Off Grid communications with Android

Meshing the mobile world

Josh Thomas
Accuvant LABS Research Consulting Group
Bedford, MA USA
jthomas@accuvant.com

Jeff Robble
The MITRE Corporation
Bedford, MA USA
jrobble@mitre.org

Nick Modly
The MITRE Corporation
McLean, VA USA
nmodly@mitre.org

The SPAN project is an open source implementation of a generalized Mobile Ad-Hoc Network framework. The project's goals are to bring dynamic mesh networking to smart phones and to explore the concepts of Off-Grid communications.

Keywords: MANET; OLSRd; Mesh Network; Android; Decentralized; Peer to Peer; Smartphone

I. Introduction to SPAN

Recent worldwide events have showcased that our current communications infrastructure is not as reliable as we would like to believe. Cellular towers can be destroyed by natural phenomena or simply overloaded beyond capacity and Wi-Fi hotspots are reliant on power and network connectivity, two things in short availability during a catastrophic event. We've seen these issues surface time and time, from Katrina to Haiti to Fukushima. It's always the same problem: no connectivity and no communication.

The SPAN project [1] (Smart Phone Ad-Hoc Networks) attempts to ameliorate these issues by providing an alternate means for information dispersal. The project utilizes MANET (Mobile Ad-Hoc Network) technology to provide a resilient backup framework for communication between individuals when all other infrastructure is unavailable or unreliable. The MANET based solution is a headless, infrastructure-less network that allows common smart phones to link together in a dynamic way. As such, the SPAN project is harnessing the ubiquity of smart phones to provide durable communications in times of need.

The MITRE initiated SPAN project has created a fully open source framework for implementing and exploring MANET networks and routing algorithms. The framework provides a full "proof of concept" implementation of a functional MANET for Android based smart phones. The project also provides a "plug and play" framework for developing and testing custom Ad-Hoc routing protocols. The routing protocols are the true cornerstone of the MANET architecture as they adapt the network for scalability, mobility and power constraints of mobile devices. The SPAN team is currently working on an adaptive routing protocol that will dynamically adjust itself based on the current runtime metrics of the mesh network itself.

Aside from resilient information sharing, the SPAN project also allows for "Off Grid" communications. There are times when transferring and sharing data is necessary but due to concerns over security, the possibility of monitoring or for other reasons participants may not wish to utilize either the Internet or existing cellular networks.

II. A Working Scenario

Consider the recent incident in Fukushima as an exemplar for the need of a ubiquitous mobile mesh network. During and soon after the disaster, world media reported numerous stories regarding the lack of connectivity and viable communications channels for rescue and emergency response personnel. Viral YouTube videos were abundant on the Internet, showcasing families attempting to stay in touch and informed only to be met with a "no service" message from their phones. In some areas the cellular infrastructure had in fact been destroyed, but a larger swath of the problems could be attributed to simple cell tower overloading. In general, most cellular service providers outside of major metropolitan areas build infrastructure and plan for roughly an 80% capacity load. There is no real return on investment seen for a more expensive and redundant capacity capability and this is generally a sound business practice until an unpredictable disaster strikes.

If a ubiquitous mobile mesh network had been in place as a backup communications platform for the affected area, the situation could have been drastically different.

Families could have easily stayed in communication using text messaging and VoIP (Voice over IP) phone calls. This movement of civilian communications off of the fragile cellular infrastructure would have allowed the system to continue functioning for the needs of the emergency disaster and response teams. Such a backup plan could easily have helped remove some of the unneeded chaos that surrounded this crisis.

In a general sense, if a sufficiently high number of smart phones in a geographic region happen to be running or are capable of running a mesh-networking program in the background then the mesh itself becomes a stable communications platform. Infrastructure for simplistic communications becomes unnecessary and irrelevant because each individual phone is simply a node in a large connected graph.

While this research has solely focused on providing backup communications during times of crisis, this is not an inherent limit of the technological solution to stop there. MANET technology can easily be used to replace geographically co-located cellular frameworks and to bring basic Internet connectivity to regions where no such connectivity exists, or, as would have been applicable during some recent Arab Spring situations, restore connectivity where it has been removed.

