A Robust Controllers' Negotiation Model in Multi-Domain SDN Environments

Yen-Chiu Chen* & Hui-Ching Hsieh**

*Department of Information and Communications, Research Laboratories (ICL), Industrial Technology Research Institute (ITRI), Hsinchu, TAIWAN, R.O.C. E-Mail: amanda.chen{at}itri{dot}org{dot}tw
**Department of Information Communication, Hsing Wu University, New Taipei City, TAIWAN, R.O.C.
E-Mail: luckyeva.hsieh{at}gmail{dot}com

***Abstract*—**In multi-domain SDN environments, the controllers need to exchange information frequently. In real situation, there may have some faulty controllers. Besides, the connection between controllers may be disturbed. All this faulty components may cause that fault-free controllers getting the wrong information, and the network cannot work well. Hence, in this paper, a robust controllers' negotiation mechanism has been proposed. Based on the mechanism, all fault-free controllers can efficiently and robustly to get a common score about the information which are needed to be shared by at most three rounds of score exchange phases. Moreover, when there are at most $\lfloor (n-1)/3 \rfloor$ allowable faulty controllers, the time complexity and the space complexity of generating the correct score are $O(n^2)$ and $O(n^2)$ respectively, where n presents the number of total controllers.

***Keywords*—**Controller Negotiation; Heterogeneous SDN Multi-Controller; Network View; OpenFlow Protocol; Open Networking Foundation; Software-Defined Networking.

***Abbreviations*—**Autonomous System (AS); Network Operation System (NOS); Open Networking Foundation (ONF); Software-Defined Networking (SDN); Venture Capital (VC).

I. INTRODUCTION

IN recent years, the concept of Software-Defined Networking (SDN) proposed by Stanford University inspires many research workers to rethink the management and design of current network because of its novel architecture. There are three layers defined in SDN architecture that are application layer, control layer and infrastructure layer shown in Figure 1. The architecture separates control planes and data planes into control layer and infrastructure layer respectively. Moreover, new network services can be deployed in application layer without binding to the other two layers. The most popular southbound API is OpenFlow protocol [1] which is endorsed by the Open Networking Foundation (ONF) [2]. A controller and OpenFlow switches communicate with each other followed by OpenFlow protocol. So far, applications use RESTFul API to communicate with a controller where RESTFul API is a way to communication but not a standard yet for SDN Northbound API. Up to the present, there is no acknowledged northbound API but ONF has a Northbound Working Group which is cooking for a common northbound framework and mechanism rather than defining all possible Northbound APIs.

According to the SDN architecture, that is a new situation for network evolution for applications and network devices which can be designed and deployed separately and managed by a central controller in a domain. This hot research topic is not only in campus network [Nick McKeown et al., 3] but also in industry; SDN Market Sizing Report [Central, 4] addresses that Venture Capital (VC) funding spent less than $10 million US dollars in SDN-focused companies in 2009, but the VC funding has increased to $500 million US dollars in 2013 that is 50X increase in the growth of the SDN market. This report also points out that the potential SDN revenues will grow to more than $35 billion US dollars in 2018.

The development of SDN environment is starting up. Most researches are focused on single SDN controller and deployed in campuses or field trial. However, there will be some issues occurred at single SDN controller environment such as scalability issues, performance issues, some attach behaviors to the single controller to cause a single of failure and so on. How to design and deploy multi-controller SDN environment must be the next trend. The conversation and negotiation between controllers to provide the global network view between more than one controller will be the first challenge that should be solved, where the conversation between controllers represents the communication or change

some information and the negotiation means that some controllers make a decision together after having conversations. Hence, we propose a robust and efficient negotiation model among multi-domain SDN environments in this paper.

The remainder of this paper is organized as follows. Some previous results and addressed problems in multi-controller SDN environments are introduced in Section 2. In Section 3, we introduce a robust and efficient conversation and negotiation model among multiple SDN controllers. We prove the correctness and complexity of the model in Section 4. Finally, Section 5, we give conclusion.

