OpenFlow-based Load Balancing for Wireless Mesh Infrastructure

Fan Yang*, Vamsi Gondi†‡, Jason O. Hallstrom†‡, Kuang-Ching Wang*, Gene Eidson§‡
*Department of Electrical and Computer Engineering, †School of Computing, ‡Institute of Computational Ecology,
§Biological Sciences
Clemson University, Clemson, SC
Email: {fany, vgondi, jasonoh, kwang, geidson}@clemson.edu

***Abstract*—Wireless mesh-based backhaul infrastructure is intended to provide reliable data transmission, with high throughput across large-scale networks. Load balancing is essential over a long period of operation to provide high throughput and uninterrupted service to end users. Load across mesh nodes is highly variable as the traffic depends on the number of clients connected to the nodes, as well as the services they use. Existing load balancing solutions are based on theoretical analysis and simulations. Most require distributed routing algorithms executed over compute-intensive routing nodes. It is also challenging to provide practical support for real-time traffic redirection using traditional mesh nodes. In this paper, we develop a prototype mesh infrastructure where flows from a source node can take multiple paths through the network. OpenFlow, an emerging technology that makes network switches programmable via a standard interface, allows flexible control of data flow paths. The Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N) mesh protocol is used to provide mesh topology and link quality information. The OpenFlow controller decides the best data path based on this information to ensure high throughput data transfer. To demonstrate the usefulness of our approach, we have implemented three test cases to enable data path setup and redirection with low complexity and overhead. Our test case measurements confirm that OpenFlow is a promising complementary technology to traditional mesh routing protocols for wireless networks.**

I. Introduction

Wireless mesh networking offers promise in meeting the challenges of backhaul infrastructure, as it provides flexibility while being cost-effective. In a mesh network, mesh routers collect information from local mesh clients and relay this information to Internet gateways for data reception and analysis at a remote server. A key design objective when designing a wireless mesh network is to provide high network throughput, as well as optimal load balancing. Many routing protocols use pre-selected nodes for relaying traffic, causing those nodes to be heavily used, while leaving other nodes idle. Load balancing is essential; it ensures that each node has approximately equal traffic to forward, so that optimal network performance is achieved throughout the network.

How to build optimal mesh networks with load balancing has been extensively studied theoretically [1][2][3][4]. However, two important conditions are assumed in existing work: 1) Routing between a given mesh node and Internet gateway can use multiple paths. 2) Data traffic between mesh routers and Internet gateways is divisible. These two assumptions are difficult to implement in traditional mobile routers and mesh nodes. There is a need for a new network protocol to manipulate real-time flow paths. Moreover, the system should discover the new mesh topology whenever a new node joins the network. The system should also be able to decide the best data path and redirect flows based on real-time link information. This requires network performance measurement tools to estimate real-time link statistics.

In this paper, we propose a new architecture and develop a prototype realization based on OpenFlow, a communication protocol that gives access to the forwarding plane of a network switch/router over the network [5][6]. We define a two-layer architecture, where nodes form a mesh network using B.A.T.M.A.N [7] as the first layer, and OpenFlow as the second layer. B.A.T.M.A.N is selected due to its reliability and layer 2 functionalities. The objective of this work is to present the OpenFlow-based infrastructure design and demonstrate load balancing test cases under practical application scenarios. System components, design procedures, and related performance analysis will be discussed.

The rest of the paper is organized as follows: A review of existing mesh protocols and OpenFlow is conducted in Section 2. The OpenFlow-based infrastructure design is presented in Section 3. Section 4 presents the load balancing test cases and related performance analysis. Conclusions and future work are presented in Section 5.

II. Background and Related Work

There has been significant prior work on load balancing strategies based on traditional wireless mesh networks. Most prior work focuses on distributed algorithms, where nodes communicate only with neighboring nodes [8][9][10]. As there is a central controller in OpenFlow networks, a centralized algorithm is possible, offering better performance and streamlined implementation compared to traditional distributed routing algorithms. Routing protocols for mesh networks can generally be divided into proactive routing, reactive routing, and hybrid routing strategies [11]. However, these protocols work in a distributed way, and most do not provide load balancing. An alternative is to exploit multiple channels by adopting multiple radios on a single mesh router [12][13]. While work in this area provides valuable insight on how to

978-1-4799-2355-7/14/$31.00 ©2014IEEE

design optimal routing protocols, their associated implementations and analysis are still based on theoretical calculations or simulations. They lack real-world demonstrations.

