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 increase reliability, methods such as placing multiple controllers in the network have been considered that add redundancy in the control plane in a brute force manner. However, these methods cannot by themselves fundamentally solve the reliability problem. To address this issue, this paper complements the controller placement methods with a new architecture that has a hybrid structure splitting the routing decision logic between the controllers and the data plane nodes. Specifically, the controllers can break the routing path into segments, similar to the segment routing technique, and broadcast the list of segment labels to the data plane nodes. The latter are able to make the actual forwarding decisions for each segment in a distributed manner, e.g., by running an existing MANET protocol like OLSR. Experiments on a testbed built from commercial mobile devices with integrated SDN functionality highlight the feasibility and benefits of the proposed architecture.**

***Index Terms*—Software defined networks, hybrid networks, segment routing, mobile ad hoc networks.**

I. Introduction

A. Motivation

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 known as the SDN controller. This is radically different from current routing protocols where decisions are made in a distributed and hop-by-hop manner at each intermediate node in the routing path [2].

While SDN has already shown its important role in plenty of real-world deployments of data centers and service provider networks (e.g., B4 [3]), it can be also beneficial in mobile ad hoc networks (MANETs). In fact, recent works have demonstrated that an SDN-based approach for MANETs can provide higher throughput and prolong network lifetime compared to distributed routing protocols such as AODV and OLSR [4], [5], [6], [7]. At the same time, however, the distinct features of the mobile network pose new *reliability* challenges to the SDN controller.

To exemplify, in order for SDN to work properly, the continuous exchange of network state information (e.g., link conditions, traffic statistics) and flow rules between the controller and the data plane nodes is needed [8]. In the MANET environment, however, this operation is challenged by the low data rates of channels and the inherent problem of unreliable connectivity. For example, the connectivity between the SDN 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 [9]. To ensure that every data plane node maintains connection with at least one controller, these methods can *adapt* the placement of controllers over time [10], [11]. Such adaptation of controller placement, however, unavoidably causes temporary *service interruptions* and performance degradation due to the time required for the data plane nodes to discover and associate with the new controllers. More importantly, these methods cannot by themselves fundamentally solve the reliability problem as this would require to place controllers at all nodes so as to handle all possible failure scenarios in highly dynamic networks.

B. Methodology and Contributions

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.

Specifically, we propose the SDN controllers to break the routing path into *segments*, similar to the segment routing technique [12], 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.

The benefits of the above hybrid architecture are manifold. First, the *reliability* is improved compared to the conventional SDN architecture since the data plane nodes are able to make the routing decisions by themselves instead of relying on the flow rules installed by the SDN controllers. The SDN controllers are only used to guide the nodes to route the packets through a specific sequence of intermediate nodes (segment labels) before reaching the destination. However, these guidelines can be neglected by the nodes if the intermediate nodes cannot be found by the distributed protocol and the controllers are not reachable to update the guidelines. Second, the *performance* is improved compared to the current MANET protocols. With their guidelines, the SDN controllers can force the packets to go through specific nodes with special roles and middlebox functionalities (e.g., access policies) as well as indirectly load balance the network. Such service enforcement and traffic engineering capabilities are not provided by the current MANET protocols which are typically shortest path routing based.

To further motivate our work, 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 commecrial SDN control plane (ONOS [13]) and data plane (Open vSwitch [14]) software components. We show that forming SDN-enabled MANETs with multiple controllers is feasible with minor modifications to the software of existing network equipment. However, adapting the placement of controllers causes overheads and service interruption that are significant, which motivates our approach of hybrid SDN control.

The rest of the paper is organized as follows. We present the testbed and experimentation results in Section II. We describe the proposed hybrid SDN control architecture in Section III. Section IV discusses open issues and technical problems that can boost the benefits of our architecture. We conclude our work in Section V.

II. Experimentation Analysis

In this section, we implement a testbed of an SDN-enabled mobile ad hoc system and perform experiments. The results show the feasibility of these systems, reveal the impact of wireless multi-hop communication on the delay of data plane management, as well as the overheads of adapting the controller placement and controller-node assignment decisions.