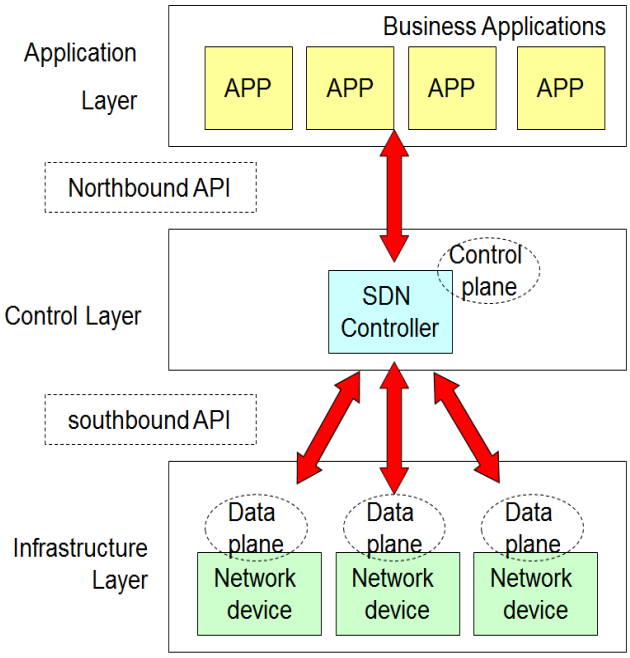

Figure 1: SDN Architecture

II. Related Work

In contrast to traditional functions in each switch, the management functions handled by Network Operation System (NOS) are decoupled from the switch and to be a central module in a server which is named as a controller. Hence, all managements, such as topology discovery, routing policy, flow dispatch, are centralized to a controller. When a new flow comes to a new domain by an ingress switch, the SDN-enabled ingress switch will send the flow to the controller by pack-in instruct. The controller will write routing information, like source/distinction MAC address/IP/port, into the flow table, and then the assigned switches will forward the flow according to the flow table.

The most advantage of the central management scheme is that a controller can control all switches, flows and network status in its domain. However, with the popularity of SDN-enabled switches deployed, how many SDN-enabled switches should be controlled by a controller and the performance issues under a controller scheme are important and interesting problems [Brandon Heller & Nick McKeown, 5]. With the network evolution, the concept of SDN must be deployed from campus network into enterprise network and wide area network, one controller control everything will be more difficult and complexity. Hence, some researches in multi-controller SDN environment are proposed. In 2010 year, a scalable flow-based networking with DIFANE was proposed by Minlan Yu et al., [6]. In DIFANE architecture, controllers install authority rules to authority switches and then non-authority switches just handle packs in data plane. That is, authority switches still share some responsibility for the controller to increase the scalability. Different from designing assistant controllers like authority switches, Onix [Teemu Koponen et al., 7] is a distributed control platform delivered by Google, NEC, Nicira Networks, and UC Berkeley together. In Onix architecture, the control plane is implemented as a distribution system. An Onix instance uses a NIB describes a graph of a network topology, where the graph includes all network entities in the corresponding network domain. In 2012 year, Yeganeh & Kandoo [8] proposed a two-layer SDN controller architecture named as Kandoo. Each local controller handles SDN-enabled switches in a sub-domain, and a root controller controls all local controllers to realize a scalable SDN environment. In 2013 year, Google proposed an experience with globally-deployed software defined WAN, named as B4 [Sushant Jain et al., 9], which is constructed between its own data centers. In B4 architecture, one site represents one data center. There are one or more than one controller in a sit, and those controllers use Paxos [Tushar D. Chandra et al., 10] to elect one controller to handle the switches in the site. Moreover, a logically centralized TE server and a SDN gateway are used to collect all traffic information of all site controllers periodically and to make some decisions for traffic engineering. More multi-controller SDN architectures and research results are in [Phemius et al., 11; Amin Tootoonchian & Yashar Ganjali, 12; Advait Dixit et al., 13].

However, there are still some open problems that are not addressed in the above-mentioned papers. Most of previous results discussed distributed control plane but with shared storage like shared memory or distributed database under the same operators or homogeny service providers [Teemu Koponen et al., 7; Sushant Jain et al., 9; Advait Dixit et al., 13]. Moreover, under hierarchy controller architecture, the root controller of the hierarchy controller architecture may not only be attacked as a single of failure but also have high space complexity or complex data structure to record all network status of child controllers. How to provide a robust and efficient mechanism to exchange and share all network status as a global view under distributed controllers is an import thing in heterogeneous multi-domain SDN environments.

