OpenFlow-Based Load Balancing for Wireless Mesh Network

Hanjie Yang[1(✉)], Bing Chen[1], and Ping Fu[2]

[1] Institute of Computer Science and Technology,
Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, China
jane09-01@hotmail.com, cb_china@nuaa.edu.cn
[2] Central Washington University, Washington, USA
pingfu@cwu.edu

Abstract. Wireless mesh network (WMN), an emerging network which is the pivotal technology for the next generation networks, is intended to provide reliable data transmission at low cost and high throughput. Load balancing is an essential problem, which affects the performance of the network. OpenFlow, an emerging technology, can effectively monitor and manage the network topology as well as network traffic, via the separation of the data layer and the control layer. This paper proposed an OpenFlow-based WMN and a channel related solution to solve the load balance. Control of each node dynamically changes while the network traffic changes, and the channel allocation along with the establishing of data paths will be promptly adjusted. We have built our WMN testbed with OpenFlow-based wireless mesh nodes. We carried out some experiments to evaluate our solution. Our experiments confirm that OpenFlow sets a good technology to solve wireless mesh network load imbalance.

Keywords: Wireless mesh network · OpenFlow · SDN · Load balancing

1 Introduction

Wireless mesh networks are intended for the last mile broadband Internet access to extend or enhance Internet connectivity for mobile clients, which can provide high network throughput, as well as optimal load balancing. In a typical mesh network, mesh routers collect information from local mesh nodes, and the routing algorithm decides the forwarding paths according to these information. Traditional wireless mesh routing algorithms are usually distributed, so that the solution of the network is usually deployed on each mesh nodes. It is difficult to increase the routing algorithm, and then achieve the higher network throughput.

In wireless mesh networks, load balancing is critical. Load imbalance makes some nodes become bottleneck nodes. Because the forwarding traffic of these nodes are too much, the network performance will decline throughout the network. How to build optimal mesh networks with load balancing has been studied theoretically. A variety of routing algorithms have been put forward to solve the load balancing problem in mesh networks. However, these algorithms can't dynamically adapt to current network topology and dataflow changes, avoid the bottleneck node, and select the most stable link to establish a route.

© Springer International Publishing Switzerland 2015
Z. Huang et al. (Eds.): ICCCS 2015, LNCS 9483, pp. 368–379, 2015.
DOI: 10.1007/978-3-319-27051-7_31

OpenFlow is a new kind of protocol created initially for communication in a programmable network, via the separation of the data layer and control layer. Using OpenFlow in a wireless network has been put forward [1]. Hence, we are considering that we can develop a wireless mesh network based-on OpenFlow to acquire information from the whole network, then the controller decides the best data path, which can ensure high throughput data transfer. In this paper, we propose an OpenFlow-based wireless mesh network architecture. The object of our work is to present our OpenFlow-based network design and the load balancing performance under practical application scenarios.

In the following sections, we present related works about the solutions for load balancing in wireless mesh networks in Sect. 2. Section 3 describes the background of OpenFlow-based Mesh nodes, as well as our design of the OpenFlow-enabled mesh nodes and controller. In Sect. 4, we describe something about the experimental environment, how we implement our testbed. Section 5 describes the detailed setup of our experiments. We considered several scenarios and the solution to solve the load imbalance and the analyzing of the results.

2 Related Work

There has been significant prior works on load balancing strategies based on traditional wireless mesh networks. Most prior works focus on distributed algorithms, where the mesh nodes communicate only with their neighborhood [3, 4]. Routing protocols for mesh networks can generally be divided into proactive routing, reactive routing, and hybrid routing strategies [2]. Nevertheless, most of these protocols do not provide load balancing strategy. Because of the put forwarding of OpenFlow, a centralized algorithm in WMN is possible.

