

A Control-Message Quenching Algorithm in OpenFlow-Based Wireless Mesh Networks with Dynamic Spectrum Access

Xin Li¹, Xiaoyan Hong², Yu Lu¹, Ke Bao¹, Fei Hu¹, Sunil Kumar³

¹Department of Electrical and Computer Engineering, University of Alabama, Tuscaloosa, AL

²Department of Computer Science, Tuscaloosa, AL

³Department of Electrical and Computer Engineering, San Diego State University, San Diego, CA

Abstract—In recent years, Open Flow is becoming a popular network architecture. It can provide fine-grained monitor to the flows in the network. The separation of control plane and data plane makes it flexible. However, to the best of our knowledge, most OpenFlow protocols are designed for wired networks. And the switch state and controller bottleneck are still major issues in OpenFlow network. In this paper, we first proposed a wireless OpenFlow based Cognitive Radio network. In order to solve the channel selection issue in CRN, we introduce the Hierarchical Dirichlet Process(HDP) model to sense and classify the wireless channel. To reduce the state of switch, we use wildcard rule scheme. In addition, an algorithm is proposed in the paper to further quench the requests to the controller. The simulation results show that our proposed CMQ algorithm with information from neighbors outperforms the conventional CMQ algorithm in throughput and packet delay performance.

I. INTRODUCTION

In recent years, Open Flow[1] is becoming a popular network architecture. As more and more users starts to join the conventional Internet, the drawbacks of the conventional networks are now gradually appearing. OpenFlow network provides us a brand new sight to the development of networks. This architecture separates the control plane and data plane from the hardware level(physically). OpenFlow has plenty of benefit compare to other network structures. Firstly, the control plane and data plane are decoupled, this means more flexible networks can be figure out with customized rules in the network. Secondly, OpenFlow provides us a new platform to design and test network protocols. Researchers could test new protocols in a real network environment. Thirdly, in OpenFlow architecture networks we could monitor flow traffic statistics. This fine-grained monitoring of flows enables us to better understanding the network protocols and scheme we applied.

Typically, the control plane is made up by the networking device named controller, which is the key part in OpenFlow architecture. The controller knows the overall topology of the network it manages. In addition, it figures out the path needed by routers and switches. When data plane devices have problems with the packets dealing. They will ask the controller for help. The major work for the data plane devices is relatively simple compared to that of controller. Routers and switches are responsible for executing the rule made by controllers and

forwarding datagram in the network.

However, there are still some challenges for OpenFlow architecture both in control plane and data plane. In data plane, one of the most challenging issue is the memory capacity requirements. As the network is growing lager, the data plane devices have to store more rules to handle the packets in the complicated network. The rules and other dynamic flow tables occupy TCAMs(Ternary Content Addressable Memory) in the devices. The TCAMs are precious and expensive. Therefore minimizing the usage of the storage is necessary for the data plane.

In this paper, we first modeled a complete wireless network architecture based on OpenFlow and Cognitive Radio Networks(CRN). The CRN is designed by FCC to improve the frequency utility efficiency by occupying the channel which the licensed user released. We employ the wildcard rule[2] to reduce the TCAM usage of the data plane devices. In CRN, a HDP model is applied to sense and classify the channel into different groups. Then we proposed an algorithm to reduce the number of request to the controller. In the proposed algorithm, we not only record the source-destination pairs, but also collect information from neighbors in order to reduce the request times. The simulation results show that our proposed algorithm improves the throughput and reduces the average packet delay of the network.

The rest of the paper is organizing as follows: In Section 2, we go over some of the related researches. The system design is proposed in section 3. We first introduce the network architecture and then explain the wireless wildcard rule. In section 4, we propose and analyze the controller bottleneck problem within the system. The Control Message Quenching(CMQ) algorithm with information from neighbors and its enhances version are provided in section 5. Section 6 shows the simulation results of channel selection and CMQ algorithms. Section 7 is the conclusion.

II. RELATED WORK

There are plenty of research papers related to OpenFlow networks. Tie Luo in [3] proposed the CMQ algorithm. The algorithm could not only be applied to wired network, but also in sensor openflow network[7]. Min Lan proposed the

DIFNE [9] system to solve the issues related to the rule of data forwarding in the data plane. The author in [5] proposed DeveFlow to devolve the control function back to switches. Other researches including enhance the OpenFlow scalability includes [11] and [12].