Among the limited number of OpenFlow-based mesh networks, [14] demonstrates simplified management of client mobility and handover performance. However, an analysis of load balancing is lacking. Load balancing using both wireless mesh network protocols and OpenFlow is discussed in [15]. The experiments demonstrate the improved performance of OpenFlow over traditional mesh routing protocols. However, the study assumes one-hop links and does not consider link quality between the mesh nodes. This approach is not able to identify next-hop nodes or available gateways when routing traffic. Also, the study doesn't adequately consider switching data and control paths to route flows within the network. Finally, the rules for establishing data paths in OpenFlow-based networks have not been carefully studied in the literature.

In summary, the challenges of designing load balancing protocols for mesh backhaul infrastructure include: 1) Most mesh routers are equipped with only one radio, preluding the feasibility of multi-radio load balancing protocols. 2) Existing routing and scheduling strategies require perfect multi-path packet traffic, which is not feasible for current routers and switches. 3) Existing protocols involve complex mathematical models that demand high computational capability at local nodes. Consequently, there is a need for building a bridge to connect theoretical protocols and practical deployments, which is the main contribution of this study. With the help of OpenFlow, multi-path packet forwarding becomes feasible by inserting customized flow rules. 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.

To the best of our knowledge, our work is the first study load balancing in wireless mesh network using OpenFlow, where one or more Internet gateways are present.

III. An OpenFlow-based Wireless Mesh Network

A. Wireless Mesh Network

A backbone mesh network is formed by mesh nodes communicating with each other in a multi-path and multi-hop fashion via wireless links. Data from mesh clients is relayed by mesh routers, and network coverage is adjusted by controlling the number of mesh routers. The mesh backbone can be built using various radio technologies, with IEEE 802.11 being the dominant standard for most applications [16][17]. Compared with other technology, mesh routers adopting the IEEE 802.11 standard can achieve the same coverage with lower power consumption and simplified deployment. The Internet gateway is the Internet access device, which routes all information packets to a remote server.

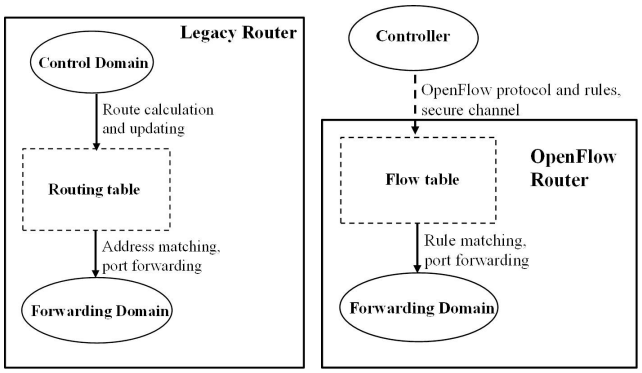
Due to variations in link quality and nodes joining and leaving the network, the network topology changes more frequently than traditional wired networks. Autonomous topology and network change discovery is needed for self-configuration of the wireless mesh network. Point-to-point links also need to be adapted. Therefore, a special mesh routing protocol for node discovery and link identification is required. Such mesh routing protocols include Optimized Link State Routing (OLSR) [18], Dynamic Source Routing (DSR) [19], Ad-Hoc On-Demand Distance Vector Routing (AODV) [20], and Better Approach to Mobile Ad-Hoc Networking (B.A.T.M.A.N). In this study, B.A.T.M.A.N is adopted due to its reliable functionality, high performance, and ability to operate in layer 2. Traditionally, mesh protocols operate at layer 3, communicating routing information through UDP, whereas B.A.T.M.A.N operates at layer 2, transporting routing information and data through Ethernet frames. B.A.T.M.A.N handles data transport by encapsulating packets from nodes to gateways. The main uses of B.A.T.M.A.N in this study are to identify next-hop nodes, maintain node topology, and provide local mesh topology knowledge when establishing data paths. The Internet gateways in the mesh network provide global mesh topology information through an API to a remote server.