In the [5], the combination of the OpenFlow and the WMN has been proposed for the first time. In that paper, OpenFlow and a distributed routing protocols (OLSR) are combined in a Wireless Mesh Software Defined Network (wmSDN). Load balancing using both wireless mesh network protocols and OpenFlow is discussed in [6]. The experiments in the paper demonstrate the improved performance of OpenFlow over traditional mesh routing protocols. As it shows that the OpenFlow controller is able to make centralized decisions on how to optimally route traffic so that the computational burden at each node is minimized. However, the study does not consider the link quality between the mesh nodes and the topology of the network does not contain the gateway nodes. In the [9], the author proposed a prototype mesh infrastructure where flows from a source node can take multiple paths through the network based on OpenFlow to solve the load balancing for wireless mesh network. However, the study doesn't adequately consider the multi radio interfaces of the mesh nodes. Therefore, considering the characteristics of the wireless mesh network, we proposed a solution for multi-interfaces wireless mesh networks. We also have implemented the OpenFlow-enabled mesh network testbed which each node in the testbed has two radio interfaces.

3 The Wireless Mesh Network Based on OpenFlow

3.1 Wireless Mesh Network

A wireless mesh network (WMN) is a kind of ad hoc network, however it communicates in multi-hop fashion instead of communicating in one-hop fashion in typical ad hoc network. All the links between the wireless mesh nodes are established by radio. Akyildiz et al. [8] describes the wireless mesh network as dynamically selforganized and selfconfigured networks, with the mesh nodes automatically establishing an ad hoc network and maintaining the mesh connectivity.

Instead of being another type of ad hoc networking, WMNs diversify the capabilities of ad hoc networks with many advantages such as low up-front cost, easy network maintenance, robustness and reliable service coverage, etc. Nowadays, the WMNs are undergoing rapid commercialization in many application scenarios such as broadband home networks, community networks and high-speed metropolitan area networks, etc. WMNs consist of two types of nodes: mesh router and mesh client. Mesh routers conform the wireless backbone of the network, while the mesh clients are connected to the network through them. Further, there are some mesh routers with a gateway (GW) capabilities, which are connected to other communication networks (including the Internet) through wired links.

In a general way, there are three types of the wireless mesh network, which is classified through architecture: infrastructure WMN/Backbone WMNs, client WMNs and hybrid WMNs. In the infrastructure WMN/Backbone WMNs, mesh routers form a mesh of self-configuring, self-healing links among themselves, providing a backbone for mesh clients. Client WMN is a kind of peer-to-peer network among clients. Thus, a Client WMN is actually the same as a conventional ad hoc network. Hybrid WMN is the combination of infrastructure and client WMNs. Hybrid WMNs offer the best coverage area, as mesh clients can access to the network through mesh routers as well as directly meshing with other mesh clients.

3.2 Software-Defined Networking

Software-defined networking (SDN) is a new kind of network architecture to enable people build programmable networks as a way to reduce the complexity of network configuration and management. It can facilitate the provisioning of network services in a deterministic, dynamic, and scalable manner. SDN currently refers to approaches for networking in which the control plane and the data plane are decoupled and is governed by a logically centralized controller. This characteristic reduces the complexity in the network, managing the network as one entity. There will be an interface provided by the controller in order to configure networks and the controller is responsible for directing the configuration to the network. In this case, network operators are able to dynamically adjust the network's traffic flows to meet the changing needs while optimizing the network resource usage.

However, under the circumstances of wireless mesh networks, SDN still faces several challenges. Firstly, the centralized controller will cause a single point of failure:

if a WMR loses communication with the controller, the new flows fail transmitting; if the SDN controller fails, the whole network breaks down. Furthermore, centralized control for WMN would require transferring a considerable amount of status information and configuration commands between WMN nodes and the centralized control entity, which will cause the longer delays. To deploy an appropriate SDN strategy, we can make better use of SDN technologies.

