EAI Endorsed Transactions
on Wireless Spectrum

Research Article **ICST.ORG**

Towards Effective Intra-flow Network Coding in Software Defined Wireless Mesh Networks

Donghai Zhu[1], Xinyu Yang[1], Peng Zhao[1,*], and Wei Yu[2]

[1]Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China
[2]Department of Computer and Information Sciences, Towson University, Towson, MD, USA

Abstract

Wireless Mesh Networks (WMNs) have potential to provide convenient broadband wireless Internet access to mobile users. With the support of Software-Defined Networking (SDN) paradigm that separates control plane and data plane, WMNs can be easily deployed and managed. In addition, by exploiting the broadcast nature of the wireless medium and the spatial diversity of multi-hop wireless networks, intra-flow network coding has shown a greater benefit in comparison with traditional routing paradigms in data transmission for WMNs. In this paper, we develop a novel OpenCoding protocol, which combines the SDN technique with intra-flow network coding for WMNs. Our developed protocol can simplify the deployment and management of the network and improve network performance. In OpenCoding, a controller that works on the control plane makes routing decisions for mesh routers and the hop-by-hop forwarding function is replaced by network coding functions in data plane. We analyze the overhead of OpenCoding. Through a simulation study, we show the effectiveness of the OpenCoding protocol in comparison with existing schemes. Our data shows that OpenCoding outperforms both traditional routing and intra-flow network coding schemes.

Received on 16 October 2015; accepted on 15 December 2015; published on 19 January 2016

Keywords: wireless mesh network, software defined networking, intra-flow network coding

doi:10.4108/eai.19-1-2016.150998

1. Introduction

Wireless mesh networks (WMNs) [1] have been considered a promising solution for providing convenient broadband wireless Internet access to mobile clients (e.g., smartphones and notebooks). In a WMN, stationary wireless mesh routers can establish the backbone for mobile clients, controlling and monitoring traffic flows in the network and managing routing paths according to algorithms and protocols implemented in routers. After being connected to one of mesh routers, a mobile client can communicate with other mobile clients in the same WMN or can get access to the Internet via the hop-by-hop forwarding way.

Nonetheless, because of the heterogeneous infrastructure of WMNs, deploying and managing WMNs is a challenging problem. The network administrator will have to setup and maintain each mesh router individually as network devices are vendor-specific. To address the issue of network deployment and management, Software-Defined Networking (SDN) [2] has been proposed. The essential idea of SDN is to exploit the ability of decoupling the data plane and the control plane in routers or switches, and keeping only data forwarding functions in network devices. By sending configuration commands from the control plane that consists of one or multiple controllers down to the data plane, the network administrator then has the ability to control and configure the data plane globally rather than configuring each device individually. As we can see, by using SDN, the network can be easily deployed and managed. In addition, with the view of the entire network, the network performance can be improved by deploying global optimization strategies through a powerful control plane [3]. For example, existing research efforts [4–6] have shown the great benefit of integrating SDN with WMNs.

[★]Please ensure that you use the most up to date class file, available from EAI at http://doc.eai.eu/publications/transactions/latex/

[*]Corresponding author. Email: p.zhao@mail.xjtu.edu.cn

Meanwhile, another challenge in establishing WMNs is to achieve a high network performance despite the unreliable and lossy wireless communication channels. The inherent broadcast nature of wireless communication channels can be exploited to improve network performance. Network coding [7] is one mechanism, which makes use of the broadcast nature in wireless networks. For example, by integrating intra-flow random linear network coding that enables relay nodes to combine information content in packets before forwarding and performing opportunistic routing [8], the MORE protocol [9] has demonstrated that network coding can improve network throughput significantly and can be extended to support applications, including data distribution in peer-to-peer networks [10], etc.

As we can see, the SDN technology and network coding scheme are two different techniques to address different challenges in WMNs. Again, the key idea of SDN is to keep routers or switches simple, while moving the control function into the control server deployed in the network. The intra-flow network coding scheme can achieve much better performance than hop-by-hop forwarding schemes because of leveraging the nature of the wireless broadcast medium. Therefore, one research problem is raised: *How can we integrate the SDN and intra-flow network coding schemes to make that the integrated scheme performs better than the individual ones?*

To answer this question, in this paper we make the following contributions.

- We designed the OpenCoding protocol to address the challenge of easily deploying WMNs and further improving the performance of WMNs. The key idea of OpenCoding is to use SDN in the network where the intra-flow network coding is deployed. That is, a controller that works on the control plane makes routing decisions for mesh routers and the hop-by-hop forwarding function is replaced by the network coding function in data plane. By decoupling control plane and data plane, mesh routers can be simple, leading to a much easier ability to manage a network. More importantly, through the aid of viewing the entire network, a global optimization strategy can also be easily applied. Based on this idea, we developed a novel scheduling scheme for intra-flow network coding, which can achieve a high network performance gain, as well as the fairness of multiple network flows.

- We conducted extensive simulations to demonstrate the effectiveness of OpenCoding in comparison with existing schemes. Our evaluation data shows that our proposed OpenCoding outperforms OLSR [11], OpenFlow [12], and MORE [9] protocols. On average, OpenCoding achieves 68.32%, 12.46% and 9.88% gain over OLSR, OpenFlow and MORE in terms of data transmission throughput, respectively. Meanwhile, the throughput of OpenCoding drops little when the number of concurrent flows increases. The results also show that OpenCoding outperforms OLSR, OpenFlow and MORE protocols in terms of end-to-end delay. To our best knowledge, there has been no previous work, which integrates the SDN technique into the intra-flow network coding in WMNs and evaluates performance gain.

An earlier version of this work was published in [13]. Based on the conference version, we have made substantial extensions and revisions. The rest of the paper is organized as follows. In Section 2, we give the background and review existing research efforts that are closely relevant to our research. In Section 3, we present our proposed OpenCoding scheme in detail, including the architecture, the detailed design, and how to achieve fairness optimization. In Section 4, we analyze the overhead of OpenCoding. In Section 5, we show the performance evaluation of OpenCoding in comparison with several representative schemes. In Section 6, we discuss some open issues related to OpenCoding. Finally, we conclude the paper in Section 7.

2. Background and Related Work

In this section, we give the background and literature review of SDN and OpenFlow, as well as intra-flow network coding.

2.1. Wireless Mesh Networks

Wireless Mesh Networks (WMNs) are developed for a new broadband Internet access technology with growing interests in both research and industry [1]. Because of highly self-organized and self-configured features of WMNs, they are used to extend last-mile access to the Internet and have become a promising technology for providing easy wireless access for mobile users.

The infrastructure of WMNs is shown in Figure 1. As we can see from the figure, WMNs consist of two types of nodes: mesh routers and mesh clients. Mesh routers are almost stationary and establish the mesh backbone for mesh clients, thus there are no power limits on mesh routers. Mesh routers can equip multiple wireless interfaces and serve as the Internet gateway or bridge and integrate heterogeneous networks, including both wired and wireless networks. To this end, the deployment and management of such a heterogeneous infrastructure will be a big challenge. Meanwhile, because of the unreliable and lossy wireless communication channels of WMNs, another challenge in establishing WMNs is to achieve a high enough network throughput performance.

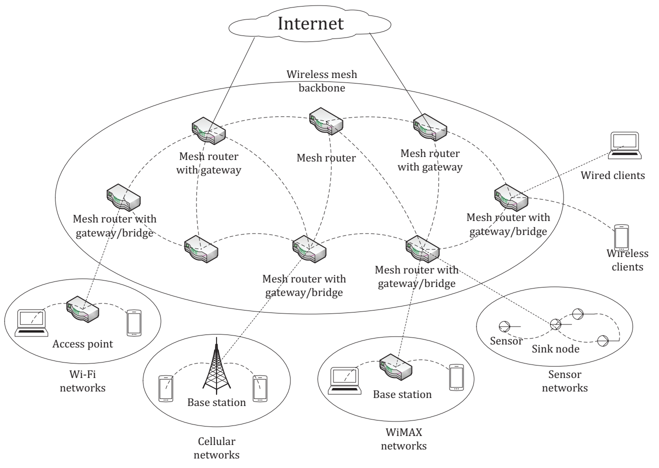


Figure 1. Infrastructure/backbone WMNs

With the rapid development of radio techniques, a number of approaches have been proposed in order to increase the capacity and flexibility of WMNs. Typical examples include cognitive radios [14], cooperative communications [15], Multi-Input Multi-Output (MIMO) systems [16] and Multi-radio Multi-channel (MRMC) systems [17]. On the other hand, by exploiting the broadcast nature of wireless medium, innovative transmission schemes, including Opportunistic Routing (OR) [8] and network coding [7], have also been developed to increase the performance of WMNs.

