

 Open access • Proceedings Article • DOI:10.1109/NCA.2016.7778610

Enabling Software-Defined Networking for Wireless Mesh Networks in smart environments — [Source link](#)

Prithviraj Patil, Akram Hakiri, Yogesh Barve, Aniruddha Gokhale


Institutions: Vanderbilt University, Hoffmann-La Roche

Published on: 31 Oct 2016 - Network Computing and Applications

Topics: Wireless mesh network, Software-defined networking, Routing protocol, Smart environment and Smart grid

Related papers:

- [Towards Software Defined Wireless Mesh Networks](#)
- [Candidate solutions to improve Wireless Mesh Networks WMNs performance to meet the needs of Smart Grid applications -Survey paper](#)
- [JANUS: A Framework for Distributed Management of Wireless Mesh Networks](#)
- [Interoperability of Wireless Mesh and Wi-Fi network using FPGA for 4G solutions](#)
- [Multigate mesh routing for smart Grid last mile communications](#)

Share this paper:    

View more about this paper here: <https://typeset.io/papers/enabling-software-defined-networking-for-wireless-mesh-3vo22k42hg>



HAL
open science

Enabling Software-Defined Networking for Wireless Mesh Networks in Smart Environments

Prithviraj Patil, Akram Hakiri, Yogesh Barve, Aniruddha Gokhale

► **To cite this version:**

Prithviraj Patil, Akram Hakiri, Yogesh Barve, Aniruddha Gokhale. Enabling Software-Defined Networking for Wireless Mesh Networks in Smart Environments. 15th International Symposium on Network Computing and Applications (NCA 2016), Oct 2016, Cambridge, MA, United States. 10.1109/NCA.2016.7778610 . hal-01633331

HAL Id: hal-01633331

<https://hal.archives-ouvertes.fr/hal-01633331>

Submitted on 12 Nov 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Enabling Software-Defined Networking for Wireless Mesh Networks in Smart Environments

Prithviraj Patil*, Akram Hakiri^{†‡}, Yogesh Barve * and Aniruddha Gokhale*

*Dept of EECS, Vanderbilt University, Nashville, TN, USA.

[†] CNRS, LAAS, 7 Avenue du colonel Roche, F-31400 Toulouse, France

[‡]Univ de Carthage, SYSCOM ENIT, ISSAT Mateur, Tunisia.

Email: {prithviraj.p.patil,yogesh.d.barve, a.gokhale}@vanderbilt.edu, hakiri@laas.fr

Abstract—Wireless Mesh Networks (WMNs) serve as a key enabling technology for various smart initiatives, such as Smart Power Grids, by virtue of providing a self-organized wireless communication superhighway that is capable of monitoring the health and performance of system assets as well as enabling efficient trouble shooting notifications. Despite this promise, the current routing protocols in WMNs are fairly limited, particularly in the context of smart initiatives. Additionally, managing and upgrading these protocols is a difficult and error-prone task since the configuration must be enforced individually at each router. Software-Defined Networking (SDN) shows promise in this regard since it enables creating a customizable and programmable network data plane. However, SDN research to date has focused predominantly on wired networks, e.g., in cloud computing, but seldom on wireless communications and specifically WMNs. This paper addresses the limitations in SDN for WMNs by allowing the refactoring of the wireless protocol stack so as to provide modular and flexible routing decisions as well as fine-grained flow control. To that end, we describe an intelligent network architecture comprising a three-stage routing approach suitable for WMNs in uses cases, such as Smart Grids, that provides an efficient and affordable coverage as well as scalable high bandwidth capacity. Experimental results evaluating our approach for various QoS metrics like latency and bandwidth utilization show that our solution is suitable for the requirements of mission-critical WMNs.

Index Terms—SDN, Wireless Mesh Networks, Smart Grids, Home Area Network.