3.3 OpenFlow

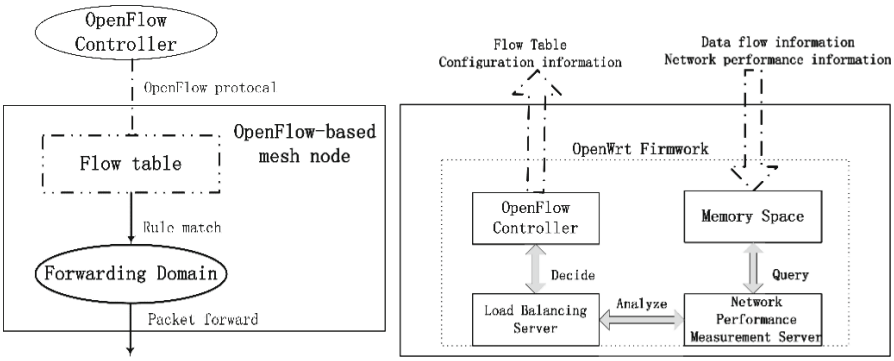
OpenFlow was originally designed for providing a real experiment platform to campus network researchers designing innovation network architecture, then McKeown et al. [10] started promoting SDN concept, and the concept aroused wide attention of academia and industry. It is a new switching protocol based on the concept of software defined networking(SDN).

OpenFlow-enabled switches move packet forwarding intelligence to the OpenFlow controller, while keeping the switches simple. In legacy switch structures, there is a control domain and a forwarding domain. As there is no control domain residing at an OpenFlow-based switch, the forwarding domain can be kept simple, and they do the forwarding function based on the flow tables. The functionality of the control domain is now moved to the control network, which was referred to as at least one OpenFlow controller or more. The controller is connected to every switch by a secure channel, using the OpenFlow protocol. OpenFlow makes packet forwarding and routing more intelligent than legacy routing solutions, making it possible to develop more complex routing protocols that further improve network performance. In our implementation, we utilize the protocol OpenFlow 1.3 [7] to construct our OpenFlow-based wireless mesh node.

3.4 OpenFlow-Based Wireless Mesh Node

In a typical mesh-based wireless network, there are generally three components: mesh clients, mesh routers and Internet gateways. In this paper, each mesh node (including Internet gateway) is based on OpenFlow and responsible for mesh connectivity and traffic routing. Because of the OpenFlow, each node is divided into two virtual planes, a data plane and a control plane. The control plane of the traditional mesh node resides on the switch. Thus, the OpenFlow Based mesh node's resides on the network-wide controller. The data plane consists of a set of rules, which the flow tables are sent by the controller. As shown in Fig. 1(a), we install OpenvSwitch for each device. The control and the data interfaces of each mesh node are two different physical interfaces.

The OpenFlow defines a secure channel between the data plane and the controller plane using a set of rules, properties, an expiration time and a list of actions. The properties specify packet source, original header values, and switch ports. When a new packet arrives at the switch network interface, while there is no match rule in the switch, the switch will ask the controller and the controller will decide whether the packet is dropped or forwarded.



(a) OpenFlow-based Mesh Node (b) OpenFlow controller

Fig. 1. Architecture of mesh node and controller

3.5 OpenFlow Controller

In our hypothesis, the OpenFlow controller does not only have the function of distributed flow table, also can change some of the configuration to the OpenFlow-based switches. On this occasion, we have modified the POX controller, making the system can identify the IP addresses of the connecting nodes, also can log on these mesh nodes and configure them.

We consider the architecture of the OpenFlow controller is like the Fig. 1(b). The controller consists of memory space, network performance measurement server, load balancing server, and OpenFlow controller. The performance server queries information from the OpenFlow-based switches/Internet Gateway and builds a data store used to support the load balancing server. Network topology changes are sent to this controller, then the controller will update the information including the channel usage based on the new network graphs. The main purpose of the OpenFlow controller is to perform basic load balancing tasks. The processing of the task is described as followed: (1) the controller achieves the source and destination addresses by searching the keyword of packet_in message from the source node, (2) the performance server may do the strategic analysis according to the information from memory space, (3) load balancing server firstly calculates a set of alternate paths using OLSR algorithm, and selects a optimal path from them in accordance with the strategics, (4) OpenFlow controller will send flow table and configuration information to ensure the data transmit correctly.