III. SYSTEM DESIGN

In this paper, the network system is CRN-based Wireless Mesh Networks. As we mentioned above, nowadays the wireless frequency resources are becoming more and more intense because the number of users is increasing. Also, the we demand higher transmission data rate and better quality of service(QoS). Cognitive Radio Network is a great idea to solve this problem. Here we employ CRN based WMN as our proposed scenario. Meanwhile, OpenFlow is a network architecture which logically separates the data plane and the control plane from each other. Now the OpenFlow architecture is getting more and more popular since it can provide better flow granularity monitoring and flow control methods. OpenFlow can also serve as a new real testbed for new network protocols.

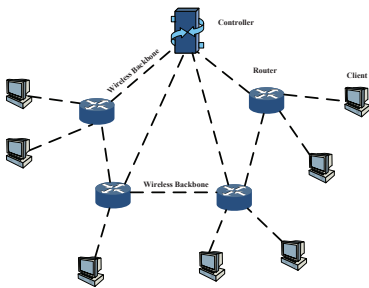


Fig. 1. Small Scale WMN

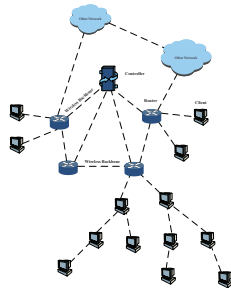


Fig. 2. Large Scale WMN

A. Cognitive Radio Networks(CRN)

Due to the low efficiency of authorized frequency possession, Cognitive Radio has been proposed to improve the efficiency of the frequency usage. A CR system or CR network(CRN) consists primary users(PU) which is also called licensed users and secondary users(SU). The PUs are the authenticated users by the authorities, typically Federal Communications Commission(FCC) in the USA. The basic idea of the CR systems is that the SUs can occupy the under-utilized frequency when PUs don't occupy it. One typical character of the CRN is dynamic spectrum access. In this paper, all the devices including controllers, routers and client terminals are working as secondary users in the Cognitive Radio Networks.

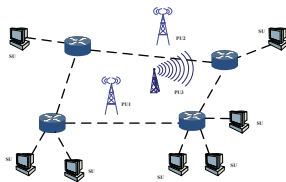


Fig. 3. Cognitive Radio based WMN

There are a large number of research papers regarding on how to detect the PU's activities and decide the channel selection. In our network, we use a Hierarchical Dirichlet Process (HDP) based channel selection model to make channel selection for every device in the network.

B. Wireless Mesh Networks(WMN)

In this paper, we mainly consider the Wireless Mesh Networks(WMN) architecture. WMN is one kind of network architecture with a wireless backbone network and plenty of wireless devices such as mesh routers(MR) and mesh clients(MC). Each MR is the 'header' of a certain region. Each MR is connected to one MR. When a MC has data to send out to the network. Basically, WMN can be classified into two types according to its network size and scalability, namely small scale WMN and large scale WMN. For small scale WMN shown in Fig.1, MCs are only one-hop from the MR in the region. The MR can contact all the MCs in its region within one-hop time. For large scale WMN shown in 2, some MCs are not connected to MR directly. There exits MC to MC connections. In this paper, we assume that the MCs form a 'tree-topology' within a mesh region since it is easy to implement synchronization method and routing algorithm. But the system complexity and performance will be compromised compare to small scale WMN.

C. OpenFlow Architecture

Figure 2 shows the OpenFlow architecture. It is easy to observe that the data plane and the control plane are separated as two layers. The function of the data plane is forwarding data to the destination. The control plane(controller) is responsible for figuring out the routing path for a flow. In our proposed network, the controller is also responsible for the channel designation. In the OpenFlow based WMN, the mesh routers serve as the data plane devices. We have an overall controller for the whole network. The controller will figure out the forwarding rules for each data flow. Forwarding rules are stored into MRs respectively according to the routing path. The Flow Table is one of the most important parts in the OpenFlow based network. Later, we will propose the algorithms to build and update the Flow Table as well as the channel selection.

D. Wireless Wildcard Action Flows

In [2], the author proposed a considerably useful approach called Wildcard identical action flows to simplify the data forwarding process. The fundamental idea is to implement a specified action on every flow. By adding the RouteHeader to each data packet, the router will find it much easier to forwarding the data to its destination. At the same time, the rule assigned to every router is almost the same command which is simple. The RouteHeader indicates the outgoing interface of the router to the next router in its path. This approach is designed for the wired networks. For Node i , the rule stored is:

$$Outgoing_Interface = Number_FirstHeader$$

Check Outgoing_Interface

Case Outgoing_Interface

- 1) forward the datagram to interface 1
- 2) forward the datagram to interface 2
- 3) forward the datagram to interface 3

From the rule above, we know that for every hop, the router pick up the value of the first header, and then forward the data to the interface which has the same number.