B. OpenFlow

OpenFlow is a new switching protocol based on the concept of software defined networking (SDN). OpenFlow-enabled switches/routers move packet forwarding intelligence to a server, while keeping the routers/switches simple. In legacy switch/router structures, there is a control domain for making packet forwarding decisions between different ports, and a forwarding domain where the actual packet forwarding happens, based on the forwarding/routing tables. As there is no control domain residing at an OpenFlow-enabled switch/router, the forwarding domain can be kept simple, containing simple rules for traffic flow processing. The functionality of the control domain is now moved to a network control server, referred to as an OpenFlow controller [6]. An OpenFlow controller can intercept and manipulate packet headers when incoming traffic matches preset rules, and transfer those packets to the port(s) as needed [21]. 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. Moreover, as the OpenFlow controller monitors the flow properties of packet traffic, many interesting network services can be implemented efficiently. Figure 1 shows a structural comparison between legacy and OpenFlow-enabled routers.

C. OpenFlow-Based Remote Monitoring Infrastructure

There are generally four components in a typical mesh-based wireless infrastructure: mesh clients, mesh routers, Internet gateways, and remote servers. In this paper, each mesh node (or Internet gateway) is OpenFlow-enabled and responsible for mesh connectivity and traffic routing. Each node is split into two virtual planes, a data plane and a control plane. The control plane no longer resides on the switch, but resides in a remote server that runs as a network-wide controller. The data plane is abstracted as a set of rules, which form a flow table for flow-processing. As shown in Fig. 1, virtualization is achieved by installing OpenFlow switch software at each



(a) Legacy Router (b) OpenFlow Router

Fig. 1. Structural Comparison Between Legacy Router and OpenFlow Router

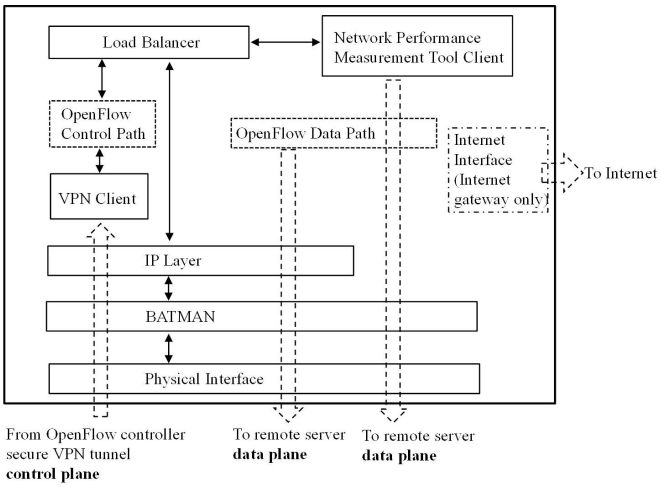


Fig. 2. Architecture of an OpenFlow-Enabled Mesh Router/Internet Gateway

device [22]. Data traffic relies on the OpenFlow data plane to reach remote servers through data interfaces. Normally, the control and data interfaces of a single mesh router are two different physical interfaces. Control traffic is usually referred to as out-of-band traffic because data traffic does not share the same path with control traffic. In this paper, however, there is only one physical interface in each mesh router, for multi-hop connectivity and traffic flow. Each Internet gateway has two interfaces, one dedicated connecting to the Internet. A secure virtual private network (VPN) tunnel is created as the control path for each mesh node and Internet gateway. Sockets are used by data path components to communicate locally with control path components.

