Implementing a software-defined networking structure on an ad hoc network of Android devices.

Brandon Guttersohn
Undergraduate Researcher
Southeast Missouri State University
bguttersohn@gmail.com

Paul Baskett
Graduate Student Mentor
University of Missouri
pkbkbc@mail.missouri.edu

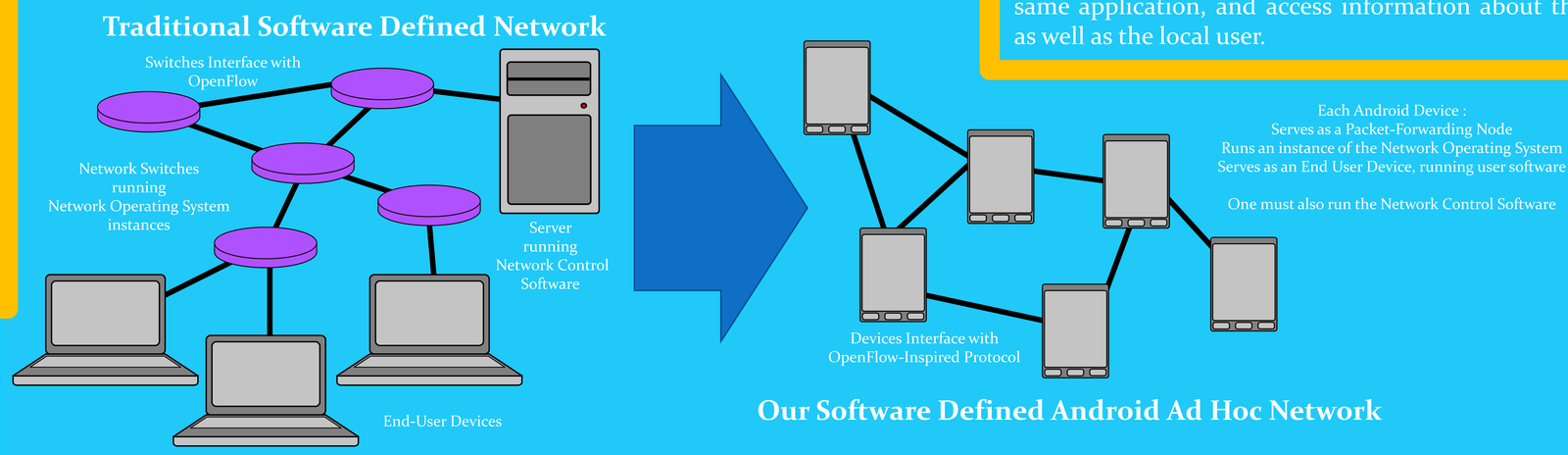
Yi Shang
Faculty Mentor
University of Missouri
shangy@missouri.edu

Goal

Contemporary network schemes are built around the use of proprietary networking devices interacting with general autonomy in the area of sending and forwarding network data. This paradigm presents a handful of issues, including a lack of transparency in data logistics, and a lack of centralized, intelligent, network control. Because each of these devices is controlled by the closed-source firmware of their manufacturers, researchers and network administrators are unable to implement experimental routing rules. However, using software abstractions and management applications, the network can be controlled centrally, and treated as a whole. This is the concept of software defined networking. With this concept as our inspiration, we seek to design a network management structure for an ad hoc Wi-Fi network of Android smart phones, and implement code demonstrating this structure in action. The structure should simplify network management, as well as 3rd party application access to network functions.

Design

There exist numerous design challenges in applying the concept of software defined networking to an ad hoc network of Android devices. One particular issue is that the ad hoc network is composed of predominately homogenous nodes – Android smartphones. There is no dedicated location for a server, and each node must be capable of running all layers of the network abstraction.



Results

Our resulting design is very flexible, and should be conducive to further implementation of software defined networking concepts on Android ad hoc networks. The current implementation has been successfully able to generate a network map, modify network routing tables, and interface 3rd party applications. The library we created for interfacing 3rd party applications is simple, and allows data to be sent over the network with only a few lines of code. Because all 3rd party application data transmission is managed by virtual subnetworks, developers can easily find peers running the same application, and access information about those users, as well as the local user.