Fortunately, we can apply it to wireless networks too. We use the Wireless Wildcard Action Flows, instead of marking the interface number as the RouteHeader. We employ the channel frequency and the ID number(e.g.MAC address) as the RouteHeader. The rules are changed to:

$$Outgoing_Node = ID_FirstHeader$$

$$Channel_Frequency = Frequency_FirstHeader$$

Where $Frequency_FirstHeader$ represents the frequency value picked out from the Frequency Flow Table. Here we simply consider the channel is ready for us. In the next section, we will explain in detail how we could classify the channel by the HDP model.

IV. PROBLEM STATEMENT

The Control-Message Quenching scheme was first proposed in [3]. The goal of the scheme is to reduce the visiting times of the controller so that the controller is not overwhelmed. Now we consider an OpenFlow based wireless mesh network with one controller and N mesh routers. There are m_i mesh clients connected to the i th mesh router, where $1 \leq i \leq N$. The data traffic between the routers is expressed as:

$$\Lambda = \begin{bmatrix} \lambda_{1,1} & \lambda_{1,2} & \lambda_{1,3} & \cdots & \lambda_{1,N} \\ \lambda_{2,1} & \lambda_{2,2} & \lambda_{2,3} & \cdots & \lambda_{2,N} \\ \lambda_{3,1} & \lambda_{3,2} & \lambda_{3,3} & \cdots & \lambda_{3,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \lambda_{N,1} & \lambda_{N,2} & \lambda_{N,3} & \cdots & \lambda_{N,N} \end{bmatrix} \quad (1)$$

Where $\lambda_{i,j}$ is the data arrive rate from $Router_i$ to $Router_j$ and $\lambda_{i,i} = 0$ since the data capacity within the same LAN can be large.

Assume that T_{c_i} is the controller processing time for a request coming from $Router_i$. Typically we have

$$T_{c_i} \gg \frac{1}{\lambda_{i,j}} \quad (2)$$

We know that the clients behind each router can generate data packets that are sent to different routers with probabilities. In order to simplify the analysis, we only consider the static probability. The probability from $Router_i$ to $Router_j$ is represented as

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} & \cdots & p_{1,N} \\ p_{2,1} & p_{2,2} & p_{2,3} & \cdots & p_{2,N} \\ p_{3,1} & p_{3,2} & p_{3,3} & \cdots & p_{3,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{N,1} & p_{N,2} & p_{N,3} & \cdots & p_{N,N} \end{bmatrix} \quad (3)$$

and with the constraints

$$\sum_{j=1}^N p(i, j) = 1 \quad (4)$$

For $Router_i$, during time T_{c_i} , the clients behind the router would still generate data and submit it to the router. The arrive rate, according to (1) and (3), will be

$$r_i = \sum_{j=1}^N \lambda_{i,j} p(i, j) \quad (5)$$

If these data packets have to be processed by the controller in order to get the routing path, during time T_{c_i} , the number of requests the controller will receive is $\frac{T_{c_i}}{r_i}$. Substitute (5) into the expression, we get

$$\frac{T_{c_i}}{r_i} = \frac{T_{c_i}}{\sum_{j=1}^N \lambda_{i,j} p(i, j)} \quad (6)$$

The total number of requests received by the controller is

$$Total_{req} = \sum_{i=1}^N \frac{T_{c_i}}{\sum_{j=1}^N \lambda_{i,j} p(i, j)} \quad (7)$$

This is a huge number. According to [5], in a network with average arrive $\lambda_{i,j} = 85.3k$ packet/sec, the messages received by the controller is about 2.7 Giga/second, which is far beyond the controller's capability to handle the messages.

The author in [2] proposed an algorithm to quench the control messages. The algorithm applies the recording and waiting scheme to the data plane devices. By establishing a table that contains the source and destination pair. The following messages that hit the record will not send the request again to the controller. The algorithm can reduce all the repeated requests thus reducing the burden of the controller significantly.

However, in cognitive radio based wireless mesh networks, the frequency and bandwidth are precious resources. We want to further quench the control messages to give the controller more opportunities to do something valuable.