There has been some simple strategics: Each data flow can be assigned to a separate path and the assignments can change dynamically based on network state; High priority traffic will get better service while the best effort traffic suffers most of the damage; The 'fat' non-realtime traffic flows can be split to multiple paths as long as the paths end at the same gateway etc.

4 Testbed and Implementation

Our testbed was a small scale wireless mesh network consisted of at least four wireless routers which was placed as depicted in Fig. 2. The experiments were conducted inside our lab building.

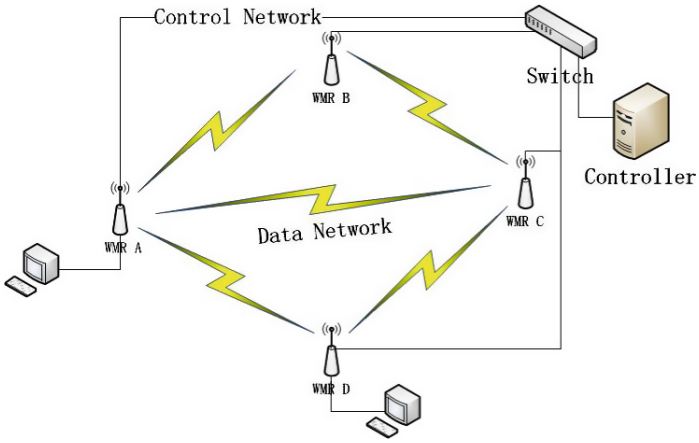


Fig. 2. WMN testbed with OpenFlow controller

All the wireless routers in our testbed are the PC Engines ALIX series of system boards, alix2d2, and their features shown in Table 1. We use the Linux-based x86 platform due to the extensible memory, radio adaptability via miniPCI cards, low power consumption, and low cost. Wireless routers have 2 Ethernet channels, 2 miniPCI slots which can use 802.11 a/n or 802.11 g/b wireless cards as wireless interfaces. The firmware of the wireless routers was replaced with the custom OpenWRT firmware [11], a system can be described as a very well-known embedded Linux distribution.

For the firmware we used the OpenWRT trunk r43753, which is based on Linux kernel version 3.14.26. This version of OpenWRT supports the OpenvSwitch 2.3. In these experiments, we use a custom POX controller, which is a controller based on NOX for rapid deployments of SDNs using Python as our OpenFlow controller. Because the interface of the wireless radio is limited, we use out-of-band control network with a wired connection.

5 Solution and Experiment

In this section we present our solution for load balancing in OpenFlow-based wireless mesh network, and the results of our experiment are implemented in our testbed. We make use of network measurement tool-iPerf for throughput and bandwidth measurements.

Table 1. Features of the system boards

	Hardware features
Interface	2 10/100/1000 Mbps LAN Ports 2 USB 2.0 Port 2 miniPCI wireless interface
	Wireless features
WLM200N2-26	2.4–2.4835 GHz
	23 dBm output power(per chain)/26 dBm(aggregate)
	802.11 b/g
WLM200N5-23 ESD	5.150–5.975 GHz
	23 dBm output power(per chain)/26 dBm(aggregate)
	802.11 a/n

We proposed a simple idea to solve the load imbalance problem in OpenFlow-based wireless mesh network in our paper. The setting of the wireless link of each mesh node is focused on in our solution. Figure 3 shows the basic topology of our network, including the basic elements of the wireless, wireless mesh nodes, and gateway nodes. Followed, we consider two load balancing scenarios: (1) the basic setup of the data flow paths, (2) data flow path redirection between links. The goal of this section is to provide load balancing configurations sample under different scenarios. Also, experiments are conducted to evaluate our OpenFlow-based wireless mesh network.

5.1 Basic Setup of the Data Flow Paths

Basic setup of the data flow paths is the first step before the load balancing. OpenFlow provides such a capability by sending flow table to establish the data path. Also, our controller will remotely connect to the switches, so that the controller can change the configuration in realtime. Figure 4 shows the schematic for setting up a data flow path.

