2013 IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing

SuperFlow: A Reliable, Controllable and Scalable Architecture for Large-Scale Enterprise Networks

Guangwu Hu, Ke Xu, Jianping Wu
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
hgw09@mails.tsinghua.edu.cn, xuke@tsinghua.edu.cn, jianping@cernet.edu.cn

***Abstract*—Enterprise networks have become increasingly important with the rising number of users accessing their services, which consequently brings challenges in various aspects. Though OpenFlow, the promising scheme for enterprise network control, can provide fine-grained and flow-level control in networks, yet it still has a few undesirable designs in security, scalability and performance. Inspired by many excellent OpenFlow-included studies, we in this paper, propose SuperFlow, a reliable, controllable and scalable architecture for large-scale enterprise networks. It not only inherits the merits of OpenFlow, but also overcomes OpenFlow's limitations by introducing many novel features, including the trustiness between hosts and users based on the compulsory authentication mechanisms, the flexible flow controling upon the well-organized control-rule designing, the scalable architecture considerations in both controller load-balancing and management scales. The prototype experiments also prove that SuperFlow possesses these features with desirable performance.**

Keywords-Enterprise network; SuperFlow; OpenFlow; SAVI; SDN

I. INTRODUCTION

Enterprise networks usually refer to large company or campus networks which have at least thousands of users and more than tens of thousands of requests per second (RPS) on average. In a large-scale enterprise network, the number of users could reach more than 25,000, and hundreds of thousands of RPSs need to be processed every second[1]. Since the scale and RPS are very large in enterprise networks, it incurs many challenges regarding the network security, flow controlling and system scalability. Nevertheless, as a promising scheme for large enterprise network control, OpenFlow[2] has been acknowledged by academic community and industry since it can take fine-grained and flow-level control in complex network scenarios and possesses the "central control and edged response" control pattern. It was also considered as the model of SDN (Software Defined Network) architecture,

As Fig.1 illustrates, the OpenFlow architecture is composed of three parts: the center controller, the edged layer3 OpenFlow switch and the communication protocol. For each outbound dataflow, the OpenFlow switch first matches its control rules to process. If it fails, then the switch will take the on-demand way to request control rules from the solo controller through the SSL-based secure channel. Thus, The whole network can get flow-level control by the center controller. Though the merit of this mode is that it not only simplifies network management, but also facilitates network programming, demerits still exist in the following aspects, which yet need to be improved. For example, 1) the single controller can easily become the system bottleneck; 2) the flow-based rule request/response mechanism may cause heavy pressure not only on the OpenFlow switches' data-plane and control-plane, but also on the communication bandwidth between these switches and the controller; 3) forwarding delay for the packets without matched rules is too large; 4) this defect could be exploited by misbehaved users who can randomly forge a large number of spoofing packets to paralyze the network. In summary, these defects might potentially jeopardize network performance and even cause the whole system to collapse.

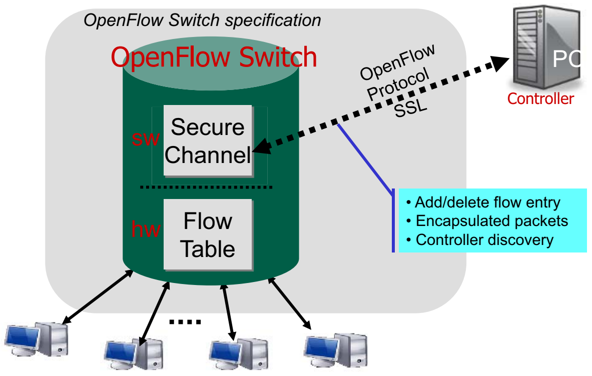

Figure 1. The architecture of OpenFlow

In this paper, we present SuperFlow, a reliable, controllable, and scalable architecture for large-scale enterprise networks. As a result of improvement, SuperFlow overcomes limitations above in OpenFlow by introducing some novel features. First it keeps the credibility for both hosts and users based on the SAVI[3] proposal and user compulsory certification mechanism. Furthermore, it provides dynamic and scalable network control through a flexible control-rule designing and a network change adaption mechanism. Meanwhile, it also achieves controller's scalability via a load-balancing approach. At last, it considers the demand of global control to subsidiary subnets since this situation is very common in many networks. These features enable SuperFlow to satisfy current popular requirements in enterprise networks, and our contribution includes:

978-0-7695-5088-6/13 $31.00 © 2013 IEEE
DOI 10.1109/HPCC.and.EUC.2013.169

- Based on the SAVI and other techniques, we design a series of mechanisms to keep the trustiness for both hosts and users. Meanwhile, we keep the security for vital control information through secure and unified communication interface.
- We achieve the dynamic and scalable control through the dedicated designing of flexible control rule, well-organized control rule description language and control rule conflict-proof mechanism
- We also achieve the goal of system scalability from both sides of controller's scalability and controlling-scale's scalability. We remove the bottleneck defection in the OpenFlow's single controller by transforming it into a DNS scalability issue, which already has many mature solutions. Moreover, considering the demand of large enterprises need a uniformly control to their sub-institution's networks, we establish a hierarchy for globally controlling of multiple allied sub-networks.
- Last, after carefully analyzing the causes of performance deterioration in OpenFlow, we make several important improvements to enhance the system performance.