III. TECHNICAL DETAILS

A. *Leveraging Open Source Projects*

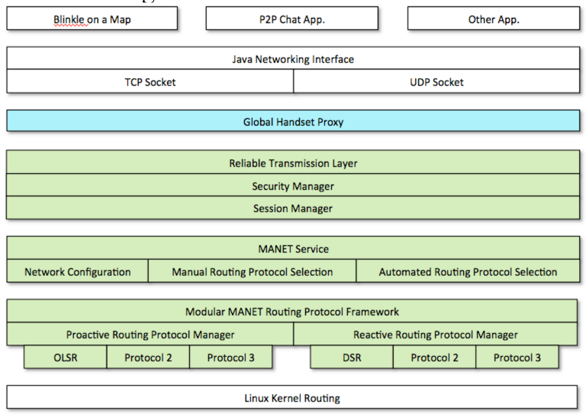
The SPAN project was initially based on the Wireless Tether for Root Users [2] application written and maintained by Harald Mueller. The application originally started out as an open source project licensed under GPLv3 but eventually became closed source to prevent profiteers from rebranding and selling it on the Android Market (now Google Play) for personal gain. The project leverages much of Mueller's interface design and follows his method for configuring a wireless chip to operate in ad-hoc mode using the iwconfig Linux command line utility as often as possible.

The SPAN project is also somewhat reminiscent of the existing Serval / BatDroid [3] project, which provides a simple management wrapper to start/stop the BATMAN daemon on a rooted Android handset. The two main differentiating factors between the SPAN project and the Serval project is the former allows for arbitrary routing protocols to be used during MANET runtime and that software need not be customized to harness the mesh technology. This is accomplished by harnessing a generalized architecture implemented as a framework instead of a simple proprietary implementation of a specific protocol.

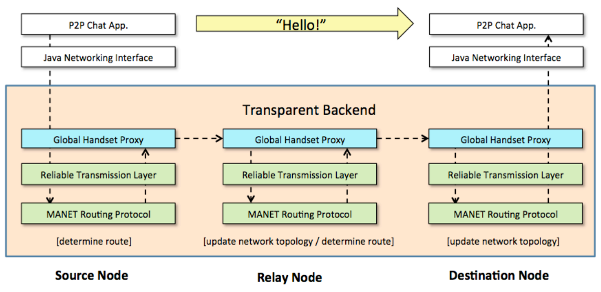
B. *Architecture*

The SPAN architecture is intentionally designed to allow for arbitrary routing protocol use during runtime. This generalized solution will allow for custom routing protocols to be developed and investigated without incurring the overhead of building a complete implementation. The intent is to provide a MANET test bed for protocol developers to experiment with real world behaviors and adapt the implementation as required. Such a framework also allows for the SPAN adaptive framework discussed later in this paper.

The framework is injected into the existing Android network stack between OSI layers 2 & 3. Given this network stack is based on the standard BSD implementation, our framework is inherently portable across most platforms, mobile or otherwise. The initial implementation of the framework is designed as such:


Blinkle on a Map
P2P Chat App.
Other App.
Java Networking Interface
TCP Socket
UDP Socket
Global Handset Proxy
Reliable Transmission Layer
Security Manager
Session Manager
MANET Service
Network Configuration
Manual Routing Protocol Selection
Automated Routing Protocol Selection
Modular MANET Routing Protocol Framework
Proactive Routing Protocol Manager
Reactive Routing Protocol Manager
OLSR
Protocol 2
Protocol 3
DSR
Protocol 2
Protocol 3
Linux Kernel Routing

The injection point of the Global Handset proxy allows the SPAN framework to control all network traffic seamlessly. From the OS and application layer viewpoint the MANET is simply another avenue for network access and its existence is completely hidden. Once the framework is running on the device, no software application or Android OS modifications are necessary. This transparent nature allows for common applications like Twitter, Facebook and VoIP to simply work, assuming a single node on the overall MANET has bridged the mesh to the outside Internet.