OpenFlow defines a secure channel between the data plane and network controller using a set of rules, properties, an expiration time and a list of actions as summarized in Fig. 2. The properties specify packet source, original header values, and switch ports. The packets from nodes are encapsulated and forwarded to the network controller when a packet arrives at the switch network interface but no matching rule is found. Otherwise, the actions allow packet headers to be manipulated

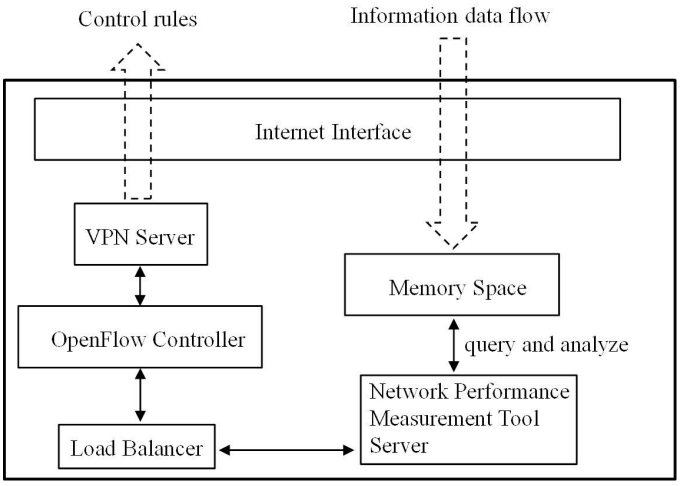


Fig. 3. Architecture of the Remote Server

and transferred to output packets for a particular switch port based on the rules. The network controller runs applications that are aware of the data flow a packet belongs to, as well as the flow properties through the control plane path, while determining the route for packets through the data plane path. In this paper, B.A.T.M.A.N is implemented on top of the physical layer to support node discovery and multi-hop connectivity among the virtual control interfaces. Load balancing is implemented on top of the IP layer with the help of the OpenFlow controller, by redirecting traffic flows along determined routes. Connectivity to the remote server is achieved using one or more Internet gateways. A network performance measurement client resides at each mesh router and Internet gateway. Information needed for load balancing, such as bandwidth and channel utilization, can be queried from the measurement clients.

As shown in Fig. 3, the architecture of the remote server consists of a network performance measurement tool server, an OpenFlow controller, a VPN server, and a memory space for data reception. The performance server queries information from mesh routers and Internet gateways and builds a datastore used to support the load balancing process. The database also contains a network graph built from connectivity information provided by the discovery protocol. The main purpose of the OpenFlow controller is to perform basic load balancing tasks, such as redirecting traffic flows and managing network addresses. The controller relies on the database to control traffic based on (soft) real-time routing information. Network topology changes are sent to the OpenFlow controller, which then updates routes based on the new network graphs. When a new node joins the network, changes in the database trigger recalculation of the load balancing process, and new flow rules are installed accordingly. The VPN server is used to provide secure tunnels for the OpenFlow control plane.

We have realized the OpenFlow-enabled mesh router using a Linux-based x86 platform due to its extensible memory, radio adaptability via PCI cards, (relatively) low power con-

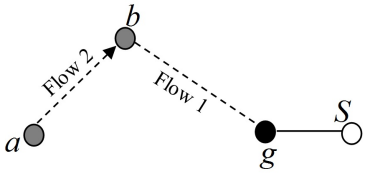
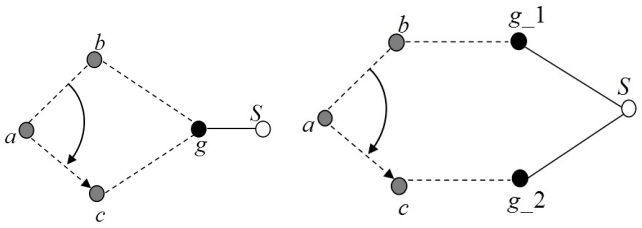


Fig. 4. Basic Setup of Data Flow Path



(a) Between Mesh Nodes (b) Between Internet Gateways

Fig. 5. Redirecting Data Traffic in the Mesh Network

sumption, and low cost. The mesh routers are built on the Debian (6.0) [23] operating system. We use the OpenFlow reference implementation, Floodlight [22] [24], as the network controller due to its easy setup and high performance. Wireless communication between individual nodes is based on the IEEE 802.11 standard. To support broader communication ranges, 900 Mhz wireless radio cards are used. The OpenFlow protocol is used for setting up the flow tables. The initial network topology and routing is setup using B.A.T.M.A.N. Network monitoring is performed using a custom-designed measurement tool.