I. INTRODUCTION

Wireless networks, and in particular, Wireless Mesh Networks (WMNs) have been actively considered as one of the most promising wireless technologies to build highly scalable wireless backhaul networks [1] for Smart Grid power systems [2], renewable energy sources [3] and solar energy harvesting base stations [4]. These WMNs assume a hierarchical structure composed of Home Area Networks (HANs) and Neighbor Area Networks (NANs), which are plugged to a Network Gateway (NG) to access the wide area networks (WANs). A HAN connects a group of sensing devices in a home to smart meters that record the energy consumption for a given home and transmit the collected data to Meter Controlling Systems (MCS). NANs connect multiple MCSs of the HANs that are geographically in close proximity and interact with cloud services in the WAN for various kinds of data collection and analytics.

Despite the promise, current wireless routing protocols for WMNs are fairly limited and their extensions to power grid systems are very difficult. For example, in traditional WMNs, the nodes (i.e., mesh switches and routers) communicate with each other using routing protocols like AODV (Ad hoc On Demand Distance Vector) and OLSR (Optimized Link State Routing Protocol), which are inherently distributed because each node broadcasts information about its directly connected end devices to all other nodes. Using this information, each node will derive its own routing path to all the other nodes independently and in a distributed fashion. As a result, they reflect a partial visibility of the network without paying attention to the real network conditions. This local visibility limits the ability of WMNs to perform network engineering in large-scale wireless networks. Additionally, WMNs are difficult to manage and upgrade because their configuration must be enforced manually at each mesh router, which is both difficult and error-prone.

Software-Defined Networking (SDN) [5], [6] shows significant promise in meeting the networking needs of Smart Grid systems [7] due to the separation of the control and forwarding plane. Recent trends in Smart Grids reveal that SDN has been used to implement a multi-rate, multicast network for Phasor Measurement Unit (PMU) data [8]. Similarly, Rinaldi et al. [9] investigated a wired SDN controller to manage the network infrastructure of smart grid and provide a resilient Smart Grid systems [10]. Despite these advances, all these efforts have used SDN only in wired networks, which make their solutions unusable in smart environments due to their distributed and often wireless nature.

Addressing these needs is challenging for the following reasons. Since SDN was initially conceived for wired networks, the controller in the traditional case statically establishes paths to every switch so that it can then run centralized routing algorithms. In wireless networks, however, the controller has to discover all the switches before it can run the centralized routing algorithms. This can be achieved by installing wireless routing algorithms like AODV and OLSR in switches, but this means that the switches lose the simplicity that is envisioned by the SDN paradigm, which calls for no intelligence to reside in the switches. In other words, the SDN requirements of a centralized routing control and simple switch design contradict with the distributed routing algorithms and sophisticated

switch design of the wireless network architecture.

Consequently, we need an approach that enhances SDN for WMNs such that the new approach can centralize the control decisions – a trait of SDN – over distributed wireless mesh networks. To realize such a capability, we present a novel approach for creating an intelligent Software-Defined Wireless Mesh Network suitable for use cases, such as Smart Grids. Our approach proposes a novel way of performing routing in three stages in SDN-based WMNs by using a modified OpenFlow protocol, which allows us to remain faithful to the SDN philosophy of keeping the switch design simple and of a logically centralized control plane while also allowing flexibility and mobility that is inherent in distributed wireless mesh networks. The rest of the paper delves into the details and evaluation of our proposed approach.

II. RELATED WORK

This section compares related work along two dimensions with our work.

A. SDN and Routing in Wireless Mesh Networks

Wm-SDN [11] attempts to address the routing challenge in SDN-based WMNs described in Section I by using a hybrid protocol. It uses the traditional AODV distributed protocol for switch-controller connection. Subsequently, when the controller-switch connection is established, it then uses SDN-based centralized protocols for switch-switch routing decisions. In this architecture each switch must support both SDN OpenFlow and also legacy routing protocols. Consequently, this approach is technically not a pure SDN-based approach since the switch is involved during the first part of routing decisions (i.e. during AODV). Moreover, it requires the switch to support complex hardware and software than what SDN envisions.