In this paper, we propose a robust and efficient negotiation model among multi-domain SDN environments. Here, multi-domain SDN environments include multiple heterogeneous SDN controllers or anyone Autonomous System (AS) controlled by a controller. We call this model as Controller Negotiation Model and using ConNe for short.

The main contributions of this paper:

- We propose a robust and efficient conversations and negotiation model among multiple SDN controllers. This model is called as controller negotiation and ConNe for short.
- When there are at most $\lfloor (n\text{-}1)/3 \rfloor$ allowable faulty controllers, the time complexity of generating the correct score is $O(n^2)$ in $O(n^2)$ space complexity by at most three rounds of score exchange for the ConNe model.
- The benefit of ConNe model is that each controller records a topology of all controllers without other status to decrease space complexity and takes robust and efficient negotiation with neighbor controllers in run time.
- Some applications, such as load balancer or service chaining services or QoS-based services, need global network status like link utilization among intra-domain or inter-domain to make policy decisions. ConNe model can provide such applications real-time global and consistent multi-domain network view.

III. CONTROL NEGOTIATION MODEL

In this section, the proposed negotiation model has been introduced.

3.1. Controllers' Topology Discovery

In conventional network, there are many mature protocols such as LLDP [Kempf et al., 15; Wei Xiao et al., 16], and BGP [Sapegin et al., 17; Jian Mai & Jiang Du, 18] to support network elements or autonomous systems to discover network topology. In multi-controller SDN environment, basically, a sample score exchange between multi-controllers can be implemented by Hello, Update, Error, Bye scores, where controllers use Hello score to notify neighbor controllers about its alive, and use Update score with each other to update new domain status. When some unexpected event occurs, controllers use Error score to notify neighbor controllers or in reverse. Any controller can disjoint the network with other controller by Bye score. In B4 [Sushant Jain, 9] architecture, controllers in different sites use iBGP to take a conversation and use eBGP to collect routing information and status from switches. In the other way, Wun-Yuan Huang et al., [14] designed and implemented a topology discovery system by extending LLDP. Consequently, a topology discovery between multi-controllers is at a possible situation and this paper is not addressed the topology discovery methods, but assume that all controller record the controller topology of all controller domains. Note that the controller topology does not include all switch topology in each controller domain.

3.2. Controller's Conversation

What information should be communicated or exchanged between controllers is an interesting problem. To our opinions, the information between controllers' conversation should be depended on the requirements from an application which is upon on a controller. For QoS-based applications, high available bandwidth, less delay time or fewer packets lost maybe considerate more. Take online conference meeting as an example, in order to ensure the quality of connection, the available bandwidth between controllers may be the first priority that should be guaranteed. Different applications have different parameters of controllers' conversation. One application may have more than one requirements should be guaranteed. Hence, the information can be normalized as a score for the controller conversation. Moreover, one robust and efficient controller negotiation based on a normal will be introduced in the next sub-chapter.

3.3. Controllers' Negotiation Mechanism

In real situation, there may have some controllers be broken, or the connection between the controllers may be disturbed. This kind of problem may cause that the negotiation results between controllers are wrong, and the network cannot work well. In order to improve the efficiency and correctness of the negotiation result, a robust controllers' negotiation mechanism is proposed. There are two phases for the controllers' negotiation mechanism: 1) exchanging score phase and 2) making decision phase. The goal of exchanging score phase is to collect and store the score in the controller's s-tree (s-tree is a tree data structure used to store the received scores) by running three rounds of score exchange. Upon completion of the exchanging score phase, the making decision phase is invoked. There are three objectives of this phase, shown as follow:

1. To find the reliable controllers.
2. To apply the replacing process to replace the scores received from the un-reliable controllers.
3. To apply the VOTE function from the third level to root *s* of each fault-free controller's s-tree and determine the final score.

In the ConNE model, each controller records a topology of all controllers without other status at a time and uses $O(n^2)$ space complexity to record scores by at most three rounds of exchange. Moreover, the time complexity of generating the correct score is $O(n^2)$ when there are at most $\lfloor (n\text{-}1)/3 \rfloor$ allowable faulty controllers. ConNE model provides an efficient method to reduce the time for exchanging score and making decisions, and it also reduces the space of storing information for each controller. The related correctness will be proved in following section.