IV. Load Balancing Test Cases

In this section, we consider three load balancing scenarios: 1) basic setup of data flow paths, 2) data flow redirection between mesh nodes, and 3) data flow redirection between Internet gateways. The goal of this section is to provide sample flow entry configurations under different scenarios. Experiments are conducted to evaluate the effectiveness of the OpenFlow-based wireless mesh network.

A. Basic setup of data flow paths

Basic setup of the data flow paths is the first step before enabling redirection of data traffic among nodes. OpenFlow provides such a capability by inserting flow entries at the controller. For nodes connected to the controller, each behaves as a virtual switch. Any incoming traffic that matches the flow rules will be redirected using the flow actions. Figure 4 shows the schematic for setting up a data flow path between two nodes.

1) Experimental setup: In Fig. 4, nodes *a* and *b* are mesh nodes, and node *g* is an Internet gateway with a connection to remote server *S*. In this section, our goal is to setup a connection between node *a* and remote server *S*. Assume node *b* already has a connection to node *g*. Any packet originating from node *b* arrives at node *g* first, and then it is transported to remote server *S*. Flow 1 in Fig. 4 shows the original flow path without OpenFlow. Now consider a scenario where a new node *a* joins the network. It needs to find its next-hop neighbor so that packets originating from node *a* can be transported to the remote server. In this setup, node *b* is the neighbor, and Flow 2 represents the added flow path.

2) Experimental procedure: To implement this scenario, traditional solutions require coordination between nodes *a* and *b*, as well as routing table updates, requiring relatively high local computation capability at each physical node. We show how to add the Flow 1 data path using OpenFlow in this section. OpenFlow addresses this by inserting flow entries at the controller, as shown in Table 1. No further action is required at local mesh routers.

For node *a*, its virtual OpenFlow switch identifier is *data_path_id_node_a*, and packets originating from node *a* have the ingress port number labeled as *virtual_port_node_a*. When the virtual OpenFlow switch receives data flow matching the flow rules, packet headers will be manipulated to follow the flow actions. In this scenario, the destination of node *a*'s packets is node *b*. Hence, packets from node *a* must have their destination IP and MAC addresses rewritten to node *b*. The set-dst-ip and set-dst-mac fields in the flow actions are used to rewrite packet headers. *ip_node_b* and *mac_node_b* are the IP and MAC addresses of node *b*, respectively. The modified packets must be output through the wireless radio interface, defined as *physical_port_node_a* in the flow entries. After node b receives the matching flows, flow actions take effect, and it forwards redirected data traffic to gateway *g* via the original link. As the virtual OpenFlow switch takes over node *b* as well, corresponding flow entries need to be set. The destination IP and MAC addresses must be set to the remote server (*ip_node_S* and *mac_node_S*), and the data flow must be output through node *b*'s wireless radio interface (*physical_port_node_b*).

Flow entries are pushed by the controller via a Floodlight module, the Static Flow Pusher API, which allows a user to manually insert flows into an OpenFlow network. After flow entries are pushed, the connection between node *a* and gateway *g* is established. The iPerf measurement tool shows the TCP throughput averaging at 3.24 Mbps between node *a* and remote server *S*. The measurement was maintained for 1 minute and repeated five times. This scenario serves as the foundation for more advanced flow redirection test cases.

B. Data flow redirection between mesh nodes

Redirecting data traffic between mesh nodes is essential for implementing network load balancing. Figure 5 (a) shows the corresponding schematic.