V. CHANNEL SELECTION AND QUENCHING ALGORITHM

A. HDP based Spectrum Access

In the previous sections we have known that Cognitive Radio(CR) technology could improve the spectrum utilization significantly. Recently, many researchers have proposed different kinds of models for spectrum sensing and accessing. Among those works, Xin-lin Huang in [4] proposed a Hierarchical Dirichlet Process(HDP) based spectrum access scheme. According to [4], the HDP is naturally fit for distributed spectrum sensing.

The 2nd-level DP is

$$G_0 = \sum_{k=1}^{+\infty} \beta_k \delta_{\tilde{\lambda}^k}, \beta|\gamma \sim GEM(\gamma), \tilde{\lambda}^k \sim H \quad (8)$$

The 1st-level DP:

$$G_j = \sum_{t=1}^{+\infty} \tilde{\pi}_{jt} \delta_{\tilde{\lambda}_{jt}}, \tilde{\pi}_j | \alpha \sim GEM(\alpha), \tilde{\lambda}^{jt} \sim G_0 \quad (9)$$

By applying the HDP model, the scheme could automatically sense the channel and classify the channel into a certain group. One of the major benefit of HDP model is that we don't have to indicate how many groups there. The model will update the group number as the probability indicated. The hidden parameter in this system is

$$\lambda^{ji} | G_j, X^{ji} | \lambda^{ji} \sim \prod_{k=W_H T/\pi}^{k=W_H T/\pi} Exponential(\lambda_k^{ji}) \quad (10)$$

The CR channel is recognized as a Rayleigh channel. After applying HDP model, we can find out the nodes which share the similar channel environment(i.e. channel frequency). One explanation of the HDP model is named Chinese Restaurant Franchise(CRF).

$$p(\lambda_i | \lambda_{-i}, \gamma, H) = \frac{\gamma}{\gamma + N - 1} H + \frac{1}{\gamma + N - 1} \sum_{k=1, k \neq i}^N \delta_{\lambda^k} \quad (11)$$

It shows that SUs can either be classified into one existed groups with probability $\frac{\gamma}{\gamma+N-1}$ or into a brand new group with probability $\frac{1}{\gamma+N-1}$. In the simulation section, we will show the results of the channel classification.

B. Quenching Algorithm

In the previous algorithm, only the repeated requests are quenched. We only record the source and destination pair in the local router for searching. Here we further explore the neighborhoods' resources in their flow tables. The basic idea of our proposed algorithm is that we can not only record the historical pair. When the new packets failed to hit the record. The router could automatically ask its neighbors for help. That is, one router could share its own flow table with its neighbors. Here are the rules involved in the idea.

- 1) Every router know its neighbors' ID and location.
- 2) The controller inform the router about the topology when there are changes in the network.
- 3) Each router can only ask for routing information from the nodes within one hop.
- 4) The items in the flow table in each router only exists for a certain period of time and will be killed by the router when the time is over.

Assume that each router maintains and updates a table list L . We denote a path from source(s) to destination(d) as $\langle s, d \rangle$. Refer to Algorithm 1.

When we choose Algorithm 1, we could quench most of the redundant requests messages to controller. Thus saving plenty of controller's computational resources. As we mentioned in the previous sections, each router has the probability to visit all other routers, when the router has the opportunity to get information from its neighbors, the probability of hitting the

Algorithm 1 Control-Message Quenching(CMQ) with Information From Neighbors

```

1: L: = empty set
2: for each incoming packet do
3:   Check the destination node and mark it as  $d_{com}$ ;
4:   Look up the flow table to find out whether there
   is a record matching the destination node and path  $\langle$ 
 $s, d \rangle = \langle s, d_{com} \rangle$ 
5:   if Matched record found then
6:     if The path is ready then
7:       Handle the packet as the rule indicated
8:     else
9:       Wait for the path
10:    end if
11:   else
12:     Ask neighbors to find out the same path
13:     if Matched record found then
14:       Extract the rule
15:       Copy the rule and paste to local flow table
16:       Handle the packet as the rule indicated
17:     else
18:       Consult the controller for path configuration
19:       Record the destination node and update list  $L$ 
20:     end if
21:   end if
22: end for
23: if Receive an answer from the controller then
24:   Record the path
25:   Record the node ID along the path
26:   Update list  $L$ 
27: end if
28: if Item time expires then
29:   Kill item
30: end if

```

record becomes much higher. For $Router_i$, assume that the neighbors of $Router_i$ form the set S . The hitting probability to $Router_j$ is

$$P_{hit(i,j)} = p_{i,j} + \sum_{l \in S} p_{l,j} \quad (12)$$

In this way, the network would have better throughput and delay performance than the previous algorithms. In the simulation section, we will show the simulation results of the algorithm.

