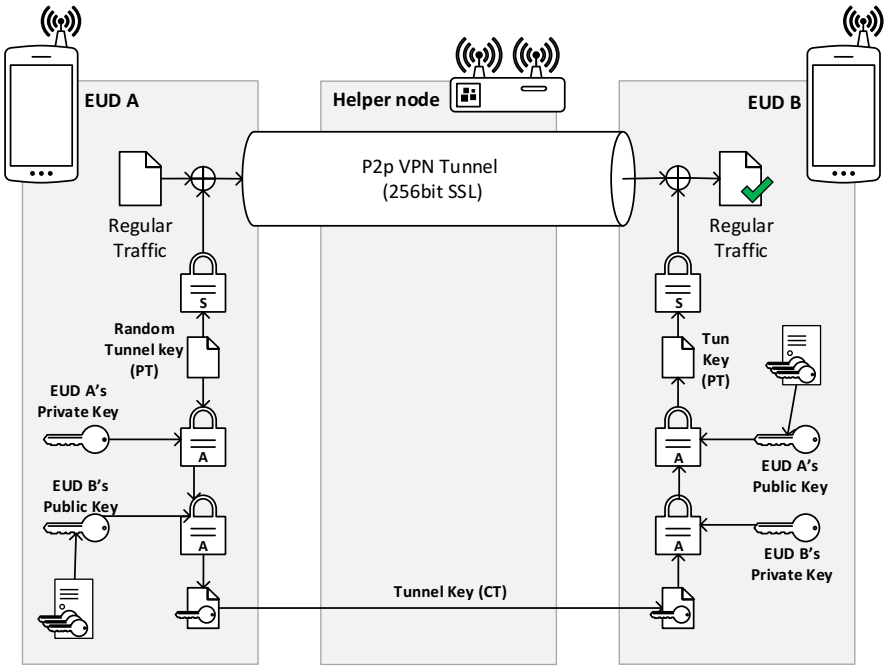


Figure 29. Example of Hybrid Encryption Scheme for P2p VPN

Our proposed helper mesh network introduces a challenge in providing a secure method for configuring and modifying the PSK and distributed key library not seen in a pure homogeneous Wifi mesh of EUDs. A secure configuration side-channel would be required to address this. Wi-Fi Direct and the Service Discovery techniques we discuss in Chapter III might offer a means for building this secure connection.

In this proposed method, we suggest using a homogeneous mesh of helper devices. This mesh of helper devices provides distributed and secure AP and DNS. EUDs can leverage the helper mesh to operate a p2p VPN. This solution meets the NSA's requirements for dual encryption for wireless network solutions

.

