SDNAN: Software-Defined Networking in Ad hoc Networks of Smartphones

Paul Baskett, Yi Shang, and Wenjun Zeng
Computer Science Department
University of Missouri
Columbia, Missouri, USA

Brandon Guttersohn
Computer Science Department
Southeast Missouri State University
Cape Girardeau, Missouri, USA

Abstract — In this paper, SDNAN, a first attempt to implement software-defined networking (SDN) over a wireless ad hoc network of smartphones, is presented. Its modular ad hoc network management structure can be easily modified and extended. Its abstractions and interfaces allow components to communicate without knowing how other components work. Third-party applications can use the interfaces to access the ad hoc network, significantly reducing development time and program complexity. A prototype system has been implemented on Android smartphones over Wi-Fi and achieved good preliminary results.

I. INTRODUCTION

Software defined networking (SDN), a new approach for the next generation Internet, is proposed to make networks more adaptable and easily configurable to meet the changing demands and operating environments [1,2]. Open standards, such as OpenFlow, transforms networking architecture and turns individual network elements into programmable entities [3,4]. By decoupling the control and data planes, network intelligence and state are logically centralized and the underlying network infrastructure is abstracted from the applications. As a result, highly adaptive, flexible, and scalable networks can be built.

In the principles of SDN, a network is separated into three distinct components: communication layer, network operating system (NOS), and control program. The communication layer consists of the physical network devices like routers and switches; the NOS manages network resources; and the control program controls the network through the NOS.

Adapting SDN to ad hoc networks could improve the performance and usage of ad hoc networks and facilitate application development. The SDNAN (Software-Defined Networking in Ad hoc Networks) system is a first attempt to do that. It has three layers: a) an ad hoc networking layer based on AODV [5], b) an NOS layer that maintains a global map of the network, manages sub-networks for each application, and supports dynamic change of routing protocols, and c) a control program that controls forwarding rules, routing tables, and routing protocol on the fly. The system enables virtualization of a physical ad hoc network through virtual network slices, which provides a convenient way to implement application-specific routing rules and resource management. A prototype has been implemented and tested in ad hoc networks of Android smartphones.

II. SDNAN DESIGN AND IMPLEMENTATION

This section presents the design and implementation of the major components of SDNAN.

A. Ad Hoc Networking Layer

While traditional SDN uses network routers with proprietary routing protocols, SDNAN relies on ad hoc networking services written in the application layer on each node. In the smartphones prototype, AODV over Wi-Fi is implemented as the underlining networking service [5]. Each node tracks its neighbors through the use of periodic Hello packets. When a route is desired, a route request (RREQ) is broadcast throughout the network. When a route to the destination is found, a route reply (RREP) is sent back to the initiating node, updating routes to both source and destination for intermediate nodes. When a node loses a neighbor a route error (RERR) is sent to nodes that rely on the node's connection with the lost neighbor for routes.

Android Interface Definition Language (AIDL) is used for inter-process communication between the three SDNAN layers [6]. The following table shows some of the methods implemented for the ad hoc networking interface.

	IAdhocService Interface
Return Type	**Method**
void	startAdHocNetwork()
Map	getNetworkMap()
void	pollDeviceForwardingTable(int node)
List	getForwardingTable(int node)
void	addFlowTableEntry(int node, RouteTableEntry entry)
void	removeFlowTableEntry(int node, RouteTableEntry entry)

B. NOS

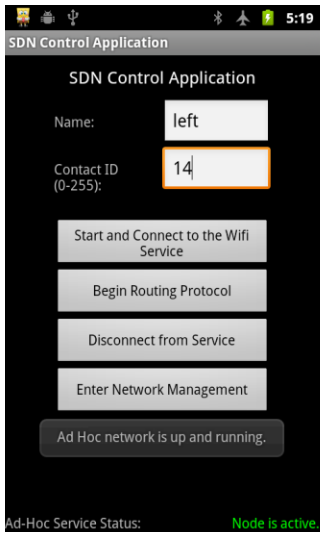
The NOS sends/receives application packets and maintains a network map and network slicing. It scans the network periodically to generate a network map. The node initiating the request sends a map request packet to all single-hop peers, including packet data unit type, a randomly generated integer id, the node address of original requester, a blacklist of already visited nodes, and a global view table. The first node labels itself as the original requester, adds itself to the blacklist, and adds a series of pairs to the global view table. Each pair contains the original node itself and one of its single-hop peers. When a peer receives the packet, it adds pairs representing its single-hop peers, blacklists itself, and continues forwarding to its peers that are not already blacklisted. The original node collects responses from other nodes to build an adjacency list, representing the whole network.