1) Experimental setup: Nodes *a, b,* and *c* are mesh nodes, and node *g* is an Internet gateway with a connection to remote server *S*. Assume a link is already established from node *a* to gateway *g* via node *b* by pushing the flow entries shown in Table 1. If node *b* experiences unexpected conditions, e.g. significant packet losses due to a full buffer or unstable

TABLE I
FLOW ENTRIES (BASIC SETUP OF DATA FLOW PATHS)

Flow Rules	'switch':*"data_path_id_node_a"* 'ingress-port':*"virtual_port_node_a"*	'switch':*"data_path_id_node_b"* 'ingress-port':*"physical_port_node_b"*
Flow Actions (push)	set-dst-ip = *ip_node_b* set-dst-mac = *mac_node_b* output = *physical_port_node_a*	set-dst-ip = *ip_node_S* set-dst-mac = *mac_node_S* output = *physical_port_node_b*

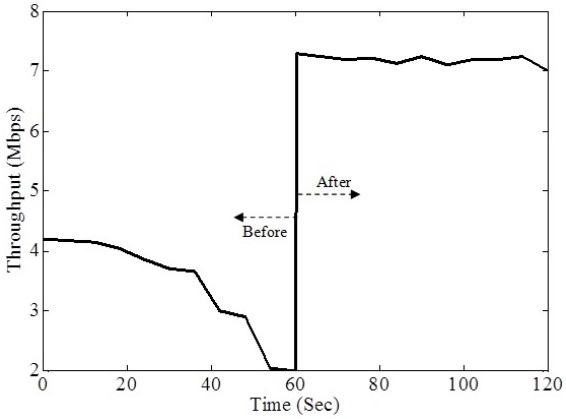
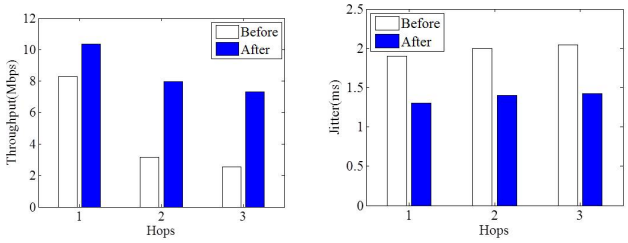


Fig. 6. Throughput Comparison Before and After Flow Redirection



(a) Throughput vs. Hops (b) Jitter vs. Hops

Fig. 7. Transmission Performance Results Before and After Redirection

wireless channel, node *a* will redirect its data flow to node *c* to avoid the degraded performance. This is a simple application model for load balancing, and the OpenFlow controller is able to make the adjustments based on reports from the installed performance measurement tool. In this section, our goal is to delete an existing connection from node *a* to server *S* via node *b* and set up a new connection to server *S* via node *c*.

2) Experimental procedure: Flow entries required for this scenario are shown in Table 2. The previous flow actions for the *a-b-g* link must be removed before the new flow actions for the *a-c-g* link are pushed. For node *a*, the destination IP and MAC addresses must be modified to node *c* (*ip_node_c* and *mac_node_c*). Node *c* receives incoming data flow from its wireless interface *physical_port_node_c*, with matching virtual OpenFlow identification number *data_path_id_node_c*. It then modifies the destination IP and MAC addresses of packet headers to be the remote server *S* (*ip_node_S* and *mac_node_S*).

In our testbed, node *a* sends data to gateway *g* via node *b* before data flow redirection. Due to a full buffer, node *b* starts dropping packets and throughput is degraded. After the new flow entries are pushed, data flow begins reaching gateway *g* via node *c* as the relay node. The latter case (after redirection) should offer higher average throughput, as we configured node *c* with a larger buffer and a less noisy channel.

Figure 6 shows the TCP throughput performance results (from node *a* to remote server *S*) measured by iPerf for one minute before and after new flow entries are pushed. Load traffic is balanced among the two wireless links after switching the data path – traffic flowing through low bandwidth links is redirected to high bandwidth links. The redirection delay was measured by checking packet timestamps captured by Wireshark [26] – less than 3 ms. The redirection delay in this test scenario is mainly the execution-time of the flow-remove and flow-push operations.

C. Data flow redirection between Internet gateways

In a wireless mesh network, the Internet gateway is the only interface between local mesh routers and remote servers. Therefore, the Internet gateway usually experiences high data flow burdens. It is appropriate to place multiple Internet gateways in a network, if possible, so that traffic load can be balanced among them. Figure 5(b) shows a setup with two Internet gateways (*g_1* and *g_2*), where data traffic will be redirected between them.