A. Testbed Set-up

We build a testbed of a mobile ad hoc system with SDN functionality integrated into mobile devices. Namely, we use five of the popular Android-based Nexus 4 smartphone devices. We establish an Ubuntu environment running along with Android, so that we can install popular SDN-related software

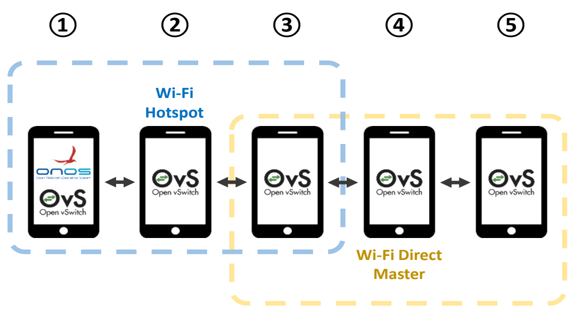


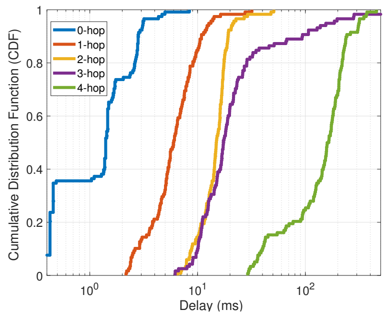
Fig. 1. Testbed consisting of five SDN-enabled smartphones connected through WiFi links.

in each device. The first necessary component is Open vSwitch (OvS) [14], by which a smartphone can be turned into a data plane node that supports OpenFlow protocol. The second necessary component is ONOS [13], by which a smartphone can be turned into an SDN controller. ONOS is designed particularly for scalability and permits multiple controllers to work together in the form of a controller *cluster*.

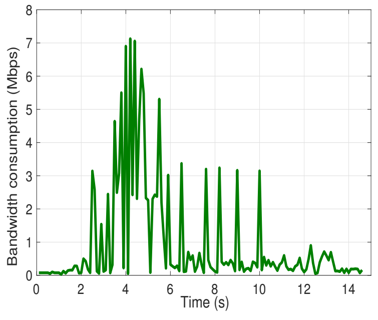
We deploy the five smartphones in a way that forms a linear topology with four wireless links, as it is depicted in Figure 1. This represents a common scenario where a node can either establish multi-hop connection to the backbone network (if any) or exchange data with its peers in a device-to-device (D2D) fashion. The devices are placed within an empty lab room, and distance between adjacent devices is 1.5 meters. Since the WiFi Ad Hoc protocol is not permitted by Nexus 4, we use the WiFi Hotspot and WiFi Direct capabilities of Android to create the four wireless links. First, we make device 2 to act as a WiFi access point (hotspot) and connect devices 1 and 3 to it. Then, we connect devices 3, 4 and 5 via the WiFi Direct protocol, where device 4 acts as the master. By applying a simple Network Address Translation (NAT) on device 3, the five devices can communicate with each other. Although constrained in resources, the smartphones are capable enough to run ONOS and OvS. We enable OvS on all the devices, while we vary the number of devices that run ONOS, and hence the number of controllers, to test different scenarios.

B. Experimentation Results

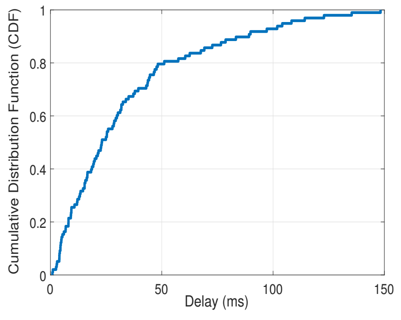
Impact of Multi-hop Communication. To examine the impact of multi-hop communication on the delay of managing the data plane nodes, we deploy an ONOS controller at device 1. We take measurements on the network delay between the controller and every data plane node. No matter what kind of routing policies is executed, a typical and universal interaction between a controller and a data plane node is the request and reply of flow statistics. Therefore, we define the delay as the interval between sending such an OpenFlow request message and receiving its corresponding reply. We keep capturing OpenFlow messages on the controller at device 1 for an



(a) CDF of controller-node delays.



(b) Histogram of bandwidth consumption when a new controller is joining a cluster.



(c) CDF of node reassignment delays.

Fig. 2. (a) The impact of number of hops on management delay. (b) Bandwidth consumption overheads due to adapting the placement of controllers. (c) Delay of reassigning a node between two controllers.

hour. As a result, Figure 2(a) is the cumulative distribution function (CDF) of the delays we measured. We can see that *the number of hops has a significant impact on the delay*. For example, the 4-hop communication has a severe delay of about 150 milliseconds on average, which is almost one order of magnitude higher than 1-hop or 2-hop connections.