A logical network refers to our implementation of network slicing. Each logical network running over the physical ad hoc network is distinguished primarily by its logical network tag. Each user application could run in its own logical network, making it easy to implement application specific routing rules and better manage network resources across applications. Our logical network implementation involves the use of a logical network manager in the NOS, which maintains a set of logical network objects. All application data sent through the network must be done via a logical network object, which currently defines a network tag and a set of node members.

C. Control Program

A simple control program is implemented in SDNAN, which communicates with NOS using an INetworkOS AIDL interface. The control program first initializes and configures the ad hoc network. Its network management portion allows the user to view the network map and to update it by changing routing rules. It can display a node's forwarding table, where routing rules can be added or removed. Routing rule modifications are carried out by configuration request packets.

In the following screen shots, the left one shows the launch of the SDNAN system via the control program and the right one shows a view of the forwarding table of a phone, while receiving a route tracing packet.

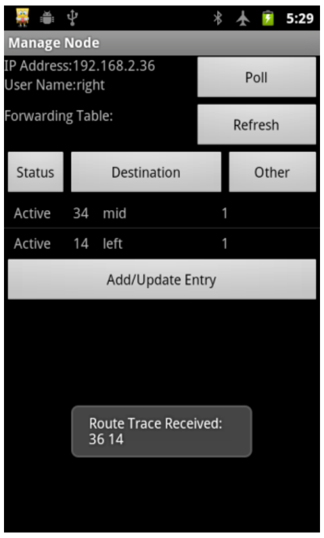


Name: left

Contact ID (0-255): 14

Ad Hoc network is up and running.

Ad-Hoc Service Status: Node is active.



IP Address:192.168.2.36
User Name:right
Forwarding Table:

Status	Destination		Other
Active	34	mid	1
Active	14	left	1

Route Trace Received:
36 14

D. Developing User Applications over SDNAN

SDNAN interface to user applications is implemented in two ways, using Android Intent system [7] and AIDL [6], respectively. The intent-based system requires third-party developers to implement an Android BroadcastReceiver, which listens for intents broadcast by the Android operating system. Data sent in an Android intent is provided as a key-value pair, so application developers need to know the keys used for the various Intents used by the NOS.

In the AIDL-based implementation, a small library was developed to simplify the use of AIDL for third-party developers. The library sets up an AIDL connection to the NOS and allows the application to view and communicate with other nodes in the network running the same application.

III. Experimental Results

Two applications were implemented, text messaging and file transferring, over SDNAN for demonstration and testing purpose. For a 3-node network setup with one phone sending 100 text messages to another phone through one intermediate phone, three implementations were tested: Intents-based SDNAN, AIDL-based SDNAN, and a native Android implementation, in which a text messaging application is built into AODV ad hoc networking that is completely independent of the NOS and control program. The three implementations have significant speed differences. The native, Intents-based, and AIDL-based implementations averaged 733 ms, 2600 ms, and 1600 ms, respectively.

A major benefit of SDNAN is that its clean interfaces between its three layers and the user applications makes it much easier to improve the functionality and performance of each component and to develop user applications on ad hoc networks. The text messaging app takes less than 20 lines of code to use the AIDL interface library and less than 50 lines to use the intent interface, whereas implementing similar functionality directly using Wi-Fi ad hoc networking, even with a library to simplify its use, can easily take hundreds lines of code.

References

[1] A. Feldmann, "Internet clean-slate design: what and why?" *ACM SIGCOMM Computer Communications Review (CCR)*, 37(3), pages 59-64, July 2007.

[2] Scott Shenker, "The Future of Networking, and the Past of Protocols", http://www.youtube.com/watch?v=WVs7Pc99S7w, 2011.

[3] Open Networking Foundation. "Software-Defined Networking: The New Norm for Networks," https://www.opennetworking.org/, 2012.

[4] The OpenFlow Switch Consortium, "OpenFlow Switch Specification Version 1.1.0," http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf, 2011.

[5] C. Perkins, E. Belding-Royer, S. Das, "Ad Hoc On Demand Distance Vector (AODV) Routing," http://tools.ietf.org/html/rfc3561, 2003.

[6] Android Developers, "Android Interface Definition Language (AIDL)," http://developer.android.com/guide/components/aidl.html, 2012.

[7] Android Developers, "Intents and Intent Filters," http://developer.android.com/guide/components/intents-filters.html, 2012.