2.2. SDN and OpenFlow

Generally speaking, SDN is an emerging network architecture, which can decouple control plane and data plane, with goals of achieving a higher transmission speed, greater scalability, and greater flexibility. The key idea of SDN is to allow software developers and network administrators to allocate and manage the network resources in an easy way via an open interface. Its concept was originally developed by Nicira Networks based on their earlier development at UCB, Stanford, CMU, Princeton [2]. Figure 2 illustrates the standard architecture of SDN, which is defined by Open Networking Foundation (ONF) [18]. As we can see from the figure, the control plane can send commands down to the data planes of the hardware (e.g., routers or switches) [19]. The hardware in the data plane can only simply deliver data among them by checking the flow tables, which are distributed by the controller in the control panel. In this way, the hardware devices can be greatly simplified.

One of well-known SDN protocol standards is the OpenFlow [12], which was originally proposed for wired networks in order to make Internet switches intelligent and programmable through a standardized interface. OpenFlow allows network devices from many different companies to utilize the abstraction of the control planes and data planes [20–22], thus OpenFlow receives a considerable amount of industry attention. Besides OpenFlow, there are other SDN implementations and standards, such as IEEE P1520 standards [23], ForCES [24] and SoftRouter [25]. Also, there are a number of earlier work related to programming networks and active networks [26, 27].

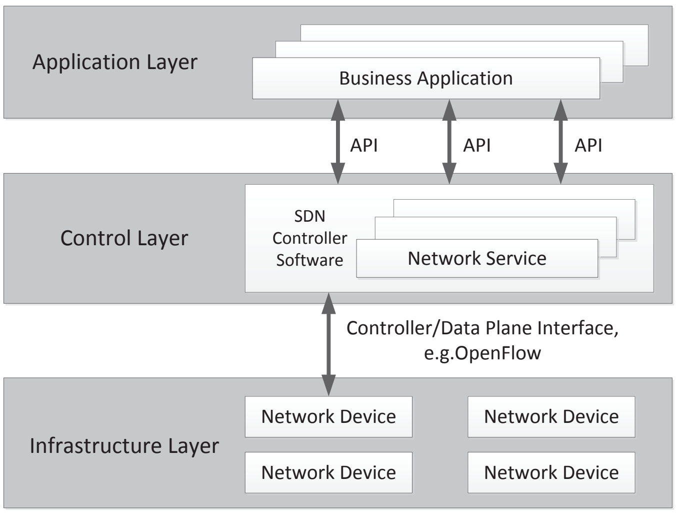


Figure 2. SDN architecture defined by ONF[18]

There have also been several research efforts on applying SDN/OpenFlow paradigm in wireless mobile networks. For example, Figure 3 shows an generic software defined wireless network-based architecture of a mobile network operator [28]. In Figure 3, we can see that SDN technology can be applied to various types of wireless networks. In this paper, we focus on applying SDN technology to wireless mesh networks [1]. In [4], an architecture that allows the flexible and efficient use of OpenFlow in WMNs was demonstrated. As the first attempt to apply OpenFlow in WMNs, the evaluation of this proposed architecture on a real test-bed called KAUMesh showed that the SDN architecture could achieve better performance than traditional schemes in terms of transmission throughout, control traffic overhead, and rule activation time. Riggio *et al.* in [29] proposed a set of high-level programming abstractions for WiFi networks, which enables new features and services to be implemented as software modules.

In order to obtain the link information and controlling link behavior, the OpenFlow protocol extensions were proposed to allow controllers to remotely control and manage wireless links through a set of Media-Independent Management (MIM) mechanisms [30]. In addition, one OpenFlow-based

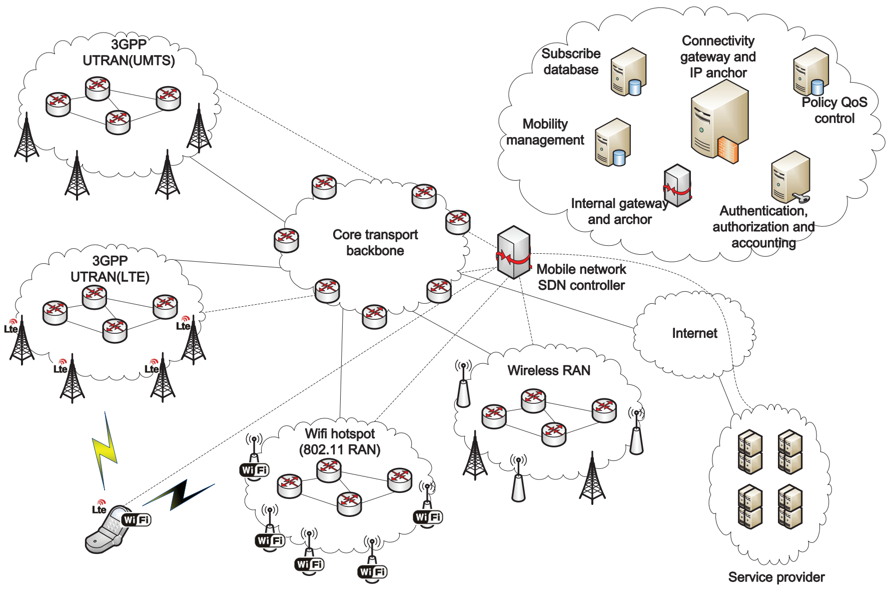

3GPP UTRAN(UMTS)
3GPP UTRAN(LTE)
Lte
Core transport backbone
Subscribe database
Connectivity gateway and IP anchor
Policy QoS control
Mobility management
Internal gateway and anchor
Authentication, authorization and accounting
Mobile network SDN controller
Internet
Wireless RAN
Wifi hotspot (802.11 RAN)
WiFi
Service provider

Figure 3. SDN-based wireless mobile network architecture

communication protocol was proposed to provide the access to the data plane of network routers [31], which can be used to balance traffic loads in the network. These existing research efforts confirm that OpenFlow can be a useful technique to improve the performance of WMNs.

There have been other research efforts that show the benefits of using the SDN/OpenFlow paradigm in wireless networks. The examples include the minimum-energy reprogramming in software-defined sensor networks [32], capacity sharing in hybrid networked environments [33], etc.

2.3. Intra-flow Network Coding

Network coding, which was initially introduced for improving the performance of wired networks in [7], has been considered as a promising communication paradigm to improve network performance in terms of throughput and energy efficiency. In the seminal paper [7], Ahlswede *et al.* proved that in a directed graph with lossless links, if operations are allowed at the intermediate nodes, the theoretical maximum multicast rate can be achieved and it is equal to the min-cut from the source to each receiver. Later, Li *et al.* showed that the linear coding operation is sufficient to achieve that maximum multicast rate [34]. Also, Ho *et al.* further proposed a random linear coding and proved that if the linear coefficients are random selected over a large enough finite field at each intermediate nodes, the destination could decode the packets with a high probability [35]. What's more, Chou *et al.* developed the random linear network coding infrastructure, which makes network coding practical [36]. As the network coding shows its significant advantage over the traditional routing in terms of bandwidth efficiency, there has been a growing interest to applying network coding to improve the performance of wireless networks. For example, MORE [9] is one protocol implemented on a real-world test-bed and achieves a significant performance gain than the traditional routing in WMNs by integrating network coding with opportunistic routing [8].

It is worth noting that MORE protocol does not take multiple flows, fairness, and scheduling into account so that its performance drops as the number of flows increases. To tackle this issue, a cross-layer optimization framework [37] was proposed to optimize the rate of packet transmissions between source and destination pairs. In this framework, a distributed heuristic algorithm was developed to solve the problem. In addition, the problem for the network coding-based opportunistic multicast routing was studied and a duality-based distributed algorithm was proposed to solve the problem [38]. In this proposed scheme, each node only needs to maintain the local information.

Although these aforementioned distributed solutions for optimizing network performance is indeed helpful and can improve performance in terms of throughput, energy efficiency and fairness, they are very costly to be deployed in real-world applications. Fortunately, with the support of the SDN technique, the controller in software-defined networks could acquire the network states dynamically by receiving regular reports from deployed network devices. With the collected information, the controller is able to acquire a global view of the network and can deliver this important information to network devices. Therefore, the global optimization can be achieved by exploiting the SDN technique.

3. OpenCoding

In this section, we present our approach. Particularly, we first brief the architecture, and then introduce the detailed design of our approach as well as how to conduct fairness optimization.