C. *Ad-Hoc Mode on Android Hardware*

The Android framework is primarily designed to configure the built-in wireless chip to operate in managed mode (a.k.a. infrastructure mode) and monitor the state of the managed network. In other words, the default behavior of the wireless chip in an Android device is to authenticate with an external access point and act as a client to connect to a pre-existing wireless network. The user can control various options for connecting to a managed network through the wireless and network preferences available through the Settings app. Many Android implementations store network information in a wpa_supplicant.conf file and perform authentication using the wpa_supplicant command line utility, which is the standard Linux approach for connecting to a managed network.

An ad-hoc network does not consist of static access points and does away with the need for dedicated devices for managing the network. Instead, each device in an ad-hoc network is capable of intelligently routing packets to other peers in the network. In order to be successful, each device must know about the network topology prior to planning routes (i.e. proactive routing) or capable of learning the network topology on demand to plan a route at the time of packet transmission (i.e. reactive routing). Both approaches have pros and cons, as we will discuss in a later section.

The pre-ICS (Ice Cream Sandwich / 4.0) Android framework does not support configuring the built-in wireless chip to operate in any other mode but managed mode. ICS offers support for Wi-Fi Direct, but the ICS implementation of the Wi-Fi Direct specification does not provide a complete ad-hoc network solution, as we will discuss in a later section. In order to configure the wireless chip in ad-hoc mode we dive deeper than the Android framework and work with the wireless chip drivers directly by using the iwconfig Linux command line utility to set the parameters of the wireless interface. In order to use iwconfig the Linux kernel must have support for the Wireless Extensions API. The following table shows which of the devices we used for development have support for the Wireless Extensions API out of the box and which do not:

Wireless Extensions Support	**No Wireless Extensions Support**
Samsung Nexus S 4G	Samsung Galaxy Nexus
Samsung Galaxy Tab 10.1	ASUS Eee Pad Transformer Prime
Samsung Galaxy S II Epic Touch 4G	Motorola Razr Maxx

Note that all of the devices that support the Wireless Extensions API use the Broadcom wireless chip, as shown in the following table:

Device	**Wireless Chip**
Samsung Nexus S 4G	Broadcom BCM4329
Samsung Galaxy Tab 10.1	Broadcom BCM4330
Samsung Galaxy S II Epic Touch 4G	Broadcom BCM4330
Samsung Galaxy Nexus	Broadcom BCM4329
ASUS Eee Pad Transformer Prime	AzureWave AW-NH615 (rebranded Broadcom BCM4329)
Motorola Razr Maxx	Texas Instruments WL1285C
iPhone 4S	Broadcom BCM4330
Nokia Lumia 900	Broadcom BCM4329

In addition to the devices that support the Wireless Extensions API, we were able to compile support into the Linux kernel for the Samsung Galaxy Nexus and ASUS Eee Pad Transformer Prime. We are very thankful that both Samsung and ASUS have provided their kernel source code to the open source community.

Thus, we have had great success with the Broadcom BCM4329 and BCM4330 wireless chipsets in Android devices and strongly believe that it is possible to use the Wireless Extensions API to configure the same wireless chipsets in the iPhone 4S and Nokia Lumia to operate in ad-hoc mode. On the other hand, we have had limited success configuring the Motorola Razr Maxx TI wireless chip to operate in ad-hoc mode using the tiwlan drivers. Motorola has made pieces of the kernel available to the open source community so it may be possible to compile Wireless Extensions support into the Motorola Razr Maxx kernel.

D. Gateway

There are many reasons why devices in the ad-hoc network may need to reach out to devices on managed network. For example, many useful apps are based on the client-server model and require access to a server hosted on a managed network. In order to "bridge" the ad-hoc network and a managed network a gateway device must be appointed. The gateway device must have one network adapter configured to operate in ad-hoc mode and another network adapter configured to operate in managed mode. Then packets can be forwarded across those two adapters.