Some other efforts [12] [13] have also proposed hybrid approaches for routing though of a different kind. They propose to combine SDN-enabled switches and legacy switches (or routers) in a wireless mesh router, where SDN-enabled switches form the SDN network while traditional switches form the legacy network. In this architecture, the legacy switches run traditional routing algorithms (OSPF in this case) while every SDN enabled switch has to be in direct contact (wireless or wired) with one such legacy switch. This allows SDN-enabled switches to not support any complex hardware and software. However it requires each SDN-enabled switch to communicate with at least one legacy switch.

B. SDN-enabled Smart Grids

There are some research initiatives to leverage the potential of SDN in Smart Grid communication. Jianchao et al. [14] discussed opportunities that bring SDN to support the potential use cases in Smart Grid. Similarly, Dong et al. [10] studied the benefits of SDN to improve network resilience in Smart Grid. Rinaldi et al. [9] proposed using a wired SDN controller to simplify and automate the network management in power grids. Cahn et al. [15] presented a Software-Defined

Energy Communication Network (SDECN) framework for self-configuring IEDs for substation automation. Likewise, Dorsch et al. [7] presented fast recovery and load management algorithms for the distribution and transmission in power grids. Additionally, multicast Phasor Measurement Unit (PMU) has been investigated [8]. Thomas et al. [16] proposed using a SDN-enabled multicast scheme to support flexible and fault-tolerant group communications in power systems.

Although the previous works enabled SDN in smart grids, they consider only wired communications. In contrast to wired networks which are known to be stable and robust, our contribution addresses wireless mesh smart grid networks so as to extend power systems to rural and disaster regions. Our solution provides an efficient and affordable coverage as well as low-latency, and flexible and network-aware communications.

III. THREE-STAGE ROUTING ARCHITECTURE AND DESIGN CONSIDERATIONS

This section delves into the details of our three-stage routing approach for SDN-enabled wireless mesh networks.

A. Three-Stage Routing in SDN-enabled WMNs

In this section, we describe our 3-stage routing approach for SDN-enabled WMNs. Our work is realized as an extension to the OpenFlow protocol for the wireless mesh networks, where the new three-level routing strategy enables existing OpenFlow protocol to adapt to the dynamic nature of the WMNs. Below we describe the three stages of this routing strategy.

- **Stage 1: Initial Controller-Switch Connection:** As shown in Figure 1, in the SDN-based wireless mesh networks, only a few switches are directly connected to the controller. The first task is to connect all the switches to (at least one) controller by setting up initial/basic routing. We propose an initial (i.e., non-permanent) routing stage where a SDN controller will find all the switches by flooding the network without considering whether the path it finds is best or not. To achieve this, we use an OpenFlow-based routing algorithm for initial controller-switch connection by adapting OLSR in two ways. First, instead of the switches broadcasting their link state (i.e. information about directly connected end devices), the controller will broadcast information about its directly connected switch. Second, instead of running a pure wireless mesh routing protocol like AODV or OLSR in the switches, we modify the OpenFlow client in the switch such that it finds the initial path to the controller without requiring any additional software.
- **Stage 2: Controller-Switch Path Optimization:** Once the initial connection is established, the routing paths set up in this stage will be used to install new alternatives (i.e., shortest, optimum or load balanced) paths. The controller can decide to choose a path among these alternative paths between itself and a switch since at this point, the controller has a global view of the network. Thereafter, it installs the corresponding rule to

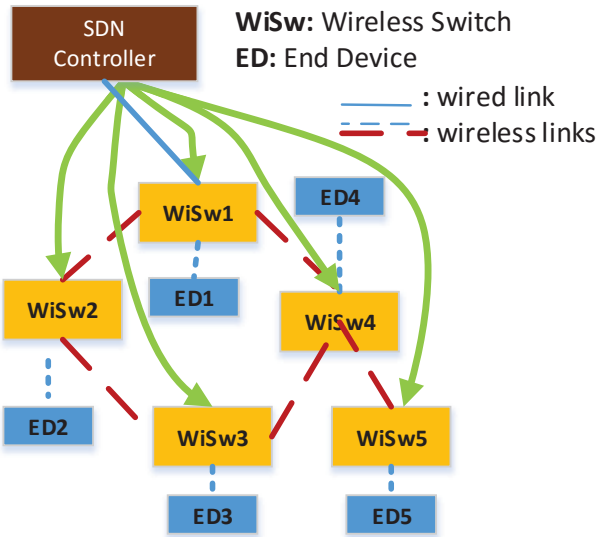


Fig. 1: Example SDN Mesh Scenario

route packets from the switches to itself in the switches by propagating the information using the original non-optimized paths.