Experimental Description. As shown in Fig. 4, both node *a* and node *b* are mesh nodes, and node *g* is a gateway with a connection to the Internet or another network.

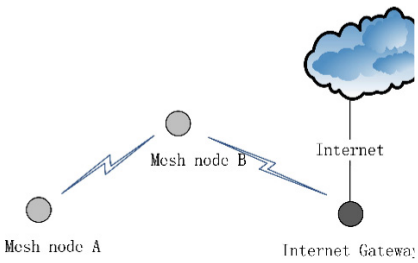


Fig. 3. Basic topology of WMN

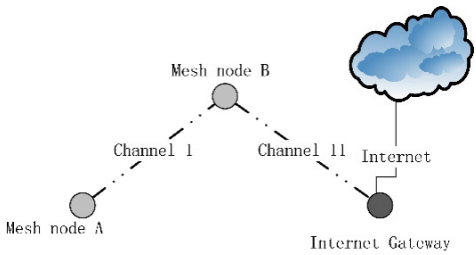


Fig. 4. Basic setup of data flow paths

All kinds of data servers are deployed in the Internet, from where the mesh networks request data. To setup a connection between node *a* and node *b* is our goal in this section. Assume node *b* already has a connection to node *g*. Now consider a scenario where a new node *a* joins the network. It needs to find its next-hop neighbor so that node *a* can communicate with the Internet. In this scenario, node *b* is the neighbor, and a wireless link using channel 1 represents the added flow path.

Experimental Procedure. To implement this scenario, traditional solutions require relatively high local computation capability. We show how to add the data path using OpenFlow controller in this section. As shown in Table 2, controller addressed this by sending flow tables and configuration instructions to mesh nodes, and no further action is required at local mesh routers. This setup servers as the foundation for the flow redirection.

When node *a* want to connect with node *b*, the controller will firstly establish an available wireless link by sending configuration instructions to them, setting the same channel and mesh_id. Thereafter, for node *a*, we identify the virtual OpenFlow switch as *id_node_a*, and the ingress port where data packets origination from node *a* labeled as *port_node_a*. When the OpenFlow switch receives data flow matching flow rules, the header of the packet will be manipulated in case of following the flow actions. In our scenario, the destination of node *a*'s packets is node *b*. Hence, packets form node *a* must modify their destination IP and MAC addresses. The *set_dst_ip* and *set_dst_mac* fields are used to rewrite packet headers, so the *node_b_ip* and *node_b_mac* are the IP and MAC addresses of node *b*, respectively. The modified packets must be output through the wireless radio interface, defined as *node_a_port*. The node *b*'s configuration is similar to node *a*, except the destination. According to the appropriate rules, the gateway will forward the packets which the IP of the destination is out of this network segment.

Flow tables are pushed by the controller, and after that the data path between node *a* and gateway *g* is established. In this scenario, the host communicates with the Internet with two hops link. The iPerf measurement tool shows the TCP throughput averaging at 4.04 Mbits between node *a* and the Internet. The measurement was maintained for 10 min and repeated five times.

Table 2. Flow tables and configuration instructions(basic setup of the data flow paths)

	Node A	Node B
Configuration Instructions	mesh channel: 1 meshid: MeshTrain	mesh channel: 1 meshid: MeshTrain
Flow rule	switch: id_node_a port: port_node_a	switch: id_node_b port: port_node_b
Flow actions (forward)	set-dst-ip: node_b_ip set-dst-mac: node_b_mac output: node_a_port	set-dst-ip: destination_ip set-dst-mac: destination_mac output: node_b_port

5.2 Data Flow Path Redirection Between Links

Redirecting data traffic between links is an essential way for implementing network load balancing. Figure 5 shows the schematic.