Specifically, we use the ASUS Eee Pad Transformer Prime for our primary gateway device. By compiling rtl8187 USB driver support into the kernel we are able to use an ALFA AWUS036H wireless USB adapter as a second network interface. We configure the Eee Pad's internal wireless adapter to operate in managed mode and the ALFA to operate in ad-hoc mode. We then use the iptables command line utility to allow the Eee Pad to masquerade as devices on the ad-hoc network and forward packets across the adapters. Thus, the Eee Pad effectively performs Network Address Translation between the ad-hoc subnet and the managed network.

We leverage the behavior of the Settings app and Android framework for configuring the Eee Pad's internal wireless adapter to operate in managed mode. The user can connect the device to an access point and the device will remain connected to the access point regardless if the ALFA is enabled to operate in ad-hoc mode or not. On the other hand, a non-gateway device will disconnect from a managed network when its internal wireless adapter is configured to operate in ad-hoc mode.

Additionally, we have used both the Samsung Galaxy S II Epic Touch 4G and Samsung Galaxy Nexus as gateway devices by forwarding packets between their internal wireless adapter configured to operate in ad-hoc mode and their internal 3G/4G adapter. This allows every other device in the ad-hoc network to access the Internet through the cell service of those devices. Note that many cell service providers do not condone "tethering" of this nature because many cell phone users use it as a way to share one service plan across multiple devices instead of paying for individual service plans.

Devices in the ad-hoc network can successfully browse the Internet through the gateway device; however, we have observed that on most devices the Browser app will prompt the user with a dialog stating that no network is available, although after dismissing the dialog the webpage will load without a problem. This is evidence that our approach to setting up the ad-hoc network works at a lower level then the Android framework, which does not recognize the device has a valid Internet connection because the wireless chip is not operating in managed mode.

IV. Field Test Results

A. *Effiective Range*

The initial field tests of the SPAN framework utilized both the OLSRd protocol and a simple implementation of the Dijkstra algorithm for packet routing, and the tests were preformed using an array of currently supported devices. It was observed that each MANET node utilizing a Broadcom BCM4329 Wi-Fi chipset could be a maximal distance of 106 feet (32 meters) from its closest neighbor and still maintain MANET connectivity. For devices harnessing the Broadcom BCM4330 chipset, the maximal distance was observed to be 98 feet (29 meters). Our initial conclusions concerning the range delta on these devices points towards differing waveforms due to channel hopping and power utilization and configuration.

B. *Upper Limits of Simple Multi-Hop Routing*

The initial testing did not reveal an upper limit on multi-hop communications, allowing a simple chat conversation to traverse a 5 hop network with minimal delay and throughput problems. The SPAN team intends to explore networks of 10 to 25 node traversals later this year and to provide a more comprehensive and exhaustive network analysis. The team expects to discover a maximal limit to multi-hop routing of VoIP data in the range of 10-12 node traversals.

C. *Node Density Limitations*

Given the channel-based nature of the 802.11 specifications, the SPAN team expects to discover an upper limit of devices that can exist in the same peer-to-peer MANET enclave. This limit was not reached during our initial test of 30 devices. The team expects to solve the maximum channel utilization limit by creating clusters or enclaves of proximal devices to allow for a scalable network beyond the typical bounds of the 802.11 specification.

V. Routing

The single most challenging aspect of implementing a robust and scalable mesh network is the design of the routing protocol. Without centralized servers and standard networking infrastructure to generate optimal paths across the network, the nodes of the mesh themselves must determine how to deliver the data in an efficient manner. The field thus far can be subdivided into two distinct approaches: Proactive and Reactive. Though neither approach can change raw bandwidth both solutions can have a large impact on network throughput.

A. *Proactive Routing*

The proactive approach (and its exemplar OLSRd) attempts to mimic standard networking paradigms to predetermine routes and store them prior to use or need. In essence, the algorithm floods the mesh network with hello messages in order to determine topology and routing data. The routes are then stored per device for a specified time and recreated once the temporal bound has expired. While this approach ensures the network is responsive to packet transfers at runtime, functionality is provided with a high cost. The proactive paradigm can easily saturate the mesh network with route discovery packets, building possibly unused and unneeded routes at the cost of actual data transfer. In addendum, the highly mobile nature of mesh networks can alter the physical topology prior to the expiration of the stored routes. This issue forces the protocol to generate new routes dynamically after the stored paths have been discarded.