The rest of this paper is organized as follows: section II summarizes the related work and section III describes the design requirements and goals. In section IV, we presents the SuperFlow's architecture in detail. The evaluation in section V demonstrates the efficacy and stability in our scheme. We draw the conclusion in the last section.

II. Related work

We derived our work from many excellent related work that focuses on the topic of enterprise network control as well as IP source address validation.

A. *Enterprise Network Control*

Essentially, the data-flow control mechanism of OpenFlow is quite simple. At the initial stage, the flow table in the OpenFlow switch is empty, and it takes the on-demand way to install control rules from the solo controller in the domain. When a packet arrives and mismatches any rules, the switch will encapsulate the whole packet and send it to the controller. The controller analyzes the packet, distributes the packet-related rules and redirects the packet to the switch. Thereafter, the switch installs rules and forwards the packet, and subsequent packets belonging to the same flow will be forwarded directly. Obviously, there is an efficacy problem on the on-demand request and response mechanism. Rob Sherwood et.al. set up a test-bed and evaluate the average forwarding delay in the first unknown packet, and the result shows that the new flow setup latency in OpenFlow reaches to 16.16ms[4]. Meanwhile, DevoFlow[5] characterizes the OpenFlow's overhead in traffic control, which indicates that there is about one control packet for each two or three data packets, and there are only 275 flows per second that can be set up with a OpenFlow-supported modern HP switch. However, this flow processing ability far from enough to satisfy the demand from large-scale enterprise and datacenter networks. For proving this, Tavakoli et.al. [6] apply OpenFlow into datacenter network and estimate that a large datacenter consisting of 2 million virtual machines may generate 20 million flows per second, which suggests that OpenFlow is not qualified to the latency-sensitive datacenter network, and DevoFlow also shares the same view.

In order to reduce the delay in the first unknown packet control, DIFANE[7] installs many "Authority Switches" to act as controller proxy, so that it can always keep packets in switch's data-plane. The shortcomings of this mechanism at least include: 1) the architecture makes the system more complex; 2) each ingress switch only has to resort to its designated Authority Switch, which causes the system hardly to adapt to the topology change because network dynamic will affect their communication reachability; 3) it still doesn't address the issue of global visibility of flow states and statistics.

For the purpose of extending the solo centralized control server and neutralizing the performance bottleneck, HyperFlow[8] tries to separate the control-plane from data-plane to achieve the controller scalability, while Maestro[9] and McNettle[10] exploit the parallelism way of multi-thread programming or multi-core CPU hardware to realize it. Nevertheless, these plans are not practical enough to satisfy enterprise networks' demand. Certainly, other novel ideas such as NOX[11], 4D[12], ForCES[13], I2RS[14], Tesseract[15], Resonance[16], XORP[17] and Onix[18] influenced our thinking.

B. *IP Source Address Validation*

In the aspect of the IP source address validation, the famous architecture of SAVA[19](Source Address Validation Architecture) can provide transparent network service to ensure that every packet can hold an authenticated source IP address. It consists of three levels, the Inter-AS, Intra-AS and first hop. In each level of this hierarchical architecture, it can get different granularities of the authenticity, and one or more mechanisms [3,20, 21] are defined to solve this problem. In the user access subnet, SAVI proposal was approved by IETF to resolve this issue. Following the SAVI specification, a normal layer2.5 switch is named the SAVI Switch, which can filter spoofing packets by establishing the triangle relationship of IP address, MAC address and uplink-port for each host. As to the binding relationship establishment and packets anti-spoofing fullfillment, it is accomplished by the IP address assignment protocols sniffing and the CPS protocol (Control Packet Snooping). Compared with the matured solution of uRPF[22] (unicast Reverse Path Forwarding), SAVI is more accurate because its effected point is the users' access switch rather than the access router. Besides, although it is very common and practical, the access control lists (ACL) has been proved to be a double-edged sword since it is error-prone and expertise required. At last, solutions like ingress filtering[23], protocol or stack redesign[24], and IP source address encryption[25] all give us many inspirations.

In short, though these studies have partially satisfied our requirements which will be introduced in the next section, to the best of our knowledge, a solution considering all these issues has not been proposed yet.

III. The requirements and goals of SuperFlow

We are running CERNET (China Education and Research Network) (native-IPv4)[26] and CERNET2(native-IPv6)[27], two of the largest academic networks around the world which connect thousands of academic institutes and universities of China. We also cooperate with the largest Chinese ISP provider—China Telecomm. From daily network running practice, we summarize the requirements of enterprise networks as follows:

A. Requirements and Goals

Reliability: In the architecture of OpenFlow, its security only appears in the confidentiality of vital control information. However, we argue that the reliability in both network user and host is as important as the critical control information, since they are the network control foundation. Otherwise, control rules cannot work on the right users and hosts.