Notation:

- $maj_3(ax)$: The majority score of the third level of the sub-tree which is expanded from vertex $v(ax)$.
- $\#maj_3(ax)$: The total number of the scores that are equal to $maj_3(ax)$ (For each sub-tree which is expanded from vertex $v(ax)$).
- RLP_x: The Reliable-Like Controller set for the sub-tree which is expanded from $v(ax)$.
- $\#p_z$: The total number of times that the controller z appears in all Reliable-Like Controller set.
- RP_x: The *reliable controllers* in each sub-tree which are expanded from vertex $v(ax)$.
- $maj_3(RP_x)$: The majority score of the *reliable controllers* for the sub-tree which is expanded from $v(ax)$.

***Exchanging score phase*:**

Round 1 do:

- The source controller s broadcasts its initial score $v(s)$ to other controllers and itself.
- Each controller stores $v(s)$ in the root of its s-tree;

Round 2 do:

- Each controller broadcasts the score which is stored in the root of its s-tree to other controllers and itself.
- Each controller stores the received scores (Received from controller x) in the second level of its s-tree (Use $v(ax)$ to represent the received scores).

Round 3 do:

- Each controller broadcasts the scores which are stored in the second level of its s-tree to other controllers and itself.
- Each controller stores the received scores (Received from controller y) in the third level of its s-tree (Use $v(axy)$ to represent the received scores).

***Making decision phase*:**

1. Finding out the *reliable controllers*

For each sub-tree of vertex $v(ax)$ in the second level of s-tree.
{
If ($v(ax) = maj_3(ax)$ and $\#maj_3(ax) \geq (n-\lfloor(n-1)/3\rfloor-1)$)
{
Add controller x into RLP_x.
For each vertex which is expanded from the vertex $v(ax)$
{
If $v(axy) = v(ax)$ then
Add controller y to RLP_x.
}
}
}
For each controller z (denoted as p_z)
{
Count $\#p_z$ from all Reliable-Like Controller set
If ($\#p_z \geq (n-\lfloor(n-1)/3\rfloor-1)$)
{
Controller z is a *reliable controller*
}
}

2. The replacing process.

For each vertex which is expanded from $v(ax)$
{
If controller y is not a *reliable controller* and $v(axy) \neq maj_3(RP_x)$) then
$v(axy) = maj_3(RP_x)$
}

3. Apply function *VOTE*(s) to root of each controller's s-tree and common score, *VOTE*(s) is obtained.

Figure 2: The Details of the Proposed Mechanism

IV. CORRECTNESS

In this section, we will prove the correctness and complexity of the proposed mechanism.

4.1. Correctness of the Proposed Mechanism

Each fault-free controller should be insulated from the influence of a faulty controller. In our mechanism, the influence of a faulty controller can be removed in the making decision phase, then the collected scores of fault-free controllers are uninfluenced and an agreement is reached. The following lemmas, corollaries and theorems are used to prove the correctness of the proposed mechanism.

Lemma 1: All correct vertices of the s-tree are common

Proof: In the making decision phase, there are no repeatable vertices in the s-tree by deleting the repeating vertices. At the second and third level, the correct vertex α has at least n-1 children in which at least n-$\lfloor(n\text{-}1)/3\rfloor$ children are correct. The score of these n-$\lfloor(n\text{-}1)/3\rfloor$ correct vertices are in common, and the majority score of vertex α is common. The correct vertex α is common in the s-tree, if the level of α is less than three. Thus, all correct vertices of the s-tree are common.

Lemma 2: The common frontier does exist in the s-tree

Proof: There are three vertices along each root-to-leaf path of the s-tree at any time, in which the root is labeled by the source name, and the others are labeled by a sequence of group names. Because at most f_m controllers can fail, at least one vertex is correct along each root-to-leaf path of the s-tree. By lemma 1, the correct vertex is common, and the common frontier exists in each fault-free controller's s-tree.

Lemma 3: Let α be a vertex, if there is a common frontier in the sub-tree rooted at α, then α is common

Proof: By induction on the height of α.

If the height of α is 0 and the common frontier (α itself) exists, α is common.

If the height of α is l, the children of α are all in common, based on the induction hypothesis with the height of the children at l-1; therefore, vertex α is common.