Adaptation Costs. We conduct additional experiments to explore the possible overheads of dynamically adapting the controller placement and assignment policies. For the controller placement adaptation overheads, 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. We note that this operation of dynamically joining a new member in the controller cluster is not supported by the existing open-source ONOS code. Hence, we need to extend the code to enable such operation. 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 2(b) indicates that the synchronization process takes around 8 seconds, exchanging messages with a large overhead of several Mbps. Apparently, *the time consumed and the burden on bandwidth for updating the controller cluster are significant costs, they will likely cause temporary service interruption, and therefore cannot be ignored.*

We note that the placed controllers take different roles according to the OpenFlow protocol. For each data plane device, only one of the controllers has a *Master* role, i.e., gets full access to it. In addition to adapting the placement of controllers, the system may also decide to adapt the assignment of one or more data plane devices to controllers, i.e., hand over the Mastership for a device to another controller. It is therefore important to understand the overheads of such operation. To this end, we consider again the scenario of two controllers placed at devices 1 and 3 and dynamically reassign data plane device 2 between the two controllers. Once the reassignment starts, one controller sends a message to forfeit the mastership, and the other one sends a message to acquire it. Only when both of the messages are received and replied the controllers as well as the data plane device can make sure that the reassignment process is finished. Measurements show that although the controllers eventually synchronize with each other, these two messages do not always reach the data plane node at the same time. *This delay gap can be regarded as the cost of reassignment*. Figure 2(c) shows the CDF of the delay gaps over 100 measurements.

Main Takeaways. *The management delay heavily depends on the number of hops between the controller and the data plane node (up to 150 milliseconds for 4 hops). Adapting the placement of controllers and the assignment of data plane nodes takes time (up to 8 seconds for placing a new controller) which may cause service interruption, and induces significant bandwidth overheads for exchanging synchronization messages.*

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 3. This architecture is hybrid as it combines both centralized operations at the SDN controller (the server deployed in the infrastructure network side) and 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. There are three key stages in the routing process: i) the collection of network state information by the SDN controller, ii) the computation and announcement of segment label lists to the data plane nodes and iii) the distributed routing decisions made by data plane nodes running a legacy (non-SDN) protocol (e.g., OLSR). The details of each stage are given below.

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, the state collection 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. In each of the two cases, the process can be automated using a standardized Southbound protocol such as OpenFlow. We also note that if the control plane consists of multiple controllers, these will have to exchange messages to synchronize their local views of the network state using an East-Westbound protocol such as RAFT.

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 3, 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. Such service enforcement and traffic engineering capabilities are not supported by the current legacy (non-SDN) routing protocols which are typically shortest path based.

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 (which is also a drawback of the segment routing technique) 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. This essentially trades the packet header overhead with the traffic overhead of label list dissemination.

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.

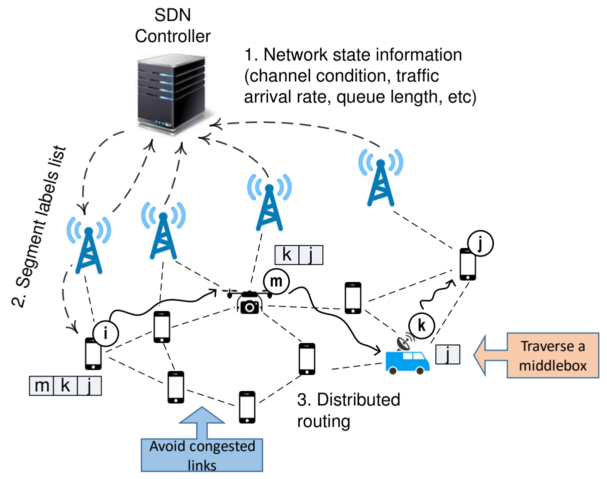


Fig. 3. Illustration of the proposed hybrid SDN architecture.

IV. Open Issues

In this section, we discuss some open issues and specific technical problems that, if addressed, can boost the benefits of the proposed architecture. Three of the most critical issues are discussed below.

i) Network optimization. The proposed hybrid architecture provides a flexible set of capabilities for traffic engineering and midllebox/service enforcement with granular control that are not supported by the current MANET protocols. These capabilities can be used to achieve several network objectives such as congestion minimization, end-to-end delay minimization, energy consumption minimization and maximization of the traffic steered through chains of services (e.g., firewall, deep packet inspection and video encoding). To achieve the above objectives, however, we need to define and solve new network optimization problems that model the relationship between the set of segment label lists and the routing paths that can be selected by the MANET protocol and compute the best segments accordingly. To this end, we can build upon and extend existing models and results in the segment routing literature (e.g., [15]) and tailor them to the distinct features of the MANET environment (e.g., mobility and dynamicity in the network, routing path discovery and selection decisions made by the MANET protocol, etc.).

ii) Overheads of hybrid SDN control. In order for the proposed architecture to work it runs two control planes simultaneously; the centralized SDN and the distributed MANET. Running two control planes instead of one, however, will unavoidably increase, and may even double, the traffic overheads associated with the collection of network state information and the configuration of routing paths. While some of the above traffic can be shared among the two control planes, provided that appropriate synchronization and routing of control messages are possible, the problem persists especially in resource-constrained ad hoc networks with low-capacity and in-band control channels. Another option is to dynamically tune the operation of the two control planes so that less traffic is consumed when the network is stable and therefore updating the segments by the SDN controller is less urgent. How to decide when the SDN controller should collect the network state information and update the segments is not straightforward. To answer this, we may benefit from online learning techniques, similar to [16], that can predict network dynamics and adapt traffic overheads accordingly.

iii) Coalition operations. An additional requirement that arose recently for MANETs is the increasing need to support coalition operations. This latter term describes communication between networks that belong to different organizations and administrative domains. In a tactical MANET, for example, communication between teams of soldiers (or, other actors in the tactical field) that belong to different command centers or even different nations is needed to circulate information such as mission data [17].