- **Stage 3: Routing among Switches:** After the second step, the controller will derive the shortest path routing among switches themselves and it will then install these routing paths and corresponding forwarding rules using the shortest paths set-up in the previous stage. As described above, to achieve the above routing strategy, we had to modify the OpenFlow client.

B. Implementation Details

We now present the details of our implementation and the message types we created for the three stages of our approach.

- 1) *OF_Initial_Path_Request*: Initially, the controller will send the *OF_Initial_Path_Request* messages to all its directly connected switches. As described in Figure 1, the switches could be either connected via wired interface or wireless interface to the controller. Additionally, the controller could be at the same location as that of the switch or could be situated in the cloud data center. Once a switch receives those messages, it updates the controller path destination with the source id found in the received *OF_Initial_Path_Request* message. Then, the switch creates a new *OF_Initial_Path_Request* message with its own source id and broadcasts it to the other switches. This step is performed periodically by every switch. Thereafter, every switch that receives the *OF_Initial_Path_Request* message establishes an initial path to the controller. This path may not provide shortest paths but only be used as a first step for obtaining the shortest path in the next stage. Also, since every switch broadcasts this message periodically, this helps to handle the mobility in the network.
- 2) *OF_Initial_Path_Response*: The switch sends this message to the controller on finding the initial path in stage

I. This message is directed towards the controller and is only sent to that neighbor from which the switch received *OF_Initial_Path_Request* message first, i.e. this message is sent via the initial path between switch and controller. However, this response message contains SSIDs of all the neighboring switches. i.e. all the neighboring switches from whom this switch received *OF_Initial_Path_Request* message.

- 3) *OF_Controller_Shortest_Path*: This OpenFlow message is used to optimize the initial connection path between the controller and the switch. The Controller sends this message to the switches to update the path towards the controller with the shortest path. This message is sent only when the initial path differs from the shortest path between controller and switch. Moreover, this message is always sent using the initial path. When the switch receives this message, it installs the new path to the controller, which is shorter than the previous path. Moreover, the switch gives this new path higher preference by installing the rule for this path before the rule of the initial path. So from that point onward, whenever the switch sends any message to the controller, it takes the shortest path. Only when the shortest path fails to deliver a message, the initial path is used as a backup.

C. Demonstrating the Approach in a Use Case

We now demonstrate how our approach works for a topology shown in Figure 1. Each switch is connected to a single edge device while only the first switch is connected to the controller directly.

Figure 2 shows the interactions in all the three stages. In Stage-I, the controller sends the *OF_Initial_Path_Request* to Switch-1 using flooding. Switch-1 then duplicates this message and sends it to Switch-2 and Switch-4. This allows each switch to find an initial path to the controller. In Stage-II, each switch sends *OF_Initial_Path_Response* message to the controller via the initial path found in Stage-I. This message contains information about neighboring switches.

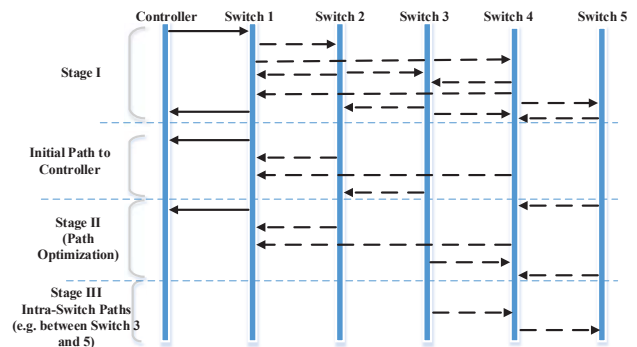


Fig. 2: Three-Stage Routing Interactions

In our example, Switch-3 has received *OF_Initial_Path_Request* from two switches (Switch-2 and Switch-4). However, it has received the message from