C. Enhanced CMQ Algorithm

The previous algorithm we have proposed do improve the overall performance of the network. It also reduces the burden of the controller. From equation (12) we know the reason is that we increase the hitting probability. However, there is still some drawbacks with this algorithm. If we want the packets to go through the shortest path, we still have to ask the controller for help. Another issue is that in some cases the controller figures out a long path that goes through some routers. When

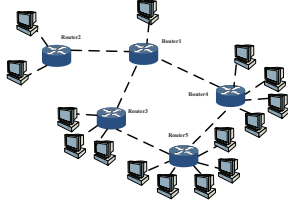


Fig. 4. Simulation Topology

some of the routers in the middle of the long path have data to send out, it has to ask the controller again for path configuration. This is also repeated work for the controller. In order to avoid this issue, we proposed an enhanced CMQ algorithm. Based on Algorithm 1, when the packets go to a remote router, it can bring the path destination information to the routers within the path. As the packet goes from one router to another, the destination and path would be stored in the routers that the packet passed by. In this way, the routers in the same path all have the information to the destination. Meanwhile, the routers have higher probability to go through the shortest path than our previous proposed algorithm does. The hitting rate goes higher. The detail is illustrated in the table Algorithm 2.

The Algorithm 2 is designed for the routers. For the controller, it should have a function to ask for the routers for the paths. Compare to the time consumed in path computation and configuration, Asking routers to check if there is existed path costs far less time and energy. The processing detail is illustrated in the table Algorithm 3.

In the next section, we will provide the simulation results for the algorithms described above.

VI. NUMERICAL SIMULATION

In this section, simulation result are provided. We first illustrate the simulation parameters, and then show the figures.

A. Simulation Parameters

The simulation was designed according to the previous proposed scenario. Shown in Fig.4, the network topology includes one controller, five mesh routers and 15 mesh clients. All the connections between the devices are wireless. The controller manages the whole topology. Each mesh router are marked by a number from 1 to 5. For router i , we simply add i clients behind it. Each client could contact any of the remaining in the whole network. The channel bandwidth between the Controller and Routers is set to 20Mbps. Generally, the data traffic between Controller and Routers is far less than that between routers and clients. The data link capacity between one router and another is 50Mbps. For simplicity, all the clients are both senders and receivers. One client could transmit data to another one randomly with equal probability. The size of one data packet is 1500Byets, which is the same as the size of IP packet in Internet. The processing time of the controller is 10 times the time of one packet transmission.

Algorithm 2 Enhanced Control-Message Quenching(CMQ) Based on Algorithm 1

```

1: L: = empty set
2: if Receive a passing packet then
3:   Check the destination from the packet
4:   if Already have the destination record then
5:     Handle the packet by the FirstHeader
6:   else
7:     Record the destination and the path
8:   end if
9: end if
10: for each incoming packet do
11:   Check the destination node and mark it as  $d_{com}$ ;
12:   Look up the flow table to find out whether there
   is a record matching the destination node and path  $<$ 
 $s, d > = < s, d_{com} >$ 
13:   if Matched record found then
14:     if The path is ready then
15:       Handle the packet as the rule indicated
16:     else
17:       Wait for the path
18:     end if
19:   else
20:     Ask neighbors to find out the same path
21:     if Matched record found then
22:       Extract the rule
23:       Copy the rule and paste to local flow table
24:       Handle the packet as the rule indicated
25:     else
26:       Consult the controller for path configuration
27:       Record the destination node and update list  $L$ 
28:     end if
29:   end if
30: end for
31: if Receive an answer from the controller then
32:   Record the path
33:   Record the node ID along the path
34:   Update list  $L$ 
35: end if
36: if Item time expires then
37:   Kill item
38: end if

```

Three indicators are used for comparison, i.e. the average network throughput, average packet delay and the number of requests to the controller.