The proposed hybrid architecture can be used to support coalition operations among MANETs. However, we need to first address a number of issues. First, different domains may use different technologies. Some of the domains may have upgraded to SDN (and thus be centrally managed by SDN controllers) whereas others not. This creates a highly heterogeneous network environment where interoperability is a key concern. Second, to meet confidentiality objectives, the SDN controller of one domain may not be aware of the exact topology information in the other domains. Instead, the controller may only maintain an abstract view of it (e.g., end-to-end connectivity rather than exact topology). Therefore, it is unclear how to compute the segments to realize a multi-domain end-to-end path. An idea is to use an hierarchical control plane architecture where a "parent" controller computes the multi-domain path based on the segments received by "child" controllers [18]. However, all the participating domains will have to agree on a trusted entity to play the role of the parent controller.

V. Conclusion

In this paper, we studied the application of SDN technology to support communication and service requirements in MANETs. We showed the feasibility of such SDN-enabled MANET systems through a testbed built from commercial mobile devices. To alleviate the reliability limitations that are inherent to centralized SDN controllers, we proposed a hybrid SDN architecture which borrows ideas from the segment routing technique and employs existing MANET protocols to enable routing path discovery and establishment. Open issues and technical challenges related to the proposed architecture were discussed which can be rich areas for future work.

Acknowledgement

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. K. Poularakis acknowledges the Bodossaki Foundation, Greece, for a postdoctoral fellowship.

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] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Holzle, S. Stuart, A. Vahdat, "B4: Experience with a Globally-Deployed Software Defined WAN", *ACM Sigcomm*, 2013.

[4] H.C. Yu, G. Quer, R.R. Rao, "Wireless SDN Mobile Ad Hoc Network: from Theory to Practice", *IEEE ICC*, 2017.

[5] J. Wang, Y. Miao, P. Zhou, M. S. Hossain, S. Mizanur, "A Software Defined Network Routing in Wireless Multihop Network", *Journal of Network and Computer Applications, vol. 85, pp. 76-83*, 2017.

[6] M. Abolhasan, J. Lipman, W. Ni, B. Hagelstein, "Software-defined wireless networking: Centralized, distributed, or hybrid?", *IEEE Network, vol. 29, no. 4, pp. 32-38*, 2015.

[7] 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.

[8] H. Xu, Z. Yu, C. Qian, X. Li, Z. Liu, "Minimizing Flow Statistics Collection Cost of SDN Using Wildcard Requests", *in Proc. IEEE Infocom*, 2017.

[9] 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.

[10] T. Wang, F. Liu, J. Guo, H.A XuI, "Dynamic SDN Controller Assignment in Data Center Networks: Stable Matching with Transfers", *IEEE Infocom*, 2016.

[11] X. Lyu, C. Ren, W. Ni, H. Tian, R.P. Liu, Y.J. Guo, "Multi-Timescale Decentralized Online Orchestration of Software-Defined Networks", *IEEE Journal on Selected Areas in Communications, vol. 36, no. 12 , pp. 2716-2730*, 2018.

[12] Z.N. Abdullah, I. Ahmad , I. Hussain "Segment Routing in Software Defined Networks: A Survey", *vol. 21, no. 1, pp. 464-486*, 2018.

[13] http://onosproject.org/

[14] https://openvswitch.org

[15] R. Bhatia, F. Hao, M. Kodialam, T. Lakshman, "Optimized Network Traffic Engineering using Segment Routing", *IEEE Infocom*, 2015.

[16] K. Poularakis, Q. Qin, L. Ma, S. Kompella, K.K. Leung, L. Tassiulas, "Learning the Optimal Synchronization Rates in Distributed SDN Control Architectures", *IEEE Infocom*, 2019.

[17] J. Spencer. T. Willink, "SDN in Coalition Tactical Networks", *IEEE Milcom*, 2016.

[18] N. Kukreja, R. Alvizu, A. Kos, G. Maier, R. Morro, A. Capello, C. Cavazzoni, "Demonstration of SDN-based Orchestration for Multi-domain Segment Routing Networks", *ICTON*, 2016.