3.1. Architecture

OpenCoding is a novel transmission protocol based on the concept of SDN for WMNs. In our study, we consider a WMN, which consists of one control server, one or more mesh gateways, and multiple mesh routers that are connected to some mobile clients (e.g., smartphones, notebooks, etc.). In such a software-defined network, the complexity of managing the network is low and the network administrator can manage the entire network. In addition, the network can be controlled in real-time based on the intelligence of network controllers.

Similar to the OpenFlow protocol, in OpenCoding-enabled wireless mesh routers, packet forwarding intelligence are moved to the control server, which is referred to as the OpenCoding controller. As there is no control functionality residing at the OpenCoding-enabled routers, the data forwarding functionality can be kept simple by only containing packet coding, recoding, and decoding functions, which are basic primitives defined in the random linear network coding. Particularly, when a new data flow arrives at an OpenCoding-enabled router, the router will perform a *PACKET_IN* function, to ask for the routing for this flow. Then, the controller makes the best decision and computes the path for this flow, and distributes flow entries, which contain actions to perform on individual routers on the selected routing. After receiving these flow entries, those designated routers will carry out corresponding actions (e.g., the packet coding function in source router, recoding function in forwarding router, and decoding function in destination router).

The OpenCoding controller, which runs a network-wide network operating system (e.g., NOX [39], etc.), is the network element that is responsible for managing routers and making decision to meet requirements. All routers are connected to the controller through the secure channel via one hop or multiple hops. This secure channel can be provided by using the SSIDs described in [4] or the use of multi-radio multi-channel (MRMC) technique [17]. Notice that it is easier and cheaper to use the different SSIDs to separate the controller-to-router communication from router-to-router communication, while the communication quality may be worse due to the interference in the wireless channel. Also notice that using the multi-radio multi-channel technique can address the interference problem, but it requires extra hardware and also needs to deal with the channel assignment issue.

3.2. The Design of OpenCoding

In the following, we first introduce the basic idea of our proposed approach and then present the detailed design of our approach.

Basic Idea. In a conventional WMN, once a packet is received and needs to be forwarded by a mesh router, it first stores this packet into the packet buffer pool and then uses a set of rules to find the next-hop of the packet according to routing protocols. When there is a transmission opportunity, the router sends the packet to the corresponding next-hop node. In such a hop-by-hop way, packets can be reliably delivered to destinations.

As stated, in a conventional mesh router, the function that handles the forwarding of packets (i.e., data plane) and the control of forwarding rules (i.e., control plane) are closely coupled. This makes it is hard for routers to extend with new functions. SDN/OpenFlow addresses this issue by decoupling the control and data plane. To be specific, the control plane is implemented partially in a server rather that resides on the router only. This leads to an easier network management. More importantly, in the view of entire network, the global optimization can be achieved, leading to an efficient use of network resources. By leveraging SDN, we can develop an effective scheme for carrying out intra-flow network coding to achieve a higher network performance gain than the network, in which the SDN technique is not used, in terms of both throughput and fairness.

Network State Measurement and Management. In order obtain a global view of the entire network, the controller in OpenCoding uses the Link Layer Discovery Protocol (LLDP), which is also adopted in OpenFlow protocol. To do so, the controller first sends a *PACKET_OUT* message to all mesh routers. As soon as the mesh router receives the *PACKET_OUT* message, it will send LLDP messages to all of its neighbor routers. Because there is no corresponding flow tables to process these LLDP messages, the neighbor routers will send *PACKET_IN* messages to the controller to report the link quality measured by itself. In this way, after receiving the *PACKET_IN* messages, the controller can acquire the topology and current network state of the global network.

In addition, to maintain the network topology and state, the controller will periodically send *PACKET_OUT* messages and the LLDP packets to mesh routers. The controller will also obtain the network information from the feedback *PACKET_IN* messages generated from mesh routers, which can be used to update network topology dramatically.

Controller. In the following, we describe the detailed protocol in the perspective of the components of OpenCoding, including controller, source router, forwarding router, and destination router. In a WMN, mobile mesh clients can access the network through mesh routers. If a mobile client wants to access the Internet or needs to communicate with other clients in the same WMN, the client shall first connects to one of mesh routers, and that mesh router will be responsible for forwarding all traffic flows. Denote S as the source mesh router that connects to the mobile client that has data to transmit and denote D as the destination mesh router that is the mesh router with gateway or connecting to the destination mobile client. When a source mobile client initiates a traffic flow, S will send *PACKET_IN* message to the controller to request for a new traffic flow.

After receiving a *PACKET_IN* message from S, the controller computes the best routing with the global view. As we know, in a traditional hop-by-hop routing, after the routing path is established by some routing protocols, a node keeps sending a packet until the next hop receives it or the number of transmissions exceeds a particular threshold. Nonetheless, in an intra-flow network coding that is similar with an opportunistic routing, there is no particular next hop. The nodes closer to the destination than the current transmitter can participate in forwarding the packet. Therefore, in an intra-flow network coding, the routing of each node consists of one or multiple next-hop forwarding routers. What is more, the threshold of transmission count for the current batch shall also be assigned to ensure the efficiency of bandwidth use.

The detail of how to derive threshold of the number of transmissions can be referred to Equations (1) and (2) in [9]. Here, we use a simple topology to demonstrate the computation of the transmission count. Consider the scenario shown in Figure 4, where the value under the arrow is the packet delivery probability of corresponding link. For every packet from S to D, in expectation, only 1.67 transmissions rather than 2 transmissions are needed when the network coding is used, because R only needs to forward 67% of linear combination of received packets. That is enough for D to recover all packets as D receives the other 33% through link "$S \to D$".

Further, notice that the number of transmissions computed in [9] is only based on the average loss rate of links and can be optimized. After the routing is computed, the controller installs these routing rules to all potential transmitting routers through *MODIFY_STATE* messages. The rules contain: (i) the flow ID that is used to identify each traffic flow, (ii) the batch number in each flow, (iii) source IP, (iv) destination IP, (v) forwarding router list for each router, and (vi) forwarding transmission count. In the simple scenario shown in Figure 4, C will tell S and

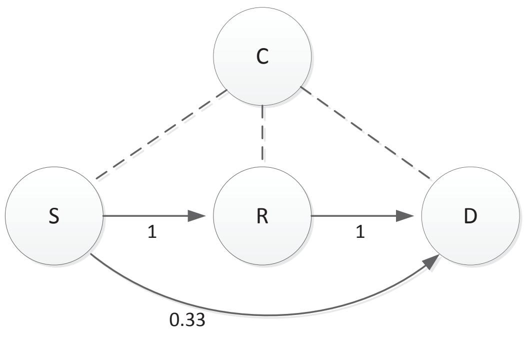


Figure 4. A simple topology

R to transmit once and 0.67 times for each encoded packet to R and D, respectively. In addition, as soon as the *BATCH_FINISHED* message that is sent from the destination router is received by the controller, the controller will immediately send *BATCH_NEXT* message to all transmitting nodes. The source router will stop forwarding packets in the current batch and activate the transmission of the next batch, while forwarding routers will also stop forwarding packets in the current batch as well and wait for the next batch.

Some key message used in the OpenCoding is summarized in Table 1.

Source. When a source mobile client initiates a traffic flow, S performs the *PACKET_IN* function, to request the controller for a new traffic. After receiving the request, the controller makes a decision whether to accept the request or not. If so, the controller computes the best routing for this traffic and installs the forwarding rules on the routers, which will participate in this traffic transmission. As soon as the rule is installed on the S, S will inform source mobile clients to start sending packets to S.