Software Abstractions

Packet Forwarding Layer

The packet-forwarding layer is primarily charged with wrapping the device's networking hardware, and obeying the rules it is given with regard to packet forwarding. It is designed to adhere to a standard protocol, similar to OpenFlow, which allows the network control software, via the network operating system, to modify the packet forwarding tables of various network nodes. While the complex decisions and controls are reserved for higher level abstractions, the packet forwarding software is in charge of ensuring that data is sent to its ultimate destination.

Network Operating System

The network operating system is the central hub of the entire control structure. Running on each device, it sits between the packet forwarding layer, the control software, and 3rd party applications. Its responsibilities include:

- Monitoring the state of the network
- Creating a map of the network, for detailed routing decisions
- Providing an interface for 3rd party applications
- Maintaining virtual subnetworks
- Supplying network information to the control software and to 3rd party applications
- Possibly, it could be extended to automatically reroute data flows to bypass network congestion.

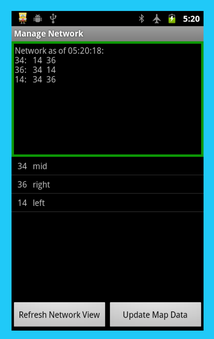
Control Software

The control software is the brain of the network. This software is trusted to design the network's routing algorithm, and create specific routing rules. The control software can accept specific rules from the end user, or create the entire routing scheme automatically. Because this software communicates with the network operating system over a well-defined protocol implemented with the Android Interface Definition Language, developers can implement their own control applications. Our implementation focused on network testing, exhibition, and direct user control.

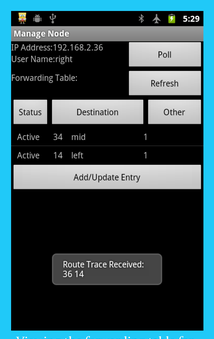
3rd Party Applications using the Network

3rd party applications interface the software-defined control structure using a library we created. Once an application connects to the network, it joins a virtual network, managed by the network operating system. This virtual network is reserved for devices running the specific 3rd party application, and makes it very easy for developers to find nearby users. To use the library, developers need only create a network connection object with a virtual network name, application context, and some implementation of the callback interface to handle received data. They can then query the object for network information, and transmit data.

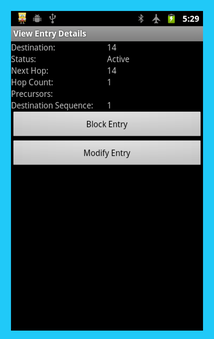
Launching the software system via the control application



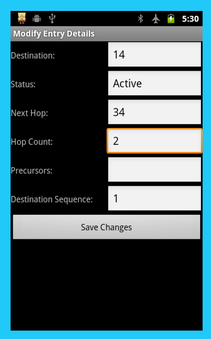
Viewing the network map, and a list of network device names.



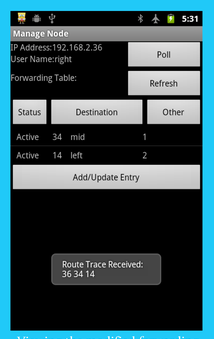
Viewing the forwarding table for a network device, while receiving a route tracing packet



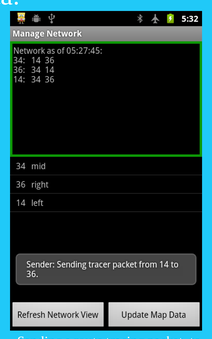
Viewing the details of a forwarding table entry.



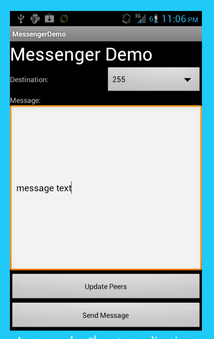
Modifying a forwarding table entry



Viewing the modified forwarding table, and receiving an affirmative route tracing packet



Sending a route tracing packet, to test whether set rules are being adhered to.



An example 3rd party application, which sends and receives text messages.