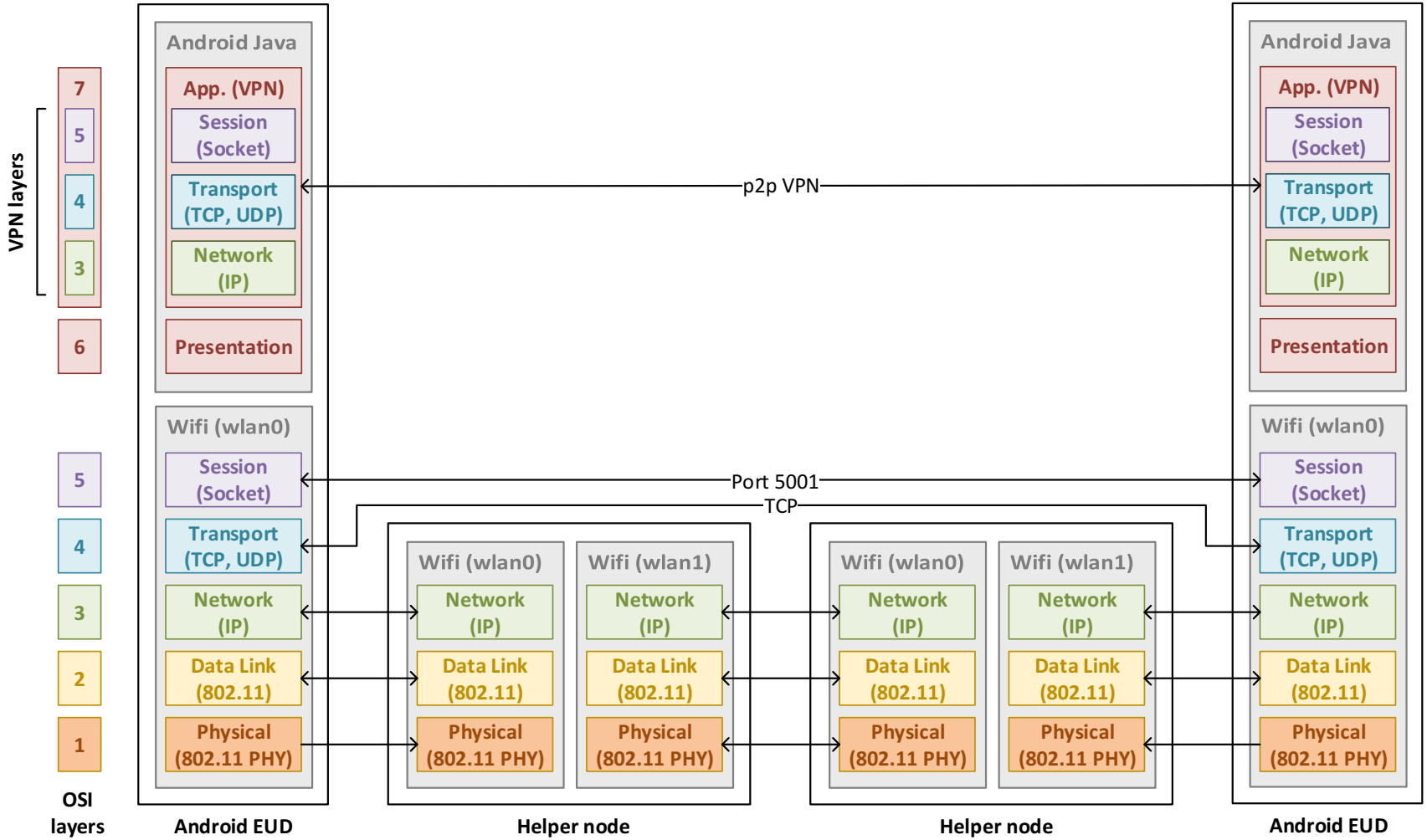


Figure 30. Network Stack Flow of EUD Based P2p VPN and Helper Nodes

3. Summary of Alternate Approaches

In this section, we discussed two ways to overcome the limitations of building a homogeneous Wifi mesh network with Android devices. With warnings against the fragility of the approach, we suggested the option of gaining root access and extending the capabilities with custom kernel modifications. We think that the better approach would be to build a mesh of helper devices.

C. IMPLEMENTATION AND TESTING SUMMARY

In this chapter, we discussed the fundamental and practical limitations to implementing a homogeneous Wifi mesh network with Android 6.0 (Marshmallow), API 23 mobile devices. Specifically, we discussed how Wi-Fi Direct provides excellent security and discovery features but blocks devices attempting to bridge multiple Wi-Fi Direct groups by using the same IP subnet for every Wi-Fi Direct group. Devices connected to two Wi-Fi Direct groups cannot open IPv4 or IPv6 unicast sockets. Also, the Wi-Fi Direct GO device cannot receive IPv4 or IPv6 multicast packets. We then discuss two approaches for overcoming these limitations. First, building a custom kernel for Android that provides 802.11 mesh mode. Second, building a mesh using helper devices to offload the functions of building a mesh to dedicated devices. Both of these techniques hold promise as future work.

V. SUMMARY AND CONCLUSION

This research explored how we might enable first responders and military C2 with a secure, well-connected, lightweight, and mobile handheld computing device using a simple and familiar interface through a secure homogeneous Wifi mesh built with COTS Android 6.0 mobile devices.