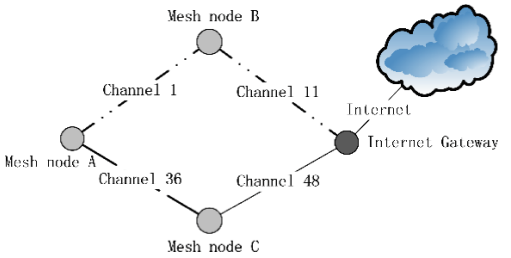


Fig. 5. Data flow path redirection between links

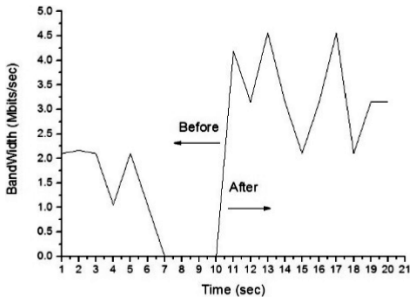


Fig. 6. Throughput before and after path redirection

Experimental Description. Node *a*, *b* and *c* are mesh nodes, and node *g* is an Internet gateway with a connection to the Internet. Assume a link is already established from node *a* and to gateway *g* via node *b*, as described in last section. The target of this section is to redirect data flows from node *a* to node *c* while node *b* experiences unexpected conditions or a new data packet arrives. This is a simple sample for load balancing.

Experimental Procedure. Flow tables and configuration instructions for this scenario are shown in Table 3. When we want to redirect a new data path, we must remove the old one. For node *a*, we remove the flow actions for *a-b* link, and send a new flow table for *a-c* by modified the destination IP and MAC addresses with node *c*'s(*node_c_ip* and *node_c_mac*). Node *c* modifies the packet headers to the destination.

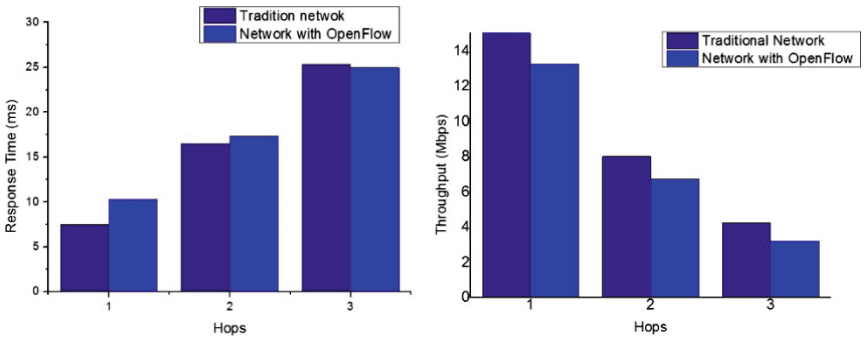
In our experiment, node *a* sends data to gateway *g* via node *b* before data flow redirection. Due to the reduction of the node *b*'s signal, the system decides to adjust the data link. The new path offers higher average throughput because we use the 802.11 a/n wireless radio card while the former uses 802.11 b/g wireless radio card. Figure 6 shows the TCP throughput performance results measured by iPerf before and after redirection. It can be seen that throughput increases after flows are redirected to the new path with stable wireless channels.

Table 3. Flow tables and configuration instructions(data flow path redirection between links)

	Node A	Node C
Configuration instructions	mesh channel: 36	mesh channel: 36
	meshid: MeshTrain	meshid: MeshTrain
Flow rules	switch: id_node_a	switch: id_node_c
	port: port_node_a	port: port_node_c
Flow actions (remove)	set-dst-ip: node_b_ip	
	set-dst-mac: node_b_mac	
	output:node_a_port	
Flow actions (forward)	set-dst-ip: node_c_ip	set-dst-ip: destination_ip
	set-dst-mac: node_c_mac	set-dst-mac: destination_mac
	output: node_a_port	output: node_c_port