On receiving the stream of packets, S stores the packets first and divides them into batches. Each batch consists of m packets

$$\mathbf{x} = \underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2, \ldots, \underline{\mathbf{x}}_m,$$

where $\underline{\mathbf{x}}_i$ is represented as a vector of

$$\underline{\mathbf{x}}_i = (\underline{x}_{i,1}, \underline{x}_{i,2}, \ldots, \underline{x}_{i,n}).$$

Also, each $\underline{x}_{i,j}$ is a symbol of finite field $\mathbb{F}_q^n$. Here, m is the batch size, q is a prime of a proper size, and all the arithmetic operations are done over $\mathbb{F}_q$. In addition, S generates an augmented packet $\mathbf{x_i}$ by prefixing $\underline{\mathbf{x}}_i$ with the i^{th} unit vector of dimension m:

$$\mathbf{x}_i = (\overbrace{\underbrace{0, \ldots, 0}_{i-1}, 1, 0, \ldots, 0}^{m}, \underline{x}_{i,1}, \underline{x}_{i,2}, \ldots, \underline{x}_{i,n})$$

Table 1. Message type in the OpenCoding

Types of Message:	Description
MODIFY_STATE:	The controller send MODIFY_STATE messages to routers to add, delete and modify routing rules on routers.
PACKET_OUT:	The controller sends PACKET_OUT message to routers that tell routers to send corresponding messages out.
PACKET_IN:	The source mesh router sends the PACKET_IN message to the controller to inform that there is a new packet received in routers and ask for corresponding instructions.
BATCH_FINISHED:	The destination mesh router sends the BATCH_FINISHED message to inform the controller that the transmission of this batch should be completed.
BATCH_NEXT:	The controller sends the NEXT_BATCH to inform the source and forwarding router to stop the transmission of this batch, and another batch of packets should be activated
FLOW_REMOVED:	Routers sends Flow_REMOVED messages to the controller to inform that one of the flow tables is removed

For a random linear network coding, S randomly picks the coefficients from the finite field $\mathbb{F}_q$ and creates the combinations of packet $\mathbf{x}_i$. This means that for packets

$$\mathbf{x} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m,$$

and coefficients

$$\alpha = \alpha_1, \alpha_2, \ldots, \alpha_m,$$

an output packet is represented as

$$\mathbf{y} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i = (\sum_{i=1}^{m} \alpha_i x_{i,1}, \ldots, \sum_{i=1}^{m} \alpha_i x_{i,m+n})$$

The first m symbol of $\mathbf{y}$ is called the global coding coefficient. Subsequently, S sends encoded packets to the forwarding routers defined in installed rules and waits for the *BATCH_NEXT* message from the controller.

Forwarders. After rules are installed, mesh routers work in promiscuous mode, listening all transmissions. When the router receives a packet, it first checks the packet to confirm whether the packet matches the rules that are installed. More specifically, it checks whether the flow ID and the batch ID extracted from the packet's header matches any entry in the flow table, and whether it is in the forwarder list of packets. If so, it further checks whether the packet is an innovate packet by *Guassian eliminations*. An innovative packet is linearly independent from the packets that the node has previously received in this batch. A packet that does not match any flow table or is not innovative will be dropped. Otherwise, the packet will be stored for encoding.

After the forwarding router accepts an innovate packet, it then randomly selects the coefficients and linearly combines the all received packets in the same batch in a similar way with the source. This can be formulated as follows:

$$\mathbf{z} = \sum_{i=1}^{m} \gamma_i \mathbf{y}_i.$$

Here, $\mathbf{z}$ is the new packet created by the forwarding router, γ_i is the random coefficient for the i^{th} packet and all the $\mathbf{y}$ are packets received in the same batch, including the newly received one. Notice that in a fault-free execution of the random linear network coding, the packets transmitted in the network are all linear combinations of the original augmented messages: $x = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$.

Subsequently, a packet header, which contains the source and destination IP, flow ID, batch ID and forwarding list, will be appended to the newly created packet. When a transmission opportunity arises, the packet will be send to its next-hop forwarding routers.

Destination. The destination router D receives packets in a similar way as the forwarders. After receiving m linearly independent packets $\mathbf{z}_j, j = 1 \ldots m$, D can recover the original packets $\mathbf{x}_i, i = 1, \ldots, m$ using the global coding matrix $\mathbf{G}$, where i^{th} row is the corresponding global coding vector through *Guassian eliminations* with a high probability:

$$[\mathbf{I}, \mathbf{X}] = \mathbf{G}^{-1} \cdot \mathbf{Z}. \qquad (1)$$

Here, $\mathbf{I}$ is the identity matrix, $\mathbf{X}$ is the matrix, in which i^{th} row is the corresponding $\mathbf{x}_i$, $\mathbf{G}^{-1}$ is the inverse of the matrix $\mathbf{G}$, and $\mathbf{Z}$ is the matrix, in which i^{th} row is the corresponding $\mathbf{z}_i$.

As soon as the current batch can be decoded, D performs the *BATCH_FINISHED* function, and sends a message to inform the controller that the transmission of this batch should be complete and the next batch transmission can be activated. The controller then performs the *NEXT_BATCH* function and sends a message to the source and all the forwarders.

In addition, D forwards the decoded packets to destination mobile clients or to the Internet.

The control flow of OpenCoding is summarized in Figure 5. On the sender's side, before sending a packet, the router will add the header to the packet, and check whether the transmission count for this batch has reached the pre-determined value. If so, the router will turn to transmit the next packet; otherwise, this packet will be sent to its next-hop forwarding routers. On the receiver's side, the router checks the header of packet to determine whether to accept or not. If so, the router will store the packet for recoding (i.e., if it is forwarding router) or decoding (i.e., if it is destination router).

3.3. Fairness Optimization for OpenCoding

Benefited from the centralized control logic and the global network view, OpenCoding can address other issues, which are not easily solved in a distributed manner. We now introduce a scheme to ensure the fairness of OpenCoding.

Considering a scenario in which there are two traffic flows f_1 and f_2 showed in Figure 6, which are transmitted with intra-flow random linear network coding. If the node R is the common forwarding router of both flows, it will participate in forwarding packets in these two flows. In the following, we present our idea using an example. We assume that R just begins to transmit a new batch from f_1 and p_1 is the first packet in this batch, while another batch from f_2 is nearly at the end and p_2 is the last packet in that batch. In Figure 6, packets belong to different flows are marked with different colors. In the original MORE protocol, these two packets will not be differentiated and they will be transmitted in First-In-First-Out (FIFO) manner. Nonetheless, it is more important to transmit p_2 than p_1 because as soon as the destination router receives p_2, it can decode all packets in this batch while the reception of p_1 will not bring any benefit at that moment. To address this issue, we develop a novel scheme to ensure the fairness of OpenCoding by exploiting the centralized control of SDN.

When there are multiple flows in the network, the controller periodically requests the destination routers to report their states of the current batch. The reports from the destination routers contain the current batch ID and the progress of the corresponding batch. In our proposed scheme, destination routers only reports whether they receives $m/2$ innovative packets. Meanwhile, forwarding routers maintain two virtual queues, one for low priority packets (e.g., the packet is in the first half of the batch), and the other for high priority packets (e.g., the packet is in the second half of the batch). When a new packet is added to the output queue, an entry is added to the appropriate virtual queue based on the progress of packet's batch.

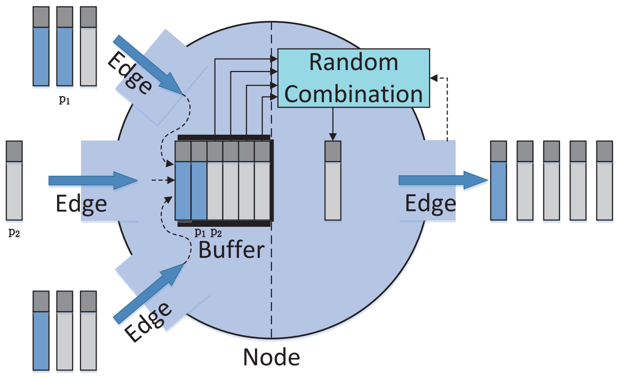


Figure 6. An Example of fairness optimization

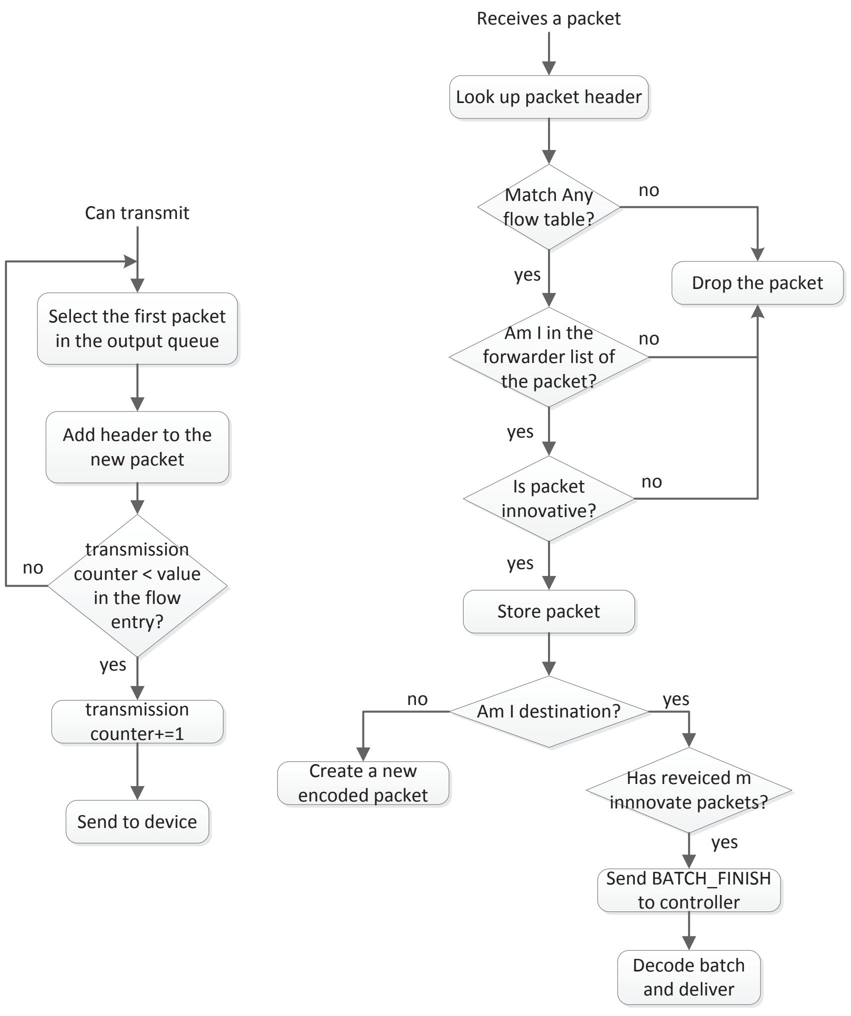
In addition, when the last packet in the high priority queue is sent, the low priority queue will be the high priority queue and the high priority queue will be the low priority queue. The controller periodically updates rules, which are installed in the forwarding routers based on the report from destination routers. The rules inform forwarding routers the progress of the current transmitting batch. To be specific, when a new packet is received, if more than half of packets in that batch are received by the destination router, the new encoded packet will be added to the high priority queue waiting to be forwarded, and vice versa. When there is a transmission opportunity, it will first look up the high priority queue. If it is not empty, the head packet in the high priority queue will be transmitted.