In Chapter II, we first examined the stringent encryption requirements for secure communication and the current military radios that meet the security requirements. We found the current solutions to be expensive and inconvenient. Next, we surveyed wireless technologies found in COTs devices and the MANET concepts needed to interconnect them: mesh types, MANET routing protocols, and approaches to storing data in MANETs. From this survey, we decided to leverage the security, bandwidth, and range of Wifi and Wi-Fi Direct with WPA2 to build a secure mesh of Android devices. We noted that Android does not support mesh networks or connecting to multiple Wi-Fi Direct groups simultaneously, however Wi-Fi Direct provides remarkable service discovery and security.

Chapter III outlines our proposed design of building an application to provide a secure mobile ad hoc network or MANET. We detail how we might overcome Android's limitations for building a Wifi mesh. We also provide a structure for how to automate and secure the mesh in a simple to use application that provides a DNS without centralized servers.

In Chapter IV, we present the limitations Android 6.0 (Marshmallow) API 23 impose on implementing a homogeneous Wifi mesh network of mobile devices. The first limitation arises from mobile devices containing a single Wifi antenna that restricts other devices on the mesh to a single Wifi Channel. Second, a Wi-Fi Direct GO AP introduces a significant drain on a mobile device's energy resources. Third, we found that the Android 6.0 API 23's Wi-Fi Direct implementation restricts devices from building multi-hop networks. Specifically, by blocking a device connected to two Wi-Fi Direct groups from opening IPv4 or IPv6 unicast sockets and Android also restricts a Wi-Fi Direct GO

from receiving or sending IPv4 or IPv6 multicast packets over the Wi-Fi Direct interface. These limitations make it impossible to build a secure homogenous Wifi mesh on Android 6.0 (Marshmallow) API 23 without gaining root permissions.

In the second half of Chapter IV, we considered two approaches for overcoming the limitations of using Android 6.0 for building a secure Wifi mesh. First, we explored how to enable custom functionality beyond what Android provides by gaining root permissions. While this technique is possible, it introduces significant concerns of sustainability outlined in Chapter IV. Second, we explored using helper devices equipped with multiple Wifi antennas for building a secure homogeneous Wifi mesh network. The helper network provides Wifi Access Points for mobile devices to join the mesh. To connect to the mesh network, the mobile device must authenticate itself with a helper node by encrypting the mesh's pre-shared key with the device's asymmetric key. To meet the NSA's security requirement all links in the mesh are encrypted using WPA2 and all devices connect to each other with a p2p VPN client. The p2p VPN client builds tunnels to other devices using dual asymmetric keys and a symmetric key to encrypt and share the random tunnel key. Devices on the mesh use a distributed DNS service embedded in the routing protocol to find each other. This technique shows promise in reducing the energy consumption on the end device while still meeting the NSA's security requirements for the mesh network. It also would allow non-Android devices to join the network.

A. KEY FINDINGS

Our research set out to discover if Android mobile devices could meet the needs of military personnel and first responders in an infrastructure-less environment by building a secure homogeneous Wifi mesh network. We discovered that Android 6.0 mobile devices cannot provide a homogeneous Wifi mesh network. We further examined means of overcoming Android's limitations in providing a Wifi mesh. This section contains a consolidated list of the key findings from this research.

- Mobile devices with a single Wifi antenna restrict a homogeneous Wifi mesh to a single Wifi channel. A single Wifi channel significantly restricts the useable bandwidth of the network when divided among closely located

devices. As a result, the number of simultaneous connections is severely limited.

- Wi-Fi Direct on Android 6.0 consumes significant energy when active.
- Wi-Fi Direct on Android 6.0 reuses the same IP subnet for every group. When associated to two Wi-Fi Direct groups, Android 6.0 cannot determine on which interface to open IPv4 sockets since both interfaces have the same IP subnet. Android devices associated to two groups cannot open unicast or broadcast sockets to other devices in either group.
- Android 6.0's Wi-Fi Direct interface blocks all IPv4 and IPv6 multicast traffic inbound and outbound. Unlike the standard Wifi interface, the Wi-Fi Direct interface's multicast block cannot be removed.
- Android 6.0 IPv6 unicast sockets do not support use of IPv6 SLAAC addresses.
- Android 6.0 cannot build homogeneous Wifi mesh networks without gaining root permissions and modifying the underlying Android processes.
- Helper devices equipped with multiple Wifi antennas provide a promising architecture for building a secure mesh network in infrastructure-less environments.
- Using a combination of pre-shared keys and asymmetric keys with a RADIUS server meets NSA requirements for unique Wifi authentication per device.
- Using a system of dual asymmetric keys and a symmetric key to build a VPN tunnel on top of a WPA2 encrypted Wifi satisfies the NSA's requirement for double encrypting Wifi links.