Switch-2 first. Hence, for Switch-3, the initial path to the controller is via Switch-2. However, when the Switch-3 replies to the Controller using `OF_Initial_Path_Response` message via initial path (i.e. via Switch-2), it includes the SSID of Switch-2 and Switch-4 in it. This allows the Controller to deduce the neighbors of each switch and hence the topology of the entire network. Using the knowledge about the topology of the network, the controller now installs the shortest path routes in switches as shown in Figure 2 where Switch-3 now has a shorter path to the Controller via Switch-4 instead of via Switch-2.

Finally in Stage-III (Figure 2), the controller installs routing paths among switches using the shortest path from Stage-II. As shown in Figure 2, the Controller installs OpenFlow rules such that Switch-3 can reach Switch-5 via Switch-4.

IV. EXPERIMENTAL EVALUATION

We have implemented our three-stage routing strategy using the SDN emulation framework called Mininet. The basic Mininet does not provide wireless link support. An extension called Mininet-WiFi provides basic support for simulating wireless links but lacks support for essential wireless network based algorithms like shortest path or AODV or OLSR, which are normally supported by other network simulators like NS2 or NS3. Hence, we have used NS3 which is a network simulator with support for wired and wireless links. NS3 also provides support for OpenFlow client, which is required for SDN based switched. Thus, in order to bridge NS3 with Mininet, we have used OpenNet. The OpenNet simulator bridges NS3 and Mininet to provide wireless simulation in the SDN based network settings.

Once the wireless mesh network (of mobile switches and mobile hosts) is created using Mininet and NS3, the SDN controller is connected to one or more of the wireless switches. For this purpose we have used the POX controller. The three stage routing strategy is implemented on top of the POX controller as a network application. This strategy also makes use of shortest-path algorithms supported by the NS3 simulator under the hood.

We have evaluated our approach by comparing its performance with the hybrid approach of Figure ???. For comparison we used three metrics (1) controller-switch connection latency, (2) controller-switch reconnection latency, and (3) switch-switch connection latency.

In the beginning, the controller will try to connect to all the switches using messages `OF_Initial_Path_Request` and `OF_Initial_Path_Response` for finding the initial path in Stage-I, which in turn will be used to install the shortest path in Stage-II. We measure the latency to perform Stage-I and Stage-II, and compare it with the hybrid approach. Figure 3 plots this controller-switch connection latency against the number of hops between the controller and switch. We measure this latency for the hybrid approach, Stage-I and Stage-II. As can be seen from the figure, our approach basically breaks down the latency required by the hybrid approach into two stages (Stage I and Stage-II). The latency incurred by the hybrid

approach is approximately the sum of the latency incurred by Stage-I and Stage-II.

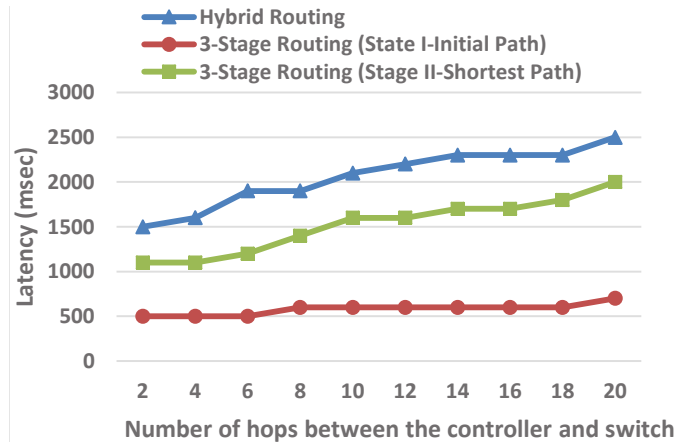


Fig. 3: Controller Switch Connection Latency

We observed the same behavior when we measured the latency for re-connecting the controller-switch link during failure. We measured this latency against the number of broken links between controller-switch as seen in Figure 4. Here also we can see the Stage-I and Stage-II reconnection latency adds up to the reconnection latency in the hybrid approach.

