Hybrid SDN Control in Mobile Ad Hoc Networks

Konstantinos Poularakis[1], Qiaofeng Qin[1], Kelvin M. Marcus[2], Kevin S. Chan[2],
Kin K. Leung[3], and Leandros Tassiulas[1]

[1]Department of Electrical Engineering and Institute for Network Science, Yale University, USA
[2]U.S. Army Research Laboratory, Adelphi, MD, USA
[3]Department of Electrical and Electronic Engineering, Imperial College London, UK

***Abstract*—Software defined networking (SDN) can be beneficial in mobile ad hoc networks (MANETs) to increase flexibility, provide programmability and simplify management. The high dynamics in mobile networks, however, raise new reliability challenges to the conventional centralized control plane of SDN. To address this issue, this paper proposes a hybrid architecture that splits the routing decision logic between centralized and distributed control planes. Experiments on a testbed built from commercial mobile devices with integrated SDN functionality highlight the feasibility and benefits of the proposed architecture.**

I. Introduction

Software Defined Networking (SDN) is a game changing, cutting edge technology in the network field that has been designed to enable more agile and simple network management [1]. With SDN, the control plane of the network becomes *centralized* and *programmable* so that the routing of traffic in the data plane can be configured at a software entity, the controller. This is radically different from current routing protocols where decisions are made in a distributed manner at each intermediate node in the routing path [2].

An SDN approach for MANETs can provide higher throughput and prolong network lifetime [3], [4]. However, the distinct features of the mobile network pose new *reliability* challenges to the SDN controller. The connectivity between the controller and a data plane node may be (temporarily) lost due to mobility of the latter. Although this node may be still connected with other nodes, its reconfiguration will be impossible as long as the controller is unavailable, resulting in outdated flow rules installed at the node.

Methods to cope with SDN reliability presented in literature typically add *redundancy* in the control plane in a brute force manner by placing multiple controllers in the network [5]. However, these methods cannot by themselves fundamentally solve the reliability problem as this would require to place controllers at all nodes and/or continuously migrate controllers among nodes so as to handle all possible failure scenarios in highly dynamic networks.

In this paper, we point out that to address the reliability issue of SDN in MANETs we need to complement the controller placement method with a distributed routing protocol (e.g., OLSR, AODV) that operates in the data plane. This leads to a new architecture that has a *hybrid* structure and splits the control (routing decision) logic between the controllers and the data plane nodes. To realize such an architecture, however, we need to define in a precise way how the SDN controllers will interface with the distributed protocol so as to coordinate their routing decisions, what information they will exchange, in what form and when.

This publication was supported by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001.

II. Experimentation Analysis

To motivate further the need for a hybrid control architecture, we implement a testbed of an SDN-enabled MANET and perform experiments. Our testbed consists of modern off-the-shelf smartphone devices that are commonly used today and which are set up with commercial SDN control plane and data plane software components.

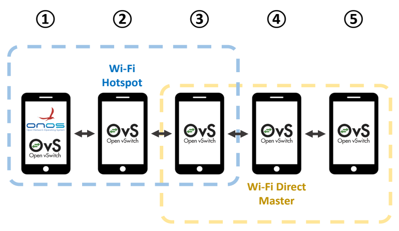
Specifically, we deploy five of the popular Android-based Nexus 4 smartphone devices to form a wireless network of line topology, as it is depicted in Figure 1(a). We use the WiFi Hotspot and WiFi Direct capabilities of Android to create the four wireless links. We run Open vSwitch (OvS) [6] on all the devices to turn them into programmable data plane nodes, while we vary the number of devices that run ONOS software [7], and hence the number of controllers, to test different scenarios.

We conduct experiments to explore the possible overheads of placing and migrating controllers among nodes to handle network failures. We place ONOS controllers at two of the devices, namely devices 1 and 3. Then, we dynamically place a new ONOS instance at device 2, and join it to the existing controller cluster. The new controller needs to exchange control messages with the two existing controllers to notify them and fetch network state information from them. We record the traffic exchanged between the new controller and its two peers since it joins the cluster. Figure 1(b) indicates that the synchronization process takes around 8 seconds, exchanging messages with a large overhead of several Mbps. *The time consumed and the burden on bandwidth for updating the controller cluster are significant costs, they will likely cause temporary service interruption, and cannot be ignored.*

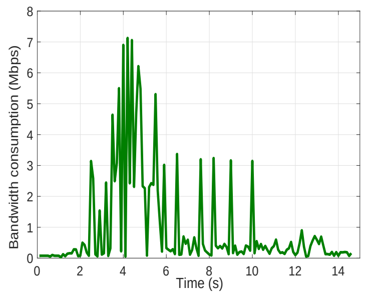
III. Hybrid SDN Architecture

The experimentation results of the previous section motivate the use of a hybrid SDN architecture to alleviate the reliability limitations and risks of service interruption inherent to the centralized SDN controllers. In this section, we describe such an architecture in detail.