These results should help guide future work toward the goal of building secure Wifi mesh networks to operate in infrastructure-less environments. In the next section, we discuss promising research areas that may leverage our findings.

B. FUTURE WORK

This section provides a brief list of research questions that build on our research.

- How to securely manage asymmetric keys in an infrastructure-less environment? How to distribute a new public key and make sure it replicates to all desired devices and how to revoke compromised keys? Our research indicates distributed hash tables provide a promising means for distributing and tracking keys.

- How to implement a distributed p2p DNS server? Any infrastructure-less mesh network or MANET will need a p2p DNS service for finding other devices. Our research indicates that close integration with the routing protocol provide the most promising means for maintaining the p2p DNS records.

- How to build a p2p VPN? Our research proposed a method for building a p2p VPN network. How might a p2p VPN handle multicast and broadcast messages?

- What is the suitability of Wifi in contested environments? Contested environments may consist of an adversary actively jamming, direction finding, or attempting to discover information from the Wifi signals. Of these activities, the danger of direction finding may provide the largest danger since it allows an adversary to target friendly forces with kinetic weapons.

LIST OF REFERENCES

AmpliFi Wi-Fi. (n.d.). AmpliFi Wi-Fi: Explore. Retrieved April 14, 2017, from https://amplifi.com/explore.html

Bartock, M., Cichonski, J., & Franklin, J. (2015). *LTE security: How good is it?* Retrieved from National Institute of Standards & Technology website: http://csrc.nist.gov/news_events/cif_2015/research/day2_research_200-250.pdf.

Beachy, A., Gibson, J., & Singh, G. (2015). *C2 at the edge: Operating in a disconnected low-bandwidth environment*. (Masters Thesis). Retrieved from http://hdl.handle.net/10945/45812

Bluetooth Special Interest Group. (2014). *Specification of the Bluetooth system: Covered Core Package version 4.2*. *History* (Vol. 0). Retrieved from https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439

Camps-Mur, D., Garcia-Saavedra, A., & Serrano, P. (2013). Device-to-device communications with WiFi Direct: Overview and experimentation. *IEEE Wireless Communications*, *20*, 96–104. doi: 10.1109/MWC.2013.6549288

Chroboczek, J. (2011). *RFC 6126 - The Babel Routing Protocol*. University of Paris. Retrieved from https://tools.ietf.org/html/rfc6126

Cisco. (2015). Cisco Connected Mobile Experiences (CMX) CVD: Radio Frequency Operating and Data Rates [Digital Book]. Retrieved April 14, 2017, from http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Borderless_Networks/Unified_Access/CMX/CMX_RFOpFreqDataRates.html

Clausen, T., & Jacquet, P. (2003). Optimized Link State Routing Protocol (OLSR). *Ietf Rfc3626*, 75. doi: 10.1.1.11.620

Decuir, J. (2011). Bluetooth 4.0: Low energy [Lecture Slides]. Retrieved from https://californiaconsultants.org/wp-content/uploads/2014/05/CNSV-1205-Decuir.pdf

Department of Defense. (2009, Nov. 3). *Commerical wireless local-area network (WLAN) devices, systems, and technologies*. (DOD Instruction 8420.01). Washington, DC: Author.