Lemma 4: The reliable controller can be obtained

Proof: Since there are n-$\lfloor(n\text{-}1)/3\rfloor$ fault-free controllers can send the received scores to others correctly and honestly. Therefore, there will have at least n-$\lfloor(n\text{-}1)/3\rfloor$ vertices (In the third level of the s-tree) that have the same score for each sub-tree which is derived from the corresponding vertex in the second level of the s-tree. For each sub-tree in the third level of the s-tree, if $maj_3(ax) = v(ax)$ and $\#maj_3(ax) \geq n$-$\lfloor(n\text{-}1)/3\rfloor$ - 1, it means that controller x (The score $v(ax)$ is received from controller x.) sends to at least n-$\lfloor(n\text{-}1)/3\rfloor$ same scores to others and there have n-$\lfloor(n\text{-}1)/3\rfloor$ controllers y (The score $v(axy)$ is received from controller y.) agree that controller x is reliable. Thus, controller x and controller y (which has the same score with controller x) can be added to the RLP. If $\#p_z$ (Each controller z in the RLP) $\geq n$-$\lfloor(n\text{-}1)/3\rfloor$, it presents that the number of controllers agree controller z is reliable are in the majority. Hence, the reliable controller z can be obtained in our protocol.

Lemma 5: The scores replaced by the majority score of reliable controllers are common

Proof: By Lemma 1, 2 and 3, all correct vertices of the s-tree are common, and each fault-free controller's s-tree also has the same common frontier. Furthermore, at least n-$\lfloor(n\text{-}1)/3\rfloor$ controllers are fault-free. Hence, all these fault-free controllers must be reliable controllers. Thus, the majority score of these reliable controllers for each sub-tree in the third level must be common. Therefore, the scores replaced by the majority score of reliable controllers are common.

Corollary 1: If the common frontier exists in the s-tree, then the root is common

Theorem 1: All fault-free controllers can determine the common set of reliable controllers

Proof: By Lemma 1, Lemma 2, Lemma 3, Lemma 4 and Corollary 1, the theorem is proven.

Theorem 2: The root of a fault-free controller's s-tree is common

Proof: By Lemma 1, Lemma 2, Lemma 3, Lemma 4, Lemma 5, Corollary 1 and Theorem 1, the theorem is proven.

Theorem 3: All controllers can get a common score agreement

Proof: To prove the theorem, we show that the proposed mechanism can meet the following requirements.

- Root s is common.
  By theorem 1, (ba_1') is satisfied.
- VOTE(s) = v_s for all fault-free controllers, if the source is fault-free.

If the source is fault-free, then it broadcasts the same initial score v_s to all controllers. The score of correct vertices for all fault-free controllers' s-tree is v_s. Thus, each correct vertex of the s-tree is common (Lemma 1), and its score is v_s. Because the source is fault-free, the root of the s-tree is also a correct vertex by Lemma 1. By Theorem 1, this root is common. The computed score vote(s) = v_s is stored in the root for all fault-free controllers. Thus, this requirement is satisfied.

4.2. Complexity of the Protocol

The complexity of the mechanism is evaluated in terms of 1) the number of rounds about score exchange, 2) the number of allowable faulty controllers and 3) the quantity of the scores that are generated during the execution of the proposed mechanism.

Lemma 6: The scores sent by the fault-free controllers are same as the majority score after applying the VOTE function

Proof: There are at least n-$\lfloor(n\text{-}1)/3\rfloor$ fault-free controllers in the system. All fault-free controllers transmit their scores to the others correctly. In each round of message exchange n-$\lfloor(n\text{-}1)/3\rfloor$ fault-free controllers can receive these scores and send them again. Then, the majority scores which are applied, using the VOTE function for the (i+1)th ($1 \leq i \leq \lfloor(n\text{-}1)/3\rfloor$) level of the s-tree must be equal to the scores in the ith level

of the s-tree. It is needless to send the scores for $\lfloor (n\text{-}1)/3 \rfloor + 1$ rounds, if the sender is a fault-free controller.

Theorem 4: Three rounds of score exchange is minimum

Proof: 1) In the second round of the message exchange phase, the scores are sent and received correctly by other $n\text{-}\lfloor (n\text{-}1)/3 \rfloor$ fault-free controllers, if the sending controllers are fault-free. All these correct scores are sent in the third rounds of the exchanging message phase. Then, these three level s-tree can be used to find the reliable controllers. Here, they have n-1 number of vertices in the second level of the s-tree. Thus, there will have n-1 number of RLP_x ($1 <= x <= (n\text{-}1)$). If the frequency of the controller x appearing in all RLP is greater than or equal to $n\text{-}\lfloor (n\text{-}1)/3 \rfloor$, it implies that $n\text{-}\lfloor (n\text{-}1)/3 \rfloor$ controllers believe that controller x is reliable. Hence, the scores sent from the reliable controllers can be used to replace the scores received from the un-reliable controllers. Based on lemma 5, there is no need to send the scores for $\lfloor (n\text{-}1)/3 \rfloor + 1$ rounds if the sender is a fault-free controllers.