B. Simulation Results

Fig. 5 shows the simulation result of the average network throughput under different schemes. It's obvious that the CMQ algorithm with information from neighbors outperforms that of the conventional CMQ algorithm. The enhanced CMQ algorithms with information from neighbors works even better. With the increasing data generation rate, the throughput is also rising. When the average input data arrival rate is larger

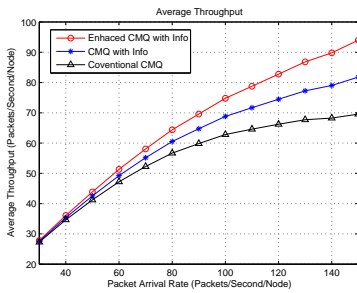


Fig. 5. Throughput Performance of Different Schemes

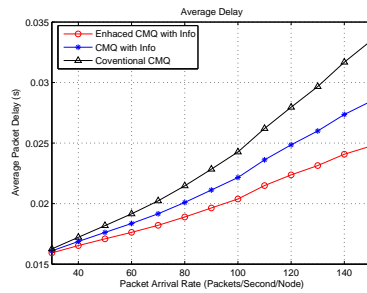


Fig. 6. Average Packet Delay of Different Schemes

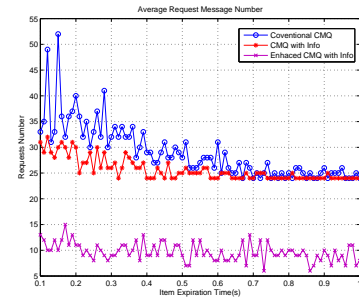


Fig. 7. Number of Requests to Controller under different expiration time

Algorithm 3 A Function added to Controller

```

1: if Request from  $i$  to  $j$  arrived then
2:   Ask routers for path
3:   if Path found from  $k$  to  $j$  then
4:     ComputePath( $i,k$ )
5:   else
6:     ComputePath( $i,j$ )
7:   end if
8: end if

```

than 130 packets/second, there is a ceiling for the conventional CMQ algorithm. However, with information from neighbors, the throughput could continue increasing.

Fig. 6 presents the average packet delay in the network. As the data generate rate is increasing, the network is becoming crowded. Still, our proposed algorithms work better than the conventional CMQ algorithm. The packet delay of the proposed algorithms are lower. The delay of the conventional CMQ increases faster.

In Fig. 7, we simulate the number of requests to controller with respect to the expiration time of the source-destination pair records. As we can see from the figure, the number of requests tend to be a fixed value as the holding time of the items in flow table increases. This is because when the holding time is long, more flows are getting easily to hit the record in the flow table. Thus less requests the controller will receive. From the result we find that the number of our proposed algorithms decreases faster. In addition, when we apply the enhanced CMQ algorithm, the minimum number is less than other algorithm. The number changes slightly. In this way, we can set the expiration time to a low level to save more TCAMs in data plane devices.

VII. CONCLUSION

In this paper, we first introduce an OpenFlow based Wireless Mesh Network system. Some of the major challenging issues are presented. An HDP model is introduced to sense and classify the channel for WMN. We also modified the wildcard rule to make it useful in wireless systems. Then, we analyze the problem and proposed an advanced CMQ algorithm to quench the number of requests to the controller. Simulation results

show that our proposed algorithms work better compared to the conventional one.

REFERENCES

- [1] N. McKeon, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," in *ACM SIGCOMM Computer Communication Review*, April 2008.
- [2] Iyer, A.S.; Mann, V.; Samineni, N.R., "SwitchReduce: Reducing switch state and controller involvement in OpenFlow networks," *IFIP Networking Conference*, 2013, vol., no., pp.1,9, 22-24 May 2013
- [3] Luo, Tie, et al. "Enhancing responsiveness and scalability for OpenFlow networks via control-message quenching." *ICT Convergence (ICTC), 2012 International Conference on. IEEE*, 2012.
- [4] Xinlin Huang, Fei Hu, Jun Wu, "Intelligent Cooperative Spectrum Sensing via Hierarchical Dirichlet Networks," *IEEE Journal on selected areas in communication*.
- [5] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," in *ACM SIGCOMM*, 2011.
- [6] Open Networking Foundation, "OpenFlow switch specification, April 16, 2012
- [7] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor OpenFlow: Enabling software-defined wireless sensor networks," *IEEE Communications Letters*, 2012, to appear.
- [8] Open Networking Foundation, "Software-defined networking: The new norm for networks, white paper, April 2012.
- [9] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," in *ACM SIGCOMM*, 2010.
- [10] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an operating system for networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105110, 2008.
- [11] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: a distributed control platform for largescale production networks," in *The 9th USENIX conference on Operating systems design and implementation (OSDI)*, 2010, pp. 16.
- [12] A. Tootoonchian and Y. Ganjali, "HyperFlow: a distributed control plane for openflow," in *INM/WREN. USENIX Association*, 2010.