Notice that the progress of corresponding batch can be further subdivided and more corresponding virtual queues can be used to further improve the fairness. Nonetheless, from the simulation results described in the performance evaluation section, dividing the batch into two halves and two virtual queues are sufficient to ensure the fairness. Again, it is worth noting that developing such a special scheduler is benefit from a global view enabled by SDN, while in the traditional system with network coding, such a scheduler is hard to achieve.

4. Overhead Analysis

In this section, we analyze the routing overhead of our OpenCoding. As we described in Section 3, there are three types of control messages needed for the controller to obtain a global view of the entire network: (i) *PACKET_OUT* message, (ii) LLDP message and (iii) *PACKET_IN* message. In addition, the flow table request and distribution incur extra routing overhead. Therefore, the routing overhead of OpenCoding consists of following two components and can be represented by

$$O_{OpenCoding} = O_{TO} + O_{FT}, \qquad (2)$$


Can transmit
Select the first packet in the output queue
Add header to the new packet
transmission counter < value in the flow entry?
no
yes
transmission counter+=1
Send to device
Receives a packet
Look up packet header
Match Any flow table?
no
yes
Drop the packet
Am I in the forwarder list of the packet?
no
yes
Is packet innovative?
no
yes
Store packet
Am I destination?
no
yes
Create a new encoded packet
Has reveiced m innnovate packets?
yes
Send BATCH_FINISH to controller
Decode batch and deliver

(a) Sender side (b) Receiver side

Figure 5. Flow chart of the OpenCoding

where O_{TO} is the overhead caused by the routing discovery and maintenance, and O_{FT} is the overhead caused by the flow table request and distribution. In the following, we derive the closed formulae for the two components. All main notations used in this section are listed in Table 2.

4.1. Overhead for Routing Discovery and Maintenance

The total overhead of routing discovery and maintenance O_{TO} is defined as the total amount of information needed by control messages. Assume that a wireless mesh network consists of N wireless nodes that are uniformly distributed in a square field with a size of $a \times b$. To simply the analysis, assume the transmission range of the routers is equal and is set to r. Then, the ratio of the node's coverage area to the whole area p is provided by $\frac{\pi r^2}{a \times b}$.

Table 2. Notation

N:	Number of nodes in the network
a, b:	Size of the square field of network
r:	Transmission range of a node
p:	Ratio of node's coverage area to the whole network area
τ:	Time step that is also referred to as the interval for network topology update
η:	Minimum average number of transmission to broadcast a message to all the other nodes
h	Random variable of number of hops between two arbitrary nodes
O_{TO}:	Overhead caused by the routing discovery and maintenance
O_{FT}:	Overhead caused by the flow table request and distribution

Denote η as the minimum number of transmissions to broadcast a message to every node in the network. According to [40], if the average number of neighbors for any node is at least 2, η can be determined as

$$\eta = \left\lceil \frac{N-2}{pN-2} \right\rceil. \tag{3}$$

Denote I_{PO} as the amount of information needed for the controller to broadcast the *PACKET_OUT* message within a time step τ. The controller periodically broadcast the *PACKET_OUT* message of length $\log N$ to all the mesh routers in the network. Here, $\log N$ is the amount of information required to identify the controller. Therefore, the amount of information that controller broadcasts the *PACKET_OUT* message can be derived by

$$I_{PO} \geq \eta \log N. \tag{4}$$

Denote I_L as the amount of information needed for mesh routers to advertise LLDP messages to their neighbor routers within time step τ. As described, when the mesh router receives the *PACKET_OUT* message from the controller, it will advertise LLDP messages of length $\log N$ to all of its neighbor routers to inform its neighbor about the existence. Here, $\log N$ is the amount of information required to differentiate one node from the others. There are $N-1$ mesh routers in network. Then, I_L can be derived by

$$I_L = (N-1)\log N. \tag{5}$$

Denote I_{PI} as the amount of information needed for mesh routers to report the link state through *PACKET_IN* messages within time step τ. As described, when the mesh router receives LLDP messages from other routers, it is informed that the sender of LLDP message is one of its neighbors. Then, it will send *PACKET_IN* message to the controller to report the link states measured by itself. That is to say, the message contains the information about itself and its pN neighbor routers. Therefore, the message length is $(pN+1)\log N$. In addition, according to [41], the expect number of hops between two randomly located nodes in a $a \times b$ square field can be determined by Equations 6 and 7.

In Equation 6, $f_L(x)$ is the probability distribution function (PDF) of the random variable x, which is the Euclidean distance between two arbitrary nodes. Then,

$$E[h] = \left\lceil \frac{\int x f_L(x) dx}{r} \right\rceil. \tag{7}$$

Therefore, the amount of information needed for mesh routers to report the link state through *PACKET_IN* messages I_{PI} can be derived by

$$I_{PI} = E[h] \cdot N(pN+1)\log N. \tag{8}$$

Overall, given a time step τ, the lower bound on the total overhead caused by the LLDP protocol for the whole network per second is given by

$$O_{TO} = \frac{1}{\tau}(I_{PO} + I_L + I_{PI}), \tag{9}$$

where I_{PO}, I_L, and I_{PI} are given in Equations 4, 5 and 8.

From the result of Equation 9, we can see that the overall overhead of routing discovery and maintenance O_{TO} is $O(N^2 \log N)$ and it depends on the overhead of sending *PACKET_IN* messages to the controller. According to the result of [40], the routing overhead of proactive routing protocols is $O(N \log N)$. Therefore, the routing overhead in this part is larger. That is because in both our OpenCoding and OpenFlow, the controller periodically asks routers for the topology information of all of their neighbors. In order to reduce the overhead of sending *PACKET_IN* messages, the controller can increase the time interval and the routers can send the topology message only when there are changes in the states of their links between themselves and their neighbors.

4.2. Overhead of Flow Table Request and Distribution

The total overhead of flow table request and distribution O_{FT} is defined as the total amount of information needed in transmitting the request and flow table itself.

Denote $I_{S\to C}$ be the amount of information of routing request from a source router S to the controller C. Denote $I_{C\to\mathbb{F}}$ as the amount of information of the flow table distribution from the controller C to all the forwarding mesh routers $\mathbb{F}$.