2) By the lemma 1, the ambiguity due to at most $\lfloor (n\text{-}1)/3 \rfloor$ faulty controllers can be resolved. Hence, the theorem is proven.

Theorem 5: The number of allowable faulty controllers is $\lfloor (n\text{-}1)/3 \rfloor$ which is maximum

Proof: If the faulty controllers are greater than $\lceil n/2 \rceil$, then all may send different scores to each controller. Fault-free controllers cannot get the common vertices or frontier. Thus, it cannot be sure that all fault-free controllers can reach agreement. If the total number of faulty controller is equal to $\lceil n/2 \rceil$, and n is an even number, then the number of 0 and 1 in the second level may be the same after applying the VOTE function. Under such conditions, all fault-free controllers cannot get a common score. Thus, the total number of allowable faulty controllers is $\lfloor (n\text{-}1)/3 \rfloor$.

Theorem 7: The time complexity of generating a correct score is $O(n^2)$, where n presents the total number of controllers

Proof: In the first round of the exchanging message phase, the source controller will send its initial score to others. Hence, one score must be generated. In the second rounds of the exchanging message phase, all controllers must send the received score in the first round of score exchange to others, and n scores must be generated. In the third rounds, $n*n$ scores must be generated. Therefore, the total quantity of scores to be generated is $(1 + n + n*n)$. The time complexity for generating the correct score is $O(n^2)$.

Theorem 8: To generate a correct score, the space complexity in a 3-level height s-tree structure in $O(n^2)$, where n presents the total number of controllers

Proof: All exchanged scores are presented as a 3-level height s-tree since there are at most three rounds of exchanging message phase by Theorem 4. In the first round of score exchange to others and n scores must be generated. In the third rounds, there are $(n\text{-}1)^2$ scores stored in the third level of a s-tree. Therefore, the space complexity for storing all exchanged scores is $O(n^2)$, where it is generated by $(n\text{-}1)^2+n$.

V. CONCLUSION

In order to improve the correctness and efficiency of the multi-domain SDN environments, it is very important to make sure that all fault-free controllers can work collaboratively. Unfortunately, there may exist some faulty controllers, or the connection between controllers may be broken or disturbed in real situation. How to make let all fault-free controllers to negotiate correctly and efficiently become the goal to be solved.

In this paper, a robust controllers' negotiation mechanism has been proposed. Based on the algorithm, all fault-free controllers can negotiate with only three rounds of score exchange. Furthermore, the time complexity and the space complexity can be reduced to $O(n^2)$ and $O(n^2)$ respectively. Hence, the overall negotiate results will be more robust.

REFERENCES

[1] Openflow Switch Specification (verson 1.4) [cited 2014/03/19]; Available from: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf.

[2] Open Networking Foundation (ONF) [cited 2013/09/30]; Available from: https://www.opennetworking.org/.

[3] T.A. Nick McKeown, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker & Jonathan Turner (2008)," OpenFlow: Enabling Innovation in Campus Networks", *ACM SIGCOMM Computer Communication Review*, Vol. 38, No. 2, Pp. 69–74.

[4] S. Central (2013), "SDN Market Sizing Report 2013/04", [cited 2014/01/05]; Available from: http://www.sdncentral.com/sdn-market-sizing-2013-april.

[5] R.S. Brandon Heller & Nick McKeown (2012), "The Controller Placement Problem", *The First Workshop on Hot Topics in Software Defined Networks (HotSDN '12)*, Pp. 7–12.

[6] J.R. Minlan Yu, Michael J. Freedman & Jia Wang (2010), "Scalable Flow-based Networking with DIFANE", *ACM SIGCOMM Computer Communication Review*, Vol. 40, No. 4, Pp. 351–362.