Department of Defense. (2014). Contracts: Press Operations. Retrieved February 6, 2017, from https://www.defense.gov/News/Contracts/Contract-View/Article/606000

Dynastream. (n.d.-a). ANT+ security. Retrieved February 10, 2017, from https://www.thisisant.com/developer/resources/tech-faq/what-kind-of-security-does-ant-provide-1/

Dynastream. (n.d.-b). What is ANT+. Retrieved February 6, 2017, from https://www.thisisant.com/consumer/ant-101/what-is-ant/

Funai, C., Tapparello, C., & Heinzelman, W. (2017). Supporting multi-hop ad hoc networks through WiFi Direct multi-group networking. *2017 International Conference on Computing, Networking and Communications*. doi: 10.1109/ICCNC.2017.7876178

Google. (n.d.). Google WiFi: How it works. Retrieved April 13, 2017, from https://madeby.google.com/wifi/how-it-works/?utm_source=ads-en-ha-na-sem

Griffith, E. (2009). New "Wi-Fi Direct" spec revamps device networks. *PC Magazine*, Oct 14, 1–2. Retrieved from http://www.pcmag.com/article2/0,2817,2354201,00.asp

Hammond, M., Davis, D., & Gibson, J. (2015). *Examining requirements for the storage, utilization, and transmission of classified data in mobile devices*. (Master's Thesis). Retrieved from http://hdl.handle.net/

Harris Corporation. (n.d.-a). *Harris Falcon III AN/PRC-117G*. Retrieved February 10, 2017, from https://www.harris.com/sites/default/files/downloads/solutions/an-prc-117g-multiband-networking-manpack-radio-datasheet.pdf

Harris Corporation. (n.d.-b). *Harris Falcon III AN/PRC-152A*. Retrieved February 10, 2017, from https://www.harris.com/sites/default/files/downloads/solutions/harris-falcon-iii-an-prc-152a-wideband-networking-handheld-radio.pdf

Harris Corporation. (n.d.-c). Harris RF-335M-STC multi-channel handheld radio. Retrieved February 10, 2017, from https://www.harris.com/videos/harris-rf-335m-stc-multi-channel-handheld-radio

IEEE Computer Society. (2012). *802.11-2012 - IEEE Standard for information technology: Telecommunications and information exchange between systems local and metropolitan area networks*. (802.11-2012). doi: 10.1109/IEEESTD.2012.6178212

Johnson, D., Hu, Y., & Maltz, D. (2007). *The Dynamic Source Routing Protocol (DSR) for mobile ad hoc networks for IPv4*. (RFC 4728). doi: 10.17487/rfc4728

Khssibi, S., Idoudi, H., Van Den Bossche, A., Val, T., & Azzouz Saidane, L. (2013). Presentation and analysis of a new technology for low-power wireless sensor network. *International Journal of Digital Information and Wireless Communications (IJDIWC)*, *3*, 81–92. Retrieved from http://sdiwc.net/digital-library/presentation-and-analysis-of-a-new-technology-for-lowpower-wireless-sensor-network

Lackner, G. (2013). A comparison of security in wireless network standards with a focus on bluetooth, WiFi and WiMAX. *International Journal of Network Security*, *15(6)*, 420–436. Retrieved from https://pdfs.semanticscholar.org/93dc/7c3ad36eb2e258d422185cc1593e31005570.pdf

Monarrez, A., Singh, G., & Buettner, R. (2015a). *Extending Wi-Fi Direct for automated operations*. (Master Thesis). Retrieved from http://hdl.handle.net/10945/45229

Monarrez, A., Singh, G., & Buettner, R. (2015b). Supporting first-responders in infrastructure-less environments. *Procedia Engineering*, *107*, 34–40. doi: 10.1016/j.proeng.2015.06.056

Narra, H., Cheng, Y., Çetinkaya, E., Rohrer, J., & Sterbenz, J. (2011). Destination-Sequenced Distance Vector (DSDV) routing protocol implementation in ns-3. *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, 439–446. doi: 10.4108/icst.simutools.2011.245588

National Security Agency. (2003). *National information assurance glossary*. (CNSS Instruction 4009). Washington, DC: Author.