Controllability: On the basis of hosts and users reliability, system should ensure that each outbound flow can get its corresponding control. Because of the user mobility and data-flow diversity, network control should be dynamic rather than static so that it can adapt to the changes in the network. Besides, control rule description should be readable and well-organized so that switches can easily interpret them into entries of rule table.

Scalability: In order to process diversified application flow, the rule should be scalable with multi-action, rather than the limited fixed actions. Most importantly, the scale of network control should be extensible to meet the requirements of global control for multiple subsidiary subnets. Under this situation, the cooperation mechanism without rule confliction to the branching controller and root controller should be considered.

Performance: The way of on-demand rule request and flow-based control incur extensive delay in the first unknown packet, therefore, the system should improve this to enhance its performance.

In a word, our goal is to propose a reliable, controllable and scalable architecture for enterprise networks to meet their requirements of reliability IP address and user identification, dynamic outbound flow-control and scalable control hierarchy with high-performance.

B. Analytical Formulation

Assuming in an enterprise network named D, we use the sets $UID_D = \{uid^i | uid^i \in D\}$, $IP_D = \{ip^i | ip^i \in D\}$, $MAC_D = \{mac^i | mac^i \in D\}$, $Port_D = \{port^i | ip^i \in D\}$, $Policy_D = \{policy^i | policy^i \in D\}$ to represent the collection of user identity, IP address, hosts' MAC address, uplink port of layer2 switch for hosts accessing and flow control policy, respectively.

If the packet format can be noted as the set of *Packet* $=\{Version, Length, IP^{src}, IP^{dst} \ldots\}$, then the packets from the host mac^j can be denoted as $Packet_{mac^j} = \cup_{i=1}^{k} \{packet^i_{mac^j} | mac^j \in D\}$. Hence, packets sent out by all hosts in the network D can be notated as $\text{Packet}_D = \cup_{i=1}^{|MAC_D|} \{ \text{Packet}_{mac^i} | mac^i \in D\}$.

Host reliability means that every host's MAC address in its broadcast domain should be unique, that is, every element in the united collection $MAC\_Port_D = \{(mac^i, port^j) | mac^i \in MAC_D, port^j \in Port_D\}$ is unique, or we say the two sets of MAC_D and $Port_D$ have the relationship of 1:1 reflection. Similarly, the united sets of $IP\_MAC_D = \{(ip^i, mac^j) | ip^i \in IP_D, mac^j \in MAC_D \}$, $UID\_IP_D = \{(uid^i, ip^j) | uid^i \in UID_D, ip^j \in IP_D \}$ express the relationship between the IP address and the host collections, and also the relationship between the user identification and the IP address sets, respectively. Then, the **IP address reliability** stands for packets sent out from any host should bring with the IP source address belonging to the owner host, which is formulated as $\forall \text{packet}^i \in \text{Packet}_D$, $\exists (ip^{src} \leftarrow packet^i) \in \{IP_{mac^j}\}$where $\text{mac}^j = f\left(ip^{src}, IP_{MAC_D}\right)$. At last, the **user reliability** refers to each packet should be sent out from the user whose identity is matched with what he/she has claimed, that is, $\forall \text{packet}^i \in \text{Packet}_D, \exists (ip^{src} \leftarrow packet^i) \in \{IP_{uid^j}\}$ where $\text{uid}^j = f(ip^{src}, UID\_IP_D)$.

Proposition 1: To achieve the system reliability related to host, IP source address and user identification reliability, the sets of $Port_D, MAC_D, IP_D, UID_D$ should keep the relationship of 1:1 reflection. In other words, every element in the collection $UID\_IP\_MAC\_Port_D = \{(uid^i, ip^j, mac^k, port^l) | uid^i \in UID_D, ip^j \in IP_D, mac^k \in MAC_D, port^l \in Port_D\}$ should be unique.

Proposition 2: In order to achieve the goal of system controllability, every flow should own corresponding policies to control, namely, each element in the set $Policy\_UID\_IP_D = \left\{(policy^i, uid^j, ip^k) | policy^i \in Policy_D, uid^j \in UID_D, ip^k \in {IP_D}^k\right\}$ should be unique.

IV. THE ARCHITECTURE OF SUPERFLOW

In this section, we depict SuperFlow's architecture in detail.

A. System Architecture

Based on the analytical formulation in the previous section, in order to get the system's reliability, we need multiple network identities to map with each other. Since the SAVI switch can achieve the IP source address reliability, naturally, we deploy them into the access layer. Further, we need network users to authenticate them so as that the system can combine IP binding entries from the SAVI switches with users' identities. Therefore, system can form the multiple identities mapping relationship from the aspect of global view, which also satisfies the requirements of user reliability and system reliability we mentioned. On the basis of system reliability, we need a layer3 access switch to control users' data-flows to realize our goal of system controllability. We named this kind of switch as SuperFlow switch, which has to accept the instructions from the central controller(s) to know how to deal with outbound flows. Eventually, we build the whole