5.3 Some Results Analysis

Since we use the OpenFlow controller change the connecting channel dynamically before the data transmitted. We need to consider the influence of the data path establishment. We measured the response time at each hop at least 100 times in the experiment. We compared the performance based-on our testbed with the traditional mesh network. It can be seen that the response time increase along with the increase of hops. As the Fig. 7(a) shown, there is only little difference between traditional networks and our OpenFlow-based networks. Figure 7(b) shows the TCP performance measured by iPerf at each hop. It can be seen that the throughput is about half of the prior hop's throughput. After three hops, traditional network averages at 4.26 Mbps throughput and the network with OpenFlow averages at 3.21 Mbps. Nowadays, people have not taken full advantage of the network bandwidth, we think the bandwidth consumption using centralization OpenFlow control is only a small amount of. Instead, the network performance will be improved, as well as improving the network bandwidth utilization. The communication among the control plane has influenced the throughput of the data plane. From the above, we confirmed that using OpenFlow in a wireless mesh network is reasonable and feasible.



(a) Response time vs. Hops (b) Throughput vs. Hops

Fig. 7. Performance results comparison

6 Conclusion and Future Work

In this paper, we described an OpenFlow-based wireless mesh network, which allows flexible control of OpenFlow-based switches, to solve the load balancing problem. Regarding the network characteristics, the traditional solutions in a typical wireless mesh network which requires compute-intensive routing algorithms running at each mesh node are abandoned. The architecture of the network presented in our paper combines a centralized OpenFlow controller and several OpenFlow-based mesh nodes (or the Internet gateway). When the monitor server in the controller finds the congestion over the network or a new arriving of the data flow, the controller will set up the data path including the channel of each link if necessary. Our experiments confirm that OpenFlow is an available technique to wireless mesh networks. As future work, we plan to develop an adaptive load-aware routing algorithm for multi-interface wireless mesh networks based-on OpenFlow, and plan to experimentally evaluate the approach.

Acknowledgment. This work was supported in part by the Industry-University-Research Combination Innovation Foundation of Jiangsu Province (No. BY2013003-03) and the Industry-University-Research Combination Innovation Foundation of Jiangsu Province (No. BY2013095-2-10).

References

1. Jagadeesan, N.A., Krishnamachari, B: Software-defined networking paradigms in wireless networks: a survey. ACM Comput. Surv. **47**(2), 11 p. (2014). Article 27, doi:10.1145/2655690
2. Alotaibi, E., Mukherjee, B.: A survey on routing algorithms for wireless ad-hoc and mesh networks. Comput. Netw. **56**(2), 940–965 (2012)
3. Hu, Y., Li, X.-Y., Chen, H.-M., Jia, X.-H.: Distributed call admission protocol for multi-channel multi-radio wireless networks. In: Global Telecommunications Conference, 2007, GLOBECOM 2007, pp. 2509–2513. IEEE 26–30 November 2007

4. Brzezinski, A., Zussman, G., Modiano, E.: Distributed throughput maximization in wireless mesh networks via pre-partitioning. IEEE/ACM Trans. Networking **16**(6), 1406–1419 (2008)
5. Detti, A., Pisa, C., Salsano, S., Blefari-Melazzi, N.: Wireless mesh software defined networks (wmSDN). In: 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), vol. 6983, pp. 89–95. IEEE (2013)
6. Chung, J., Gonzalez, G., Armuelles, I., Robles, T., Alcarria, R., Morales, A.: Characterizing the multimedia service capacity of wireless mesh networks for rural communities. In: 2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 628–635. IEEE (2012)
7. Open Networking Foundation. https://www.opennetworking.org/
8. Akyildiz, I.F., Wang, X.: A survey on wireless mesh networks. IEEE Commun. Mag. **43**(9), S23–S30 (2005)
9. Yang, F., Gondi, V., Hallstrom, J.O., Wang, K.C., Eidson, G.: OpenFlow-based load balancing for wireless mesh infrastructure. 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC), pp. 444–449. IEEE (2014)
10. Parulkar, G.M., Rexford, J., Turner, J.S., Mckeown, N., Anderson, T., Balakrishnan, H., et al.: Openflow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)
11. OpenWrt. https://openwrt.org/