1) Experimental setup: Nodes *a, b,* and *c* are mesh nodes, and nodes *g_1* and *g_2* are Internet gateways with connections to remote server *S*. Assume a link is already established from node *a* to remote server *S* via node *b* and *g_1*. In this scenario, we manually make wireless channel conditions unstable for the *a-b-g_1-S* links and keep the *a-c-g_2-S* links stable. The OpenFlow controller must be able to make the adjustments based on reports from the installed performance measurement tool. Our goal is to delete an existing connection from node *a* to server *S* via nodes *b* and *g_1*, and set up a new connection to server *S* via nodes *c* and *g_2*.

2) Experimental procedure: Flow entries for node *a* are identical to those inserted in the second test scenario (Table 2). Controller needs to remove the original flow entries to node *b* and push new flow entries to node *c*. Nodes *b* and *c* must be configured with the gateway as the destination. Hence, flow entries for *b* to *g_1*, and *c* to *g_2* are required for nodes *b* and *c*, respectively. At Internet gateways *g_1* and *g_2*, destination IP addresses in their flow entries need to be set to the IP address of the remote server. These flow entries can be set similarly to those in Table 1. As shown in Fig. 5 (b), nodes

TABLE II
FLOW ENTRIES (DATA FLOW REDIRECTION BETWEEN MESH NODES)

Flow Rules	'switch':*"data_path_id_node_a"* 'ingress-port':*"virtual_port_node_a"*	'switch':*"data_path_id_node_c"* 'ingress-port': *"physical_port_node_c"*
Flow Actions (remove)	set-dst-ip = *ip_node_b* set-dst-mac = *mac_node_b* output = *physical_port_node_a*	
Flow Actions (push)	set-dst-ip = *ip_node_c* set-dst-mac = *mac_node_c* output = *physical_port_node_a*	set-dst-ip = *ip_node_S* set-dst-mac = *mac_node_S* output = *physical_port_node_c*

at the first hop are *b* and *c*, nodes at the second hop are *g_1* and *g_2*, and the node at the third hop is *S*. Data transmission performance at each hop before and after flows are redirected is measured and compared. The results are presented in Figure 7.

Figure 7 (a) shows the TCP throughput performance measured by iPerf at each hop. It can be seen that throughput increases after flows are redirected to the stable wireless channel. The increase percentage gets larger when data transmission is at the second and third hop. This is because throughput degrades to a low number when the wireless links are unstable, before the flows are redirected. It is also worth noting that throughput decreases as the number of hops increases. This is due to a higher probability of packet loss when packets need to go through more wireless links. Figure 7 (b) shows the jitter performance measured by iPerf using UDP at each hop. Jitter performance does not change significantly at different hops. However, after the flows are redirected, jitter drops nearly 25% due to the stable wireless channels. Moreover, Wireshark measurements show a less than 5 msec redirection delay. Our results successfully demonstrate the use case where redirection of flows is needed when there are multiple Internet gateways in the network.

V. CONCLUSION AND FUTURE WORK

In this paper, we described a new OpenFlow-based wireless infrastructure for remote monitoring applications, which allows flexible control of data flow paths. With regard to network load balancing, traditional solutions are typically based on theoretical analysis and simulation and require compute-intensive routing algorithms running at each mesh node. The architecture presented in this paper combines a centralized OpenFlow controller and a performance monitoring tool to enable real-time data flow redirection. Whenever the monitoring tool detects performance degradation due to over-burdened nodes, the controller pushes corresponding flow entries to redirect traffic to other nodes with less workload. Our test case measurements confirm that OpenFlow is a useful complementary technology to traditional mesh routing protocols. As future work, we plan to develop an algorithm for adaptive load balancing based on network performance, and plan to experimentally evaluate the approach.

ACKNOWLEDGMENTS

This work was supported through awards from the NSF (CNS-1126344, CNS-0745846) and the City of Aiken.