It is evident that our approach breaks up routing into two stages where the first stage finds a potentially inefficient route to the controller but takes lesser time while the second stage tries to optimize the route found in the first stage but takes more time. This helps overall performance of actual routing between switch-switch connection in Stage-III as seen in Figure 5. This figure shows the connection latency among switches against number of hops between them. It is seen that the stage-III of our approach outperforms the hybrid approach as the number of hops increase between switches. The reason for this result is that the switch has better connection to the controller in our approach than in the hybrid approach as shown in Figures 3 and 4.

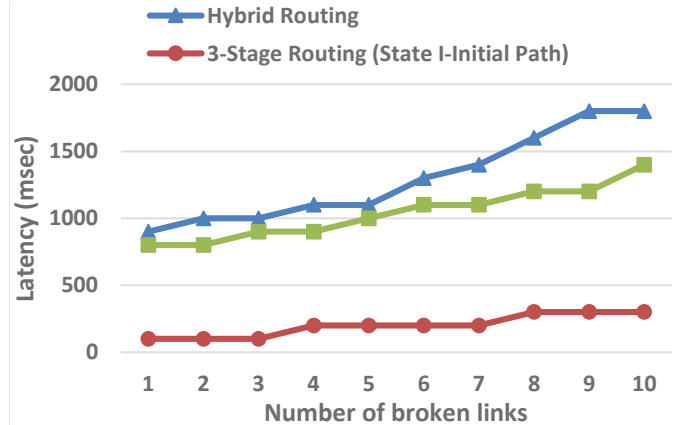


Fig. 4: Controller Switch Re-Connection Latency

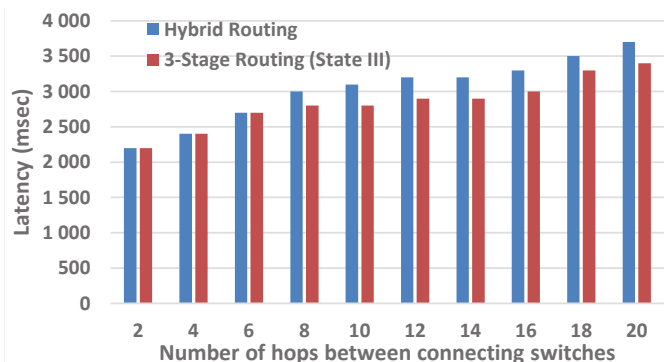


Fig. 5: Switch-Switch Connection Latency

In SDN, whenever a switch wants to connect to another switch, it requests the controller to install the routing rules. Hence, the connection to the controller plays a big role in the switch-switch connection latency. In wired networks, there is almost no mobility of nodes and hence once a controller installs rules in switches, these rules may not need to be changed frequently. Hence, routing among switches is not impacted by the controller-switch latency in wired networks. However in wireless mesh networks, as nodes can move more frequently, the switch needs to establish a reliable connection to the controller in order to improve the switch-switch routing.

This is the advantage of our approach over the hybrid approach. The hybrid approach always tries to find the best route to the controller, which in turn incurs higher latency and hence the controller-switch connection becomes unavailable for longer durations. Our three-stage approach, however, tries to find the first available route to the controller incurring smaller latency in Stage-I which helps to keep controller-switch connection alive for longer durations and thereby improving system availability as well as helping in reducing the latency in Stage-III.

V. CONCLUSIONS

In this paper, we proposed a three-stage routing strategy to efficiently use SDN in wireless mesh networks. In this regard, we proposed extensions to the existing OpenFlow protocol with three new type of messages which facilitates the three stage routing. We then evaluated the three-stage routing approach using latency metrics: one for the connections between the controller and switch, and another for connection between switches. The code for the prototype simulation can be found at ¹. In this work, we used a centralized controller in this work. However, in wireless scenarios, distributed controllers are more realistic. We plan to extend our work to centralized controllers and compare the performance.