National Security Agency. (2013). Mobility Capability Package. Retrieved from http://www.nsa.gov/ia/programs/mobility_program/index.shtml

National Security Agency. (2015). Commercial National Security Algorithm (CNSA) Suite. Retrieved February 10, 2017, from https://www.iad.gov/iad/programs/iad-initiatives/cnsa-suite.cfm

National Security Agency. (2016a). Campus WLAN Capability Package. Retrieved from https://www.nsa.gov/resources/everyone/csfc/capability-packages/assets/files/campus-wlan-cp.pdf

National Security Agency. (2016b). Changes to CNSA Suite and Quantum Computing Policy. Retrieved February 4, 2016, from https://www.iad.gov/iad/news/changes-to-cnsa-suit-and-quantum-computing-policy.cfm

National Security Agency. (2016c). Commercial Solutions for Classified Program (CSfC). Retrieved February 3, 2017, from https://www.nsa.gov/resources/everyone/csfc/

National Security Agency. (2016d). CSfC: Capability Packages. Retrieved February 3, 2017, from https://www.nsa.gov/resources/everyone/csfc/capability-packages/

Parkvall, S., Dahlman, E., Furuskär, A., Jading, Y., Olsson, M., Wänstedt, S., & Zangi, K. (2008). LTE-Advanced: Evolving LTE towards IMT-Advanced. In *2008 IEEE 68th Vehicular Technology Conference*. doi: 10.1109/VETECF.2008.313

Perkins, C., Belding-Royer, E., & Das, S. (2003). RFC 3561 AODV. *Network Working Group*. Retrieved from https://www.ietf.org/rfc/rfc3561.txt

Perkins, C. E., Bhagwat, P., E. Perkins, C., Bhagwat, P., Perkins, C. E., & Bhagwat, P. (1994). Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. *Proceedings of the ACM SIGCOMM94, London, UK*, *24*, 234–244. doi: 10.1145/190809.190336

Peterson, L. L., & Davie, B. S. (2011). *In praise of computer networks: A systems approach*. (R. Adams & N. McFadden, Eds.) (5th ed.). Burlington, MA: Morgan Kaufmann. doi: 10.1016/B978-008043924-2/50055-9

Plume Wifi. (n.d.). Plume: How adaptive WiFi works. Retrieved April 13, 2017, from https://www.plumewifi.com/how-it-works

Qualcomm. (n.d.). LTE Direct device-to-device discovery technology. Retrieved January 29, 2017, from https://www.qualcomm.com/invention/research/projects/lte-direct

Samsung Corporation. (n.d.). Samsung Galaxy S6 Edge Plus specifications. Retrieved April 10, 2017, from http://www.samsung.com/global/galaxy/galaxy-s6-edge-plus/%5Cnhttp://www.samsung.com/global/galaxy/galaxy-s6-edge-plus/#!/spec

Soares, E., Brandão, P., Prior, R., & Aguiar, A. (2017). Experimentation with MANETs of smartphones. *CoRR*, *abs/1702.0*. doi: 10.1109/WD.2017.7918133

The H. Open. (2011). Wi-Fi Protected Setup made easier to brute force. Retrieved January 29, 2017, from http://h-online.com/-1401822

Wakelin, R. (2014). 802.11s as link layer in libre-mesh. Retrieved January 29, 2017, from https://www.mailmanlist.net/pipermail/devel/2014-September/003467.html

Wi-Fi Alliance. (n.d.). Wi-Fi Direct: Discover Wi-Fi Direct. Retrieved January 29, 2017, from http://www.wi-fi.org/discover-wi-fi/wi-fi-direct

Wi-Fi Alliance. (2014). Wi-Fi certified Wi-Fi Direct: Personal, portable Wi-Fi technology. Retrieved from http://www.wi-fi.org/download.php?file=/sites/default/files/private/wp_Wi-Fi_CERTIFIED_Wi-Fi_Direct_Industry_20140922_0.pdf

Wirtz, H., Heer, T., Backhaus, R., & Wehrle, K. (2011). Establishing mobile ad-hoc networks in 802.11 infrastructure mode. *CHANTS '11 Proceedings of the 6th ACM Workshop on Challenged Networks*, 49–51. doi: 10.1145/2030652.2030666

Zahn, T., & Schiller, J. (2005). MADPastry: A DHT Substrate for Practicably sized MANETs. In *Proc. of 5th Workshop on Applications and Services in Wireless Networks (ASWN2005)*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=D8898951731F8BAEE7CDC2B421AD8EC6?doi=10.1.1.710.9128&rep=rep1&type=pdf

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California