REFERENCES

[1] V.C. Gungor and F. Lambert, "A Survey on Communication Networks for Electric System Automation", Computer Networks Journal (Elsevier), vol. 50, pp. 877-897, May 2006
[2] K. Jain et al., "Impact of interference on multi-hop wireless network performance", in Proc. of ACM MOBICOM '03, San Diego, CA, 2003
[3] C. Chereddi, P. Kyasanur, and N. H. Vaidya, "Design and implementation of a multi-channel multi-interface network", in Proc. of Realman, 2006
[4] H. Pathak and R. Dutta, "A survey of network design problems and joint design approaches in wireless mesh networks", IEEE Commun. Surveys Tuts., vol. 13, no. 3, pp. 1-33, June 2011.
[5] OpenFlow. Available online at: www.en.wikipedia.org/wiki/OpenFlow. (accessed on 30^{th} August 2013).
[6] Open Networking Foundation. Available online at: www.OpenFlow.org. (accessed on 30^{th} August 2013).
[7] D. Johnson, N. Ntlatlapa, and C. Aichele, "Simple pragmatic approachto mesh routing using B.A.T.M.A.N", in 2nd IFIP International Symposiumon Wireless Communications and Information Technology in DevelopingCountries, CSIR, Pretoria, South Africa, 6-7 October 2008.
[8] H. Cheng et al., "Distributed scheduling algorithms for channel access in TDMA wireless mesh network", The Journal of Supercomputing, v.45 n.1, p.105-128, July 2008.
[9] Y. Hu et al., "Distributed call admission protocol for multi-channel multi-radio wireless networks", Proc. IEEE Globecom, 2007.
[10] Brzezinski, A., Gil Zussman, A., Modiano,E., "Distributed Throughput Maximization in Wireless Mesh Networks via Pre-Partitioning", IEEE Transaction on Networking, VOL. 16, NO. 6, DECEMBER 2008
[11] S. Rao, Y.K Sundara Krishna, K NageswaraRao., "A Survey: Routing Protocols for Wireless Mesh Networks", International Journal of Research and Reviews in Wireless Sensor Networks (IJRRWSN), 2011.
[12] Kyasanur P, Vaidya NH, "Capacity of multi-channel wireless networks: impact of number of channels and interfaces", MobiCom 05, ACM Press, New York, NY, USA, pp 4357
[13] F. Li et al., "Gateway placement for throughput optimization in wireless mesh networks", Mobile Network Application (2008), Pages: 198-211
[14] P. Dely et al., "Open flow for wireless mesh networks", Computer Communications and Networks (ICCCN), pages 1-6. IEEE, 2011
[15] Joaquin Chung et al., "Characterizing the Multimedia Service Capacity of Wireless Mesh Networks for Rural Communities", CNBuB, 2012
[16] R. Prasad, H. Wu, "Gateway deployment optimization in cellular Wi-Fi mesh networks", Journal of Networks, vol 3, 2006.
[17] C. Bemmoussat, F. Didi, M. Feham, "Efficient Routing Protocol to Support QOS in Wireless Mesh Network", International Journal of Wireless & Mobile Networks, 4(5), 2012
[18] P. Jacquet et al., "Optimized link state routing protocol for ad hoc networks", IEEE INMIC 2001, pp. 62-68, 2001.
[19] D.B. Johnson and D.A. Maltz, "Dynamic source routing in ad hocwireless networks", Mobile Computing, vol. 353, pp. 153-181, 1996.
[20] C.E. Perkins and E.M. Royer, "Ad-hoc on-demand distance vectorrouting", 2nd IEEE Workshop on Mobile Computing Systems and Applications, vol. 2, pp. 90-100, 1999.
[21] N. McKeown et al., "Openow: enabling innovation in campus networks", SIGCOMM, vol. 38, pp. 69-74, March 2008
[22] OpenFlow Consortium, Available online at: www.openowswitch.org. (accessed on 30^{th} August 2013).
[23] Debian OS, Available online at: www.debian.org. (accessed on 30^{th} August 2013).
[24] Floodlight, Available online at: www.floodlight.OpenFlowhub.org. (accessed on 30^{th} August 2013).