A source router S sends a routing request message to the controller C when there is a new traffic to be transmitted. The message contains the information of source and destination. Then, the message length is $2\log N$. In addition, if the source and the controller are randomly located in the network, the expect number of hops between them is $E[h]$. Therefore, $I_{S\to C}$ can be derived by

$$I_{S\to C} = E[h] \cdot 2\log N. \tag{10}$$

After receiving the request message, the controller computes the best routing path for this traffic flow and installs the flow table on the forwarding routers, which will participate in the transmission of this traffic flow. The flow table contains the information of all the forwarding routers. Notice that as $E[h]$ is the expect number of hops between two random located routers, it can also represent the number of routers in the shortest path from source to destination. In a random linear network, the number of nodes participating in the transmission will be larger than that of traditional routing. Therefore, the length if flow table message

$$f_L(x) = \begin{cases} 2x\left[\frac{\pi}{ab} - \frac{2x}{a^2b} - \frac{2x}{ab^2} + \frac{x^2}{a^2b^2}\right] & 0 \le x < b \\ \frac{2x}{ab}\left[\frac{\pi}{2} - \arcsin(1 - \frac{2b^2}{x^2})\right] - \frac{4x}{ab^2}\left[x - \sqrt{x^2 - b^2}\right] - \frac{2x}{a^2} & b \le x < a \\ \frac{2x}{ab}\left[\arcsin(\frac{2a^2}{x^2} - 1) - \arcsin(1 - \frac{2b^2}{x^2})\right] + \frac{2x}{a^2b^2}[a^2 + b^2 - x^2] - \frac{4x}{ab^2}[a - \sqrt{x^2 - b^2}] + \frac{4x}{a^2b}[\sqrt{x^2 - a^2} - b] & a \le x < \sqrt{a^2 + b^2} \end{cases} \tag{6}$$

is at least $E[h]\log N$. The flow table will be installed to at least $E[h]$ routers, and every flow table will be transmitted through $E[h]$ hops. Thus, $I_{C\rightarrow\mathbb{F}}$ can be estimated by

$$I_{C\rightarrow\mathbb{F}} \ge E[h]^3 \log N. \tag{11}$$

Then, the overhead caused by the flow table request and distribution can be estimated by

$$O_{FT} = \left(E[h]^3 + 2E[h]\right)\log N. \tag{12}$$

From the result of Equation 12, we can see that the overall overhead of each flow table distribution O_{FT} is $O(\log N)$. According to the result of [41], the routing overhead of reactive routing protocols is $O(N \log N)$. Therefore, routing overhead of this part is smaller. That is because in the flow table distribution procedure of OpenCoding and OpenFlow, the source is aware of the route to the controller and the controller also knows the routes to all the forwarding routers so that the broadcast RREQ messages are not needed.

5. Performance Evaluation

We perform simulation experiments to show the effectiveness of our proposed protocol. In the following, we first present the simulation methodology and then show the simulation results.

5.1. Methodology

We evaluate the performance of our proposed scheme in comparison with existing schemes in terms of throughput, average end-to-end delay, and rule activation time. In our simulations, the end-to-end delay is defined as the duration from a packet is sent from the source router until the packet is received by the destination router. In MORE and OpenCoding, the end-to-end delay is averaged across batches. The rule activation time of both OpenFlow and OpenCoding is defined as the duration from the time when the first packet of a new flow arrives at the source router and the source router performs *PACKET_IN* function until all the flow-tables are distributed to the corresponding forwarding routers.

We use ns-3 simulator [42] and compare the performance of our proposed OpenCoding protocol in comparison with the following protocols: OLSR [11], OpenFlow [12], and MORE [9]. In the simulations to evaluate the throughput, we use the topology of Roofnet

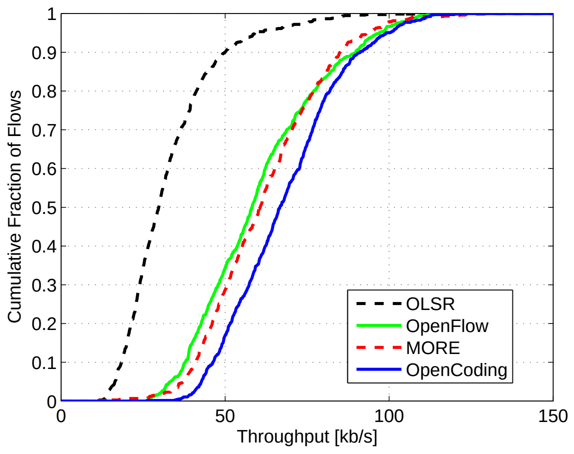


Figure 7. Throughput CDF of OLSR, OpenFlow, MORE and OpenCoding

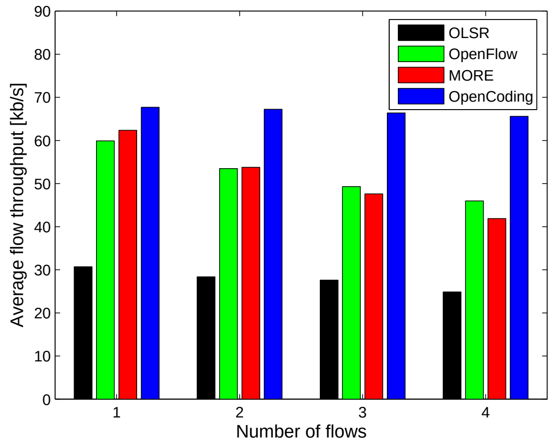


Figure 8. The throughput of OLSR, OpenFlow, MORE and OpenCoding with different numbers of concurrent flows

[43], which is a known experimental 802.11b/g wireless mesh network developed by MIT (Massachusetts Institute of Technology). While in the simulations to evaluate end-to-end delay and rule activation time, we use the random topology that consists of 20, 35, and 50 nodes to evaluate the scalability of network, respectively. To keep the same node density, nodes are randomly placed in a 377*377, 500*500 and 600*600 area, respectively. In all simulations, we use User Datagram Protocol (UDP) traffic and the packet size for

all protocols is set to 1500B. In addition, to implement the network visualization as described in OpenFlow, the nodes in a software-defined network are equipped with two network cards, one for the control path and the other for the data path. In each simulation, we randomly select the source-destination pairs in the corresponding topology. In addition, the nodes works in 802.11b and the data transmission rate is 1Mbps. For both MORE and our OpenCoding, the batch size is set to 32.

5.2. Results

In the following, we show the evaluation results in terms of throughput, end-to-end delay and rule activation time.

Throughput. Figure 7 shows the Cumulative Distribution Function (CDF) of the transmission throughput of OLSR, OpenFlow, MORE and our proposed OpenCoding, measured over 2000 different runs. In each run, there is only one traffic flow in the network and the source-destination pairs are randomly selected. Our data shows that OpenCoding outperforms the other three protocols in terms of throughput. Particularly, as we can see from the figure, OpenCoding achieves a significant larger throughput gain than the traditional routing OLSR, and outperforms the OpenFlow and MORE as well. In the median case, OpenCoding has a 68.32%, 12.46%, and 9.88% throughput gain over OLSR, OpenFlow and MORE, respectively. We also observe that there is a performance gap between our results and the results of [4] which is based on a real test-bed. In our simulation, the data communication and control communication is separated by using multi-radio multi-channel technique to avoid the channel interference, while they used different SSIDs that may degrade the throughput performance to some extent.

Figure 8 shows the average per-flow throughput as a function of the number of concurrent flows for these protocols. In all cases, OpenCoding still outperforms the other three protocols. In addition, the throughput gain increases as the number of flows increases. As we can see from the figure, the average per-flow throughout of OpenCoding drops little, while the others decline much more than that of OpenCoding. The evaluation data confirms that OpenCoding can ensure the fairness with the global optimization.

Because of the intelligent management of the controller introduced by the concept of SDN, the best routing can be optimally selected by the controller, which could have a global view of the network. To this extent, OpenFlow outperforms traditional routing mechanisms. Meanwhile, because of the utilization of the broadcast nature of wireless communication medium, MORE also achieves a significant throughput gain over the traditional routing mechanisms. Exploiting the benefits of both characteristics, the throughput performance of OpenCoding is the best. In addition, the developed scheduler for OpenCoding can effectively ensure the fairness of multiple flows.

End-to-end Delay. Figure 9 illustrates the relationship between the average per-packet end-to-end delay and the number of flows for different numbers of nodes in the network for OLSR, OpenFlow, MORE, as well as our OpenCoding, respectively. Each simulation is repeated 1000 times and the results are averaged as illustrated. As we can see from the figure, all the average delays increase as the number of flows and number of nodes in the network increases. In particular, when the number of nodes is 20, as the number of concurrent flows increases from 1 to 4, the end-to-end delay of OpenCoding increases 18.1% only, while those of OLSR, OpenFlow and MORE increase 75.4%, 53.5%, and 124%, respectively. The result shows that OpenCoding achieves a better performance as the number of network flows increases. In addition, when the number of flows is 1, as the number of nodes in the network increases from 20 to 50, the end-to-end delay increases 43.9%, 41.7%, 37.2%, and 43.1% for OLSR, OpenFlow, MORE and OpenCoding, respectively. This result shows that the performance of OpenCoding is almost same as the network size increases.