B. *Reactive Routing*

Reactive protocols await an actual need for a network traversal path prior to exploring the mesh for a route. This ensures the network remains uncluttered with possibly unnecessary hello packets. The inherent downside to this approach is a sluggish behavior visible to the end user when trying to utilize any new node on the network. Given the lack of exploratory traffic, pure reactive networks also have known issues with determining exactly what nodes are available for potential use. This problem becomes apparent when you consider issues with DHCP or other network identification mechanisms.

C. *New Routing Paradigms*

BATMAN (The Better Approach To Mobile Adhoc Networking) is a routing protocol currently under development by the Freifunk Community and is intended to replace OLSRd. BATMAN's main differentiating design aspect is the concept of route knowledge decentralization. The paradigm attempts to ensure no single node needlessly collects all the routing data in the network. Instead each individual node only saves information about the "direction" it received data from prior to packet forwarding. As the data gets passed on from node to node around the mesh, packets get individual dynamically created routes based on current network topology. In essence, a network of collective routing intelligence is created and dynamically harnessed at runtime.

D. Sensory Intelligence

In future versions of the SPAN framework, the team will provide reference implementations for routing protocols based on smart phone sensor data. The team expects vast improvements in mesh network stability and speed when harnessing location, speed and vector of movement information into the packet headers of exploratory packets. Nodes will be cognizant of neighbor node mobility when selecting potential routes.

Aside from movement-based information, the SPAN team will explore battery and power consumption leveling across the mesh in the near future. In this paradigm, the routing protocol will prioritize next hop nodes based on available battery level and charging state of the device.

E. Self Evolving Algorithms

In the coming year, the SPAN team will explore an automated adaptive routing protocol. The protocol will perform self-analysis during runtime and adjust the routing fingerprint based on current use of the network. Simply put, the protocol will attempt to automatically adjust battery leveling, network throughput and bandwidth based on how the network itself is being utilized by the participants at any given time. An optimal solution for a sparsely populated network attempting to pass VoIP packets will be drastically different than the solution for a highly dense, large network passing simple text data

VI. SECURITY

While far from a complete solution, the SPAN team has generated a simplistic design for mesh network security. Each node on the mesh will have a shared key for initial network exploration. This key will be either prepackaged into the mesh client or transferred to the device by Bluetooth / NFC when joining the network. Once the node has joined the network, it will share its own public key with any node requesting communication. Once keys have been transferred, the network will harness the standard encryption scheme for secure client /server based communications. The network will also support the expected collection of VPN tunnels, WEP & WPA.

Apart from data protection, the SPAN team is cognizant of DDOS issues with the OLSRd protocol. Given the protocol itself can saturate the network with hello packets during normal operation, it is not beyond comprehension that a malicious attack could do the same. Our initial modifications to the OLSRd protocol should, at a minimum, limit such disturbances to a localized enclave of the mesh.

VII. CONCLUSIONS AND FUTURE WORK

The SPAN team expects to continue refining the framework and developing routing protocols in the near term. We expect to harden our security posture both for network and data protection.

Please contact the SPAN team if you have any questions, comments or concerns. Also, please contact us if you use the framework and have interesting stories to tell.

VIII. LINKS AND RESOURCES

[1] The SPAN Project. https://github.com/monk-dot

[2] Android WiFi Tether. http://code.google.com/p/android-wifi-tether/

[3] The Serval Project. http://www.servalproject.org

[4] Initial SPAN interview. http://www.youtube.com/embed/RrI3MUnExJM

[5] DefCon 20 Presentation materials: http://defcon.org/html/defcon-20/dc-20-speakers.html#Thomas

[6] DerbyCon 2.0 Presentation Materials: https://github.com/monk-dot

[7] Current version of this paper https://github.com/monk-dot