[7] M.C. Teemu Koponen, Natasha Gude, Jeremy Stribling, Leon Poutievskiy, Min Zhuy, Rajiv Ramanathany, Yuichiro Iwataz, Hiroaki Inouez, Takayuki Hamaz, Scott Shenker & Onix (2010), "A Distributed Control Platform for Large-scale Production Networks", *9th USENIX Conference on Operating Systems Design and Implementation (OSDI'10)*.

[8] S.H. Yeganeh & Kandoo (2012), "A Framework for Efficient and Scalable", *First Workshop on Hot Topics in Software Defined Networks (HotSDN '12)*, Pp. 19–24.

[9] A.K. Sushant Jain, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jonathan Zolla, Urs Hölzle, Stephen Stuart & Amin Vahdat (2013), "B4: Experience with a Globally-Deployed Software Defined WAN", *ACM SIGCOMM 2013 Conference on SIGCOMM*, Hong Kong, China, Pp. 3–14.

[10] Tushar D. Chandra, Robert Griesemer & Joshua Redstone (2007), "Paxos Made Live - An Engineering Perspective", *Twenty-sixth Annual ACM Symposium on Principles of Distributed Computing (PODC '07)*, New York, USA, Pp. 398–407.

[11] K. Phemius, M. Bouet & J. Leguay (2014), "DISCO: Distributed Multi-domain SDN Controllers", *IEEE Network Operations and Management Symposium (NOMS)*, Pp. 1–4.

[12] Amin Tootoonchian & Yashar Ganjali (2010), "HyperFlow: A Distributed Control Plane for OpenFlow", *2010 Internet Network Management Conference on Research on Enterprise Networking (INM/WREN'10)*, USENIX Association, Berkeley, CA, USA.

[13] Advait Dixit, Fang Hao, Sarit Mukherjee, T.V. Lakshman & Ramana Kompella (2013), "Towards an Elastic Distributed SDN Controller", *Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN '13)*, New York, USA, Pp. 7–12.

[14] Wun-Yuan Huang, J.-W. Hu, Shu-Cheng Lin, Te-Lung Liu, Pang-Wei Tsai, Chu-Sing Yang, Fei I. Yeh, Jim Hao Chen & J.J. Mambretti (2012), "Design and Implementation of an Automatic Network Topology Discovery System for the Future Internet Across Different Domains", *26th International Conference on Advanced Information Networking and Applications Workshops (WAINA '12)*, Washington, DC, USA, Pp. 903–908.

[15] J. Kempf, E. Bellagamba, A. Kern, D. Jocha, A. Takacs & P. Skoldstrom (2012), "Scalable Fault Management for OpenFlow", *IEEE International Conference on Communications (ICC)*, Pp. 6606–6610.

[16] Wei Xiao, Ruixing Wang & Xiaohong Huang (2012), "Design and Implementation of Ethernet Topology Discovery Algorithm", *IEEE 2nd International Conference on Cloud Computing and Intelligent Systems (CCIS)*, Pp. 767 –770.

[17] Sapegin, Feng Cheng & C. Meinel (2013), "Catch the Spike: On the Locality of Individual BGP Update Bursts", *IEEE Ninth International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, Pp. 78–83.

[18] Jian Mai & Jiang Du (2013), "BGP Performance Analysis for Large Scale VPN", *International Conference on Information Science and Technology (ICIST)*, Pp. 722–725.

Yen-Chiu Chen received her PhD degree in Computer Sciences from National Tsing-Hua University, Taiwan in 2010 and BS degree in information management from Chung Hua University, Taiwan in 2004. Yen-Chiu works as a R&D engineer at Information and Communications Research Laboratories (ICL) of Industrial Technology Research Institute (ITRI) in Taiwan. Her research interests are in software-defined networking, cloud computing, scheduling theory, real-time computer systems. Up to 2014 year, she has published 7 journal papers, 13 conference papers and more than 20 conferences/seminars attended.

Hui-Ching Hsieh received her PhD degree in Computer Sciences from National Tsing-Hua University, Taiwan in 2010 and MS degrees in Information Management from Chaoyang University of Technology, Taiwan in 2004. Hui-Ching is an assistant professor at Department of Information Communication in Hsing Wu University. Her research interests include e-Commerce, cloud computing, streaming, distributed data processing, fault tolerant computing, and P2P network computing. Up to 2014 year, she has published 5 journal papers, 6 conference papers and more than 25 conferences/seminars attended.