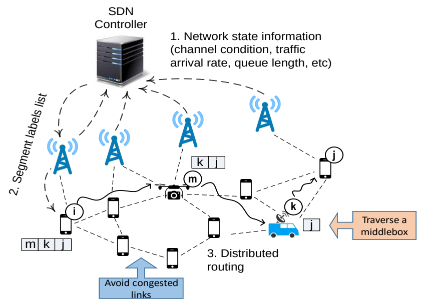
An illustrative example of the proposed architecture design is provided in Figure 1(c). This architecture is hybrid as it combines both centralized operations at the SDN controller (the server deployed in the infrastructure network side) and



(a) Testbed of five SDN-enabled smartphones.



(b) Controller placement adaptation overhead.



(c) Hybrid SDN architecture.

Fig. 1. (a) SDN-enabled MANET testbed. (b) Histogram of bandwidth consumption when a new controller is joining a cluster. (c) The proposed hybrid SDN architecture.

distributed non-SDN operations at the data plane nodes (the smartphones, drones, vehicles and other mobile devices in the ad hoc network side). We note that SDN controllers can be also placed in the ad hoc side of the network as shown by our experimentation analysis in Section II. The proposed architecture is independent of the exact controller placement strategy used.

To realize such a hybrid architecture, we use an idea similar to the segment routing technique [8]. Specifically, we propose the SDN controllers to break the routing path into *segments* and periodically broadcast the list of segment labels (endpoint nodes of the segments) to the data plane nodes rather than the entire set of flow rules required to be installed to support routing over this path. Having received the list of segment labels, the data plane nodes will have to make the routing decisions from one segment label to the next in the list. This can be achieved by running an existing distributed routing protocol such as OLSR. There are the following three key stages in the routing process.

i) Collection of network state information. This stage is similar to the conventional SDN architecture. The data plane nodes that experience changes in their states such as channel conditions, traffic arrival rates and queue lengths report the changes to the SDN controller in an *asynchronous push-based* manner. To avoid arbitrarily large traffic overheads in highly dynamic networks, this process can be instead initiated by the controller in a *synchronous pull-based* manner, e.g., request from nodes to report their state changes periodically.

ii) Computation and announcement of segment labels. Having collected the state changes from the data plane nodes, the SDN controller is able to construct the state of the entire network. This will be used to compute in a centralized manner the lists of segment labels. Specifically, for each pair of a source node i and a destination node j, the segment label list is a sequence of nodes the traffic should traverse during its transport. In the extreme case, this list contains only the destination node j in which case the controller has no impact on the routing path. In general, however, this list can be augmented with additional nodes that can act as *temporary destinations* for the traffic.

For example, in Figure 1(c), the controller has computed a segment list of nodes m, k and j for the communication pair i,j. This indicates that the traffic should be first routed from i to m, then from m to k and finally from k to i. This is particularly important since the nodes in the list can be selected such as the traffic traverses middleboxes or avoids being routed through congested regions of the network.

The SDN controller will have to announce the segment list to the data plane nodes. There are two alternative ways to perform the announcement. The first way is to send the list only to the source node (i in the example) and request from it to add the segment list labels to the IP header of the packets destined to j. Then, the next nodes in the list (m and k in the example) will have to pop the top label so as the temporary destination becomes the node after it. The main drawback of this way is the overhead in the packet header. The second way is to send the list to all the nodes participating in that list so that each node is aware of the temporary destination it has to route the traffic.

iii) Distributed routing. To perform the actual packet forwarding, the data plane nodes employ legacy (non-SDN) distributed routing protocols. These protocols are responsible for discovering and establishing routing paths between the nodes appearing sequentially in the same segment list. For example, the OLSR or AODV protocols can be employed. To achieve routing path discovery, these protocols typically rely on topology information flooding processes that can be optimized and have been proven to be quite robust and adaptive to network changes. Each time the SDN controller sends a new announcement to the data plane nodes, the latter will update their segment lists so that the new temporary destinations will be fed to the distributed routing protocol and the packet forwarding decisions will be updated accordingly.

References

[1] D. Kreutz, F. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, S. Uhlig, "Software-Defined Networking: A Comprehensive Survey", *in Proc. of the IEEE, vol. 103, no. 1, pp. 14-76*, 2015.
[2] A. Hinds, M. Ngulube, S. Zhu, H. Al-Aqrabi "A Review of Routing Protocols for Mobile Ad-Hoc NETworks (MANET)", *International Journal of Information and Education Technology, vol. 3, no. 1*, 2013.
[3] H.C. Yu, G. Quer, R.R. Rao, "Wireless SDN Mobile Ad Hoc Network: from Theory to Practice", *IEEE ICC*, 2017.
[4] K. Poularakis, G. Iosifidis, L. Tassiulas, "SDN-enabled Tactical Ad Hoc Networks: Extending Programmable Control to the Edge", *IEEE Communications Magazine, vol. 56, no. 7, pp. 132-138*, 2018.
[5] Q. Qin, K. Poularakis, G. Iosifidis, S. Kompella, L. Tassiulas, "SDN Controller Placement with Delay-Overhead Balancing in Wireless Edge Networks", *IEEE Transactions on Network and Service Management, vol. 15, no. 4, pp. 1446-1459*, 2018.
[6] https://openvswitch.org
[7] http://onosproject.org/
[8] R. Bhatia, F. Hao, M. Kodialam, T. Lakshman, "Optimized Network Traffic Engineering using Segment Routing", *IEEE Infocom*, 2015.