ACKNOWLEDGMENTS

This work was funded by the Fulbright Visiting Scholars Program, NSF CNS US Ignite 1531079 and by AFOSR

DDAS FA9550-13-1-0227. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of Fulbright, NSF and AFOSR.

REFERENCES

- [1] IEEE Std 1646-2004, "Ieee standard communication delivery time performance requirements for electric power substation automation," *IEEE Std 1646-2004*, pp. 1–24, 2005.
- [2] Y. Xu and W. Wang, "Wireless mesh network in smart grid: Modeling and analysis for time critical communications," *Wireless Communications, IEEE Transactions on*, vol. 12, no. 7, pp. 3360–3371, July 2013.
- [3] L. Cai, Y. Liu, T. Luan, X. Shen, J. Mark, and H. Poor, "Sustainability analysis and resource management for wireless mesh networks with renewable energy supplies," *Selected Areas in Communications, IEEE Journal on*, vol. 32, no. 2, pp. 345–355, February 2014.
- [4] Z. Fadlullah, T. Nakajo, H. Nishiyama, Y. Owada, K. Hamaguchi, and N. Kato, "Field measurement of an implemented solar powered bs-based wireless mesh network," *Wireless Communications, IEEE*, vol. 22, no. 3, pp. 137–143, June 2015.
- [5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.
- [6] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: challenges and research opportunities for future internet," *Computer Networks*, vol. 75, pp. 453–471, 2014.
- [7] N. Dorsch, F. Kurtz, H. Georg, C. Hagerling, and C. Wietfeld, "Software-defined networking for smart grid communications: Applications, challenges and advantages," in *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, Nov 2014, pp. 422–427.
- [8] A. Goodney, S. Kumar, A. Ravi, and Y. Cho, "Efficient pmu networking with software defined networks," in *Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on*, Oct 2013, pp. 378–383.
- [9] S. Rinaldi, P. Ferrari, D. Brandao, and S. Sulis, "Software defined networking applied to the heterogeneous infrastructure of smart grid," in *Factory Communication Systems (WFCS), 2015 IEEE World Conference on*, May 2015, pp. 1–4.
- [10] X. Dong, H. Lin, R. Tan, R. K. Iyer, and Z. Kalbarczyk, "Software-defined networking for smart grid resilience: Opportunities and challenges," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, ser. CPSS '15, 2015.
- [11] A. Detti, C. Pisa, S. Salsano, and N. Blefari-Melazzi, "Wireless mesh software defined networks (wmsdn)," in *2013 IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob)*. IEEE, 2013, pp. 89–95.
- [12] Y. Peng, L. Guo, Q. Deng, Z. Ning, and L. Zhang, "A novel hybrid routing forwarding algorithm in sdn enabled wireless mesh networks," in *High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conference on Embedded Software and Systems (ICSS), 2015 IEEE 17th International Conference on*. IEEE, 2015, pp. 1806–1811.
- [13] A. Abujoda, D. Dietrich, P. Papadimitriou, and A. Sathiseelan, "Software-defined wireless mesh networks for internet access sharing," *Computer Networks*, vol. 93, pp. 359–372, 2015.
- [14] J. Zhang, B.-C. Seet, T.-T. Lie, and C. H. Foh, "Opportunities for software-defined networking in smart grid," in *Information, Communications and Signal Processing (ICICSP) 2013 9th International Conference on*, Dec 2013, pp. 1–5.
- [15] A. Cahn, J. Hoyos, M. Hulse, and E. Keller, "Software-defined energy communication networks: From substation automation to future smart grids," in *Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on*, Oct 2013, pp. 558–563.
- [16] T. Pfeifferberger, J. L. Du, P. Bittencourt Arruda, and A. Anzaloni, "Reliable and flexible communications for power systems: Fault-tolerant multicast with sdn/openflow," in *New Technologies, Mobility and Security (NTMS), 2015 7th International Conference on*, July 2015, pp. 1–6.

¹<https://github.com/prithviraj6116/sdn-wireless-mesh>