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

**Xin Li, Xiaoyan Hong, Yu Lu, Fei Hu, Ke Bao and Sunil Kumar**

## 1 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.

X. Li(✉) · Y. Lu · F. Hu · K. Bao
Electrical and Computer Engineering, University of Alabama, Tuscaloosa, AL, USA
e-mail: xli120@crimson.ua.edu

X. Hong
Computer Science, University of Alabama, Tuscaloosa, AL, USA

S. Kumar
Electrical and Computer Engineering, SDSU, San Diego, CA, USA



S. Latifi (ed.), *Information Technology New Generations*,
Advances in Intelligent Systems and Computing 448,
DOI: 10.1007/978-3-319-32467-8_7

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.

## 2 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].

## 3 System Design

### *3.1 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

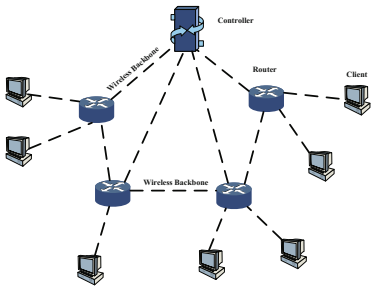


**Fig. 1** Small Scale WMN

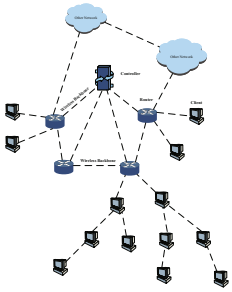

**Fig. 2** Large Scale WMN

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 Fig. 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.

## *3.2 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:

$$Outgoing\_Interface = Number\_FirstHeader$$

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:

$$Outgoing\_Node = ID\_FirstHeader$$

$$Channel\_Frequency = Frequency\_FirstHeader$$

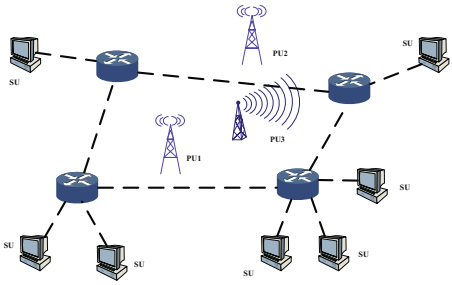


**Fig. 3** Cognitive Radio based WMN

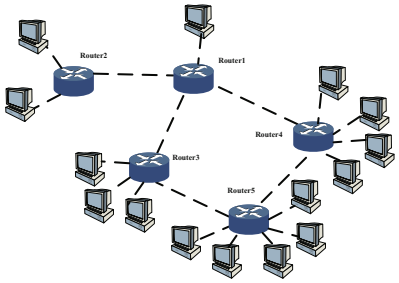


**Fig. 4** Simulation Topology

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.

## 4 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:

$$\Lambda = \begin{bmatrix} \lambda_{1,1} & \lambda_{1,2} & \cdots & \lambda_{1,N} \\ \lambda_{2,1} & \lambda_{2,2} & \cdots & \lambda_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{N,1} & \lambda_{N,2} & \cdots & \lambda_{N,N} \end{bmatrix} \tag{1}$$

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

$$T_{c\_i} >> \frac{1}{\lambda_{i,j}} \tag{2}$$

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

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

and with the constraints

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

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

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

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

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

The total number of requests received by the controller is

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

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.

# 5 Channel Selection and Quenching Algorithm

## 5.1 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

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

The 1st-level DP:

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

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

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

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).

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

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.

## *5.2 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.

Every router know its neighbors' ID and location.
The controller inform the router about the topology when there are changes in the network.
Each router can only ask for routing information from the nodes within one hop.
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 $< s, d >$. 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

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

```
L: = empty set
for each incoming packet do
    Check the destination node and mark it as d_com;
    Look up the flow table to find out whether there is a record matching
  the destination node and path < s,d >=< s,d_com >
    if Matched record found then
        if The path is ready then
            Handle the packet as the rule indicated
        else
            Wait for the path
        end if
    else
        Ask neighbors to find out the same path
        if Matched record found then
            Extract the rule
            Copy the rule and paste to local flow table
            Handle the packet as the rule indicated
        else
            Consult the controller for path configuration
            Record the destination node and update list L
        end if
    end if
end for
if Receive an answer from the controller then
    Record the path
    Record the node ID along the path
    Update list L
end if
if Item time expires then
    Kill item
end if
```

**Fig. 5**

**Algorithm 2** A Function added to Controller

```
if Request from i to j arrived then
    Ask routers for path
    if Path found from k to j then
        ComputePath(i,k)
    else
        ComputePath(i,j)
    end if
end if
```

**Fig. 6**

all other routers, when the router has the opportunity to get information from its neighbors, the probability of hitting the 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

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

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.

### 5.3 *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 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.

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 2.

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

## 6 Numerical Simulation

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

## 6.1 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.

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

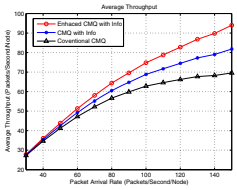


**Fig. 7** Throughput Performance of Different Schemes

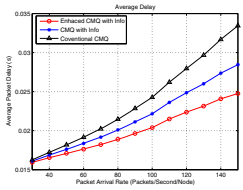


**Fig. 8** Average Packet Delay of Different Schemes

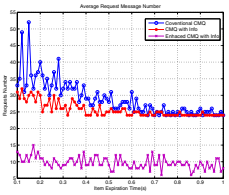


**Fig. 9** Number of Requests sent to Controller

## 6.2 Simulation Results

Figure 7 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 than 130 packets/second, there is a ceiling for the conventional CMQ algorithm. However, with information from neighbors, the throughput could continue increasing.

Figure 8 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. 9, 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 tent 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.

## 7 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. McKeon, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: Enabling Innovation in Campus Networks. 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. In: IFIP Networking Conference, 2013, pp. 1–9, May 22–24, 2013
3. Luo, T., et al.: Enhancing responsiveness and scalability for OpenFlow networks via control-message quenching. In: 2012 International Conference on ICT Convergence (ICTC). IEEE (2012)
4. Huang, X., Hu, F., Wu, J.: Intelligent Cooperative Spectrum Sensing via Hierarchical Dirichlet Networks. IEEE Journal on Selected Areas in Communication
5. Curtis, A.R., Mogul, J.C., Tourrilhes, J., Yalagandula, P., Sharma, P., Banerjee, S.: DevoFlow: Scaling flow management for high-performance networks. ACM SIGCOMM (2011)
6. Open Networking Foundation. OpenFlow switch specification, April 16, 2012
7. Luo, T., Tan, H.-P., Quek, T.Q.S.: 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, April 2012 (white paper)
9. Yu, M., Rexford, J., Freedman, M.J., Wang, J.: Scalable flow-based networking with DIFANE. ACM SIGCOMM (2010)

10. Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., Shenker, S.: NOX: Towards an operating system for networks. SIGCOMM Comput. Commun. Rev. **38**(3), 105–110 (2008)
11. Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., Shenker, S.: Onix: a distributed control platform for largescale production networks. In: The 9th USENIX Conference on Operating Systems Design and Implementation (OSDI), pp. 1–6 (2010)
12. Tootoonchian, A., Ganjali, Y.: HyperFlow: a distributed control plane for openflow. In: INM/WREN. USENIX Association (2010)