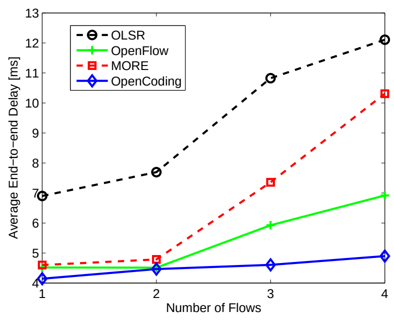
Rule Activation Time. In order to compare the rule activation time of OpenFlow and OpenCoding, we change the number of concurrent flows, as well as the number of network nodes, and plot the average value as shown in Figure 10. As we can see from the figure, the rule activation time of OpenFlow is a little shorter than that of OpenCoding. That is because in a network with network coding, there are more routers that participate in the transmission to fully exploit the broadcast nature in wireless communication. In particular, when the number of flows increases, the rule activation time will grows as well. That is because the controller needs to compute the best routing for more flows. Therefore, when there are a large number of traffic flows concurrently transmitted over the network, the computation ability of the controller needs to be powerful enough to avoid being performance bottleneck.

6. Discussion

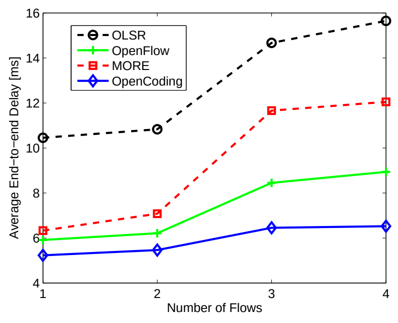
In this section, we discuss some open issues related to OpenCoding.

6.1. Controller Design

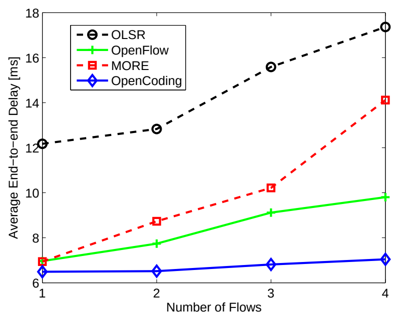
When the network size becomes larger, the SDN controllers could become the performance bottleneck due to a large number of requests from routers. For



(a) node number:20

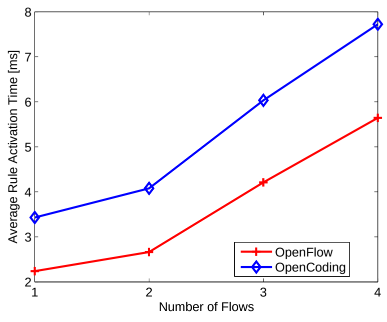


(b) node number:35

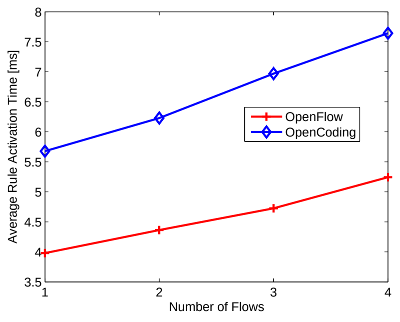


(c) node number:50

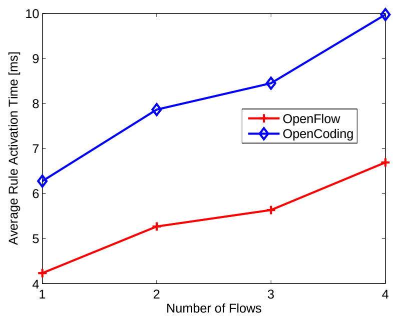
Figure 9. Average Delay of OLSR, OpenFlow, MORE and OpenCoding versus number of flows with different node numbers



(a) node number:20



(b) node number:35



(c) node number:50

Figure 10. Average Rule Activation Time of OLSR, OpenFlow, MORE and OpenCoding versus number of flows with different node numbers

the sake of scalability, reliability and robustness, it has been realized that the logically-centralized controller must be physically distributed [44] in Software-Defined Networks. Because of the lossy and unreliable wireless channels, the transmission delay will be unacceptable if the distance between controller and router is long. Therefore, it is more necessary to distribute multiple controllers in software-defined wireless mesh networks.

Nonetheless, the controllers located in a distributed way may compete for common resources (such as communication channels). To this end, an optimal controller placement strategy and a carefully designed scheduling strategy to avoid collision should be developed. On designing such a strategy, two essential questions should be concerned. First, given a wireless network topology, how many controllers are needed in order to achieve a desirable performance? Second, where should the controllers be located? When we design the optimal deployment strategy, we need to consider the dynamics of traffic, the density distribution of mobile customers and routers, and realistic constraints (e.g., geo-restriction and information availability), deployment costs, the coverage and quality of service for mobile customers, and applications. We will develop scenarios for validate the optimal deployment of controllers. In addition, there are other questions to be solved: How to synchronize the information among the controllers and how do the mesh routers connect to the controllers? We will answer these questions in the future work.

6.2. Security

There has been limited research effort to date on the security issues associated with SDN. Nonetheless, as SDN is a new network architecture, it also brings some new targets for potential attacks. Our developed OpenCoding could operate in hostile environments and all modules and devices could increase the possibility of being compromised. Therefore, this calls for the framework to systematically explore possible attacks against OpenCoding and develop mitigation schemes [45]. Based on security objectives (e.g., availability, integrity, etc.), the adversary can launch attacks against various against various components (e.g., mesh routers, communication networks, and controller) associated with data and control planes.

Some vulnerabilities of SDN are listed below. At the controller level, authentication and authorization is a critical issue. When there is multiple organizations and applications accessing the network resources, if the resources are not enough to be allocated to all the applications, the demand of application with higher privilege should be satisfied with less delay. That is to say, a security model should be designed to isolate the applications with different privileges. Meanwhile, privacy preservation is another critical issue. Since the controller obtains the global view of network traffic information, adversaries may try to compromise the controller or overhear and analyze the traffic through the controller to get these critical information. Therefore, a privacy preservation strategy is also necessary.

At the data transmission level, there are numerous attacks that can be launched to disrupt the effectiveness of networks. As an example, one potential attack is denial-of-service or jamming attacks. The adversary launches an attack by originating a large number of new traffic flows. As the computation and bandwidth resources of controller is limited, this type of attack may be devastating. Moreover, as we know, because of the mixing nature, network coding is vulnerable to network attacks such as pollution attacks [46]. The security problem of OpenCoding should also be considered in future work.

7. Conclusions

In this paper, we proposed a novel OpenCoding protocol, which integrates the software-defined networking technique with intra-flow network coding technique for WMNs. Our developed protocol makes a full use of the broadcast nature of the wireless transmission medium and the global intelligence of SDN controller, which enables the ability of improving the network performance (e.g., throughput, end-to-end delay, and fairness). Similar to the known OpenFlow protocols for wired software-defined networks, OpenCoding decouples the data plane and the control plane in wireless mesh routers, and leaves only network coding functions in each router for easy deployment and management of WMNs. Benefited from a global view of the controlled network, OpenCoding can ensure the fairness of flows. Through a simulation study, our evaluation data shows that OpenCoding outperforms intra-flow network coding protocols such as MORE, and OpenFlow. As ongoing work, we are implementing OpenCoding in a real-world test-bed to further evaluate its performance gain.

Acknowledgement. This work is sponsored in part by the following funds: the National Natural Science Foundation of China under Grant 61373115, Grant 61402356 and Grant 61502381, the Fundamental Research Funds for the Central Universities under Grant xjj2015065, and the China Post Doctoral Science Foundation under Grant 2015M570836. This work was also supported in part by US National Science Foundation (NSF) under grant: CNS 1350145. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] Akyildiz, I. and Wang, X. (2005) A survey on wireless mesh networks. *IEEE Communications Magazine* **43**(9): S23–S30.

[2] Ortiz, S. (2013) Software-defined networking: On the verge of a breakthrough? *IEEE Computer* **46**(7): 10–12.

[3] Hu, F., Hao, Q. and Bao, K. (2014) A survey on software-defined network and openflow: From concept to implementation. *Communications Surveys Tutorials, IEEE* **16**(4): 2181–2206.

[4] Dely, P., Kassler, A. and Bayer, N. (2011) Openflow for wireless mesh networks. In *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)* (IEEE): 1–6.

[5] Abolhasan, M., Lipman, J., Ni, W. and Hagelstein, B. (2015) Software-defined wireless networking: centralized, distributed, or hybrid? *IEEE Network* **29**(4): 32–38.

[6] Huang, H., Li, P., Guo, S. and Zhuang, W. (2015) Software-defined wireless mesh networks: architecture and traffic orchestration. *IEEE Network* **29**(4): 24–30.

[7] Ahlswede, R., Cai, N., Li, S.Y. and Yeung, R. (2000) Network information flow. *IEEE Transactions on Information Theory* **46**(4): 1204 –1216.

[8] Biswas, S. and Morris, R. (2005) Exor: Opportunistic multi-hop routing for wireless networks. *SIGCOMM Computer Communication Review* **35**(4): 133–144.

[9] Chachulski, S., Jennings, M., Katti, S. and Katabi, D. (August 2007) Trading structure for randomness in wireless opportunistic routing. In *Proceedings of ACM SIGCOMM Conference*.

[10] Bioglio, V., Grangetto, M., Gaeta, R. and Sereno, M. (2013) A practical random network coding scheme for data distribution on peer-to-peer networks using rateless codes. *Performance Evaluation* **70**(1): 1–13.

[11] Clausen, T., Jacquet, P., Adjih, C., Laouiti, A., Minet, P., Muhlethaler, P., Qayyum, A. *et al.* (2003) Optimized link state routing protocol (olsr) .

[12] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. *et al.* (2008) Openflow: Enabling innovation in campus networks. *SIGCOMM Computer Communication Review* **38**(2): 69–74.

[13] Zhu, D., Yang, X., Zhao, P. and Yu, W. (2015) Towards effective intra-flow network coding in software defined wireless mesh networks. In *Proceedings of IEEE International Conference on Computer Communication and Networks (ICCCN)* (IEEE): 1–8.

[14] Chowdhury, K.R. and Akyildiz, I.F. (2008) Cognitive wireless mesh networks with dynamic spectrum access. *IEEE Journal on Selected Areas in Communications (JSAC)* **26**(1): 168–181.

[15] Hong, Y.W., Huang, W.J., Chiu, F.H. and Kuo, C.C.J. (2007) Cooperative communications in resource-constrained wireless networks. *IEEE Signal Processing Magazine* **24**(3): 47–57.

[16] Bhatia, R. and Li, L.E. (2007) Throughput optimization of wireless mesh networks with mimo links. In *Proceedings of 26th IEEE International Conference on Computer Communications (INFOCOM)* (IEEE): 2326–2330.

[17] Kodialam, M. and Nandagopal, T. (2005) Characterizing the capacity region in multi-radio multi-channel wireless mesh networks. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking* (ACM): 73–87.

[18] ONF (2012) Software-defined networking: The new norm for networks. *ONF White Paper* .

[19] Yeganeh, S., Tootoonchian, A. and Ganjali, Y. (2013) On scalability of software-defined networking. *IEEE Communications Magazine* **51**(2): 136–141.

[20] Hu, Y., Wang, W., Gong, X., Que, X. and Cheng, S. (2014) On reliability-optimized controller placement for software-defined networks. *Communications, China* **11**(2): 38–54.

[21] Congdon, P.T., Mohapatra, P., Farrens, M. and Akella, V. (2014) Simultaneously reducing latency and power consumption in openflow switches. *IEEE/ACM Transactions on Networking (ToN)* **22**(3): 1007–1020.

[22] Khan, A. and Dave, N. (2013) Enabling hardware exploration in software-defined networking: A flexible, portable openflow switch. In *Proceedings of 2013 IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*: 145–148.

[23] Biswas, J., Lazar, A., Huard, J.F., Lim, K., Mahjoub, S., Pau, L.F., Suzuki, M. *et al.* (1998) The ieee p1520 standards initiative for programmable network interfaces. *IEEE Communications Magazine* **36**(10): 64–70.

[24] Doria, A., Salim, J.H., Haas, R., Khosravi, H., Wang, W., Dong, L., Gopal, R. *et al.* (2010) *Forwarding and control element separation (ForCES) protocol specification*. Tech. rep.

[25] Lakshman, T., Nandagopal, T., Ramjee, R., Sabnani, K. and Woo, T. (2004) The softrouter architecture. In *Proceedings of ACM SIGCOMM Workshop on Hot Topics in Networking*, **2004**.

[26] Yu, W., Chellappan, S., Xuan, D. and Zhao, W. (2004) Distributed policy processing in active-service based infrastructures. *International Journal of Communication Systems (IJCS)* **19**(7): 727–750.

[27] Galis, A., Plattner, B., Smith, J.M., Denazis, S., Moeller, E., Guo, H., Klein, C. *et al.* (2000) A flexible ip active networks architecture. In *Proceedings of Second International Working Conference on Active Networks, ed. H. Yasuda* (Springer-Verlag): 1–15.

[28] Bernardos, C., De La Oliva, A., Serrano, P., Banchs, A., Contreras, L., Jin, H. and Zu?niga, J. (2014) An architecture for software defined wireless networking. *IEEE Wireless Communications* **21**(3): 52–61.

[29] Riggio, R., Gomez, K.M., Rasheed, T., Schulz-Zander, J., Kuklinski, S. and Marina, M.K. (2014) Programming software-defined wireless networks. In *Proceedings of 2014 10th International Conference on Network and Service Management (CNSM)*: 118–126.

[30] Guimaraes, C., Corujo, D. and Aguiar, R. (2014) Enhancing openflow with media independent management capabilities. In *Proceedings of 2014 IEEE International Conference on Communications (ICC)*: 2995–3000.

[31] Yang, F., Gondi, V., Hallstrom, J., Wang, K.C. and Eidson, G. (2014) Openflow-based load balancing for

wireless mesh infrastructure. In *Proceedings of IEEE International Conference on Consumer Communications and Networking Conference (CCNC)*: 444–449.

[32] Zeng, D., Li, P., Guo, S. and Miyazaki, T. (2014) Minimum-energy reprogramming with guaranteed quality-of-sensing in software-defined sensor networks. In *Proceedings of 2014 IEEE International Conference on Communications (ICC)*: 288–293.

[33] Santos, M., de Oliveira, B., Margi, C., Nunes, B., Turletti, T. and Obraczka, K. (2013) Software-defined networking based capacity sharing in hybrid networks. In *Proceedings of 2013 21st IEEE International Conference on Network Protocols (ICNP)*: 1–6.

[34] Li, S.Y., Yeung, R. and Cai, N. (2003) Linear network coding. *IEEE Transactions on Information Theory* **49**(2): 371–381.

[35] Ho, T., Medard, M., Koetter, R., Karger, D., Effros, M., Shi, J. and Leong, B. (2006) A random linear network coding approach to multicast. *IEEE Transactions on Information Theory* **52**(10): 4413–4430.

[36] Chou, P.A., Wu, Y. *et al.* (2007) Network coding for the internet and wireless networks. *IEEE Signal Processing Magazine* **24**(5): 77.

[37] Radunović, B., Gkantsidis, C., Key, P. and Rodriguez, P. (2010) Toward practical opportunistic routing with intra-session network coding for mesh networks. *IEEE/ACM Transactions Networking (ToN)* **18**(2): 420–433.

[38] Khreishah, A., Khalil, I.M. and Wu, J. (2012) Distributed network coding-based opportunistic routing for multicast. In *Proceedings of the Thirteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '12 (New York, NY, USA: ACM): 115–124.

[39] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N. and Shenker, S. (2008) Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review* **38**(3): 105–110.

[40] Tran, Q.M. and Dadej, A. (2014) Optimizing topology update interval in mobile ad-hoc networks. In *Proceeding of 2014 IEEE 79th Vehicular Technology Conference (VTC Spring)*: 1–5.

[41] Naserian, M., Tepe, K. and Tarique, M. (2005) Routing overhead analysis for reactive routing protocols in wireless ad hoc networks. In *Proceedings of IEEE International Conference on Wireless And Mobile Computing, Networking And Communications (WiMob)*, **3**: 87–92 Vol. 3.

[42] Henderson, T.R., Lacage, M., Riley, G.F., Dowell, C. and Kopena, J. (2008) Network simulations with the ns-3 simulator. *SIGCOMM demonstration* **15**: 17.

[43] *MIT Roofnet* (http://pdos.csail.mit.edu/roofnet/doku.php).

[44] Nunes, B., Mendonca, M., Nguyen, X.N., Obraczka, K. and Turletti, T. (2014) A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys Tutorials* **16**(3): 1617–1634.

[45] Bhattarai, S., Rook, S., Ge, L., Wei, S., Yu, W. and Fu, X. (2014) On simulation studies of cyber attacks against lte networks. In *Proceedings of IEEE International Conference on Computer Communication and Networks (ICCCN)* (IEEE): 1–8.

[46] Zhu, D., Yang, X. and Yu, W. (2015) Spais: A novel self-checking pollution attackers identification scheme in network coding-based wireless mesh networks. *Computer Networks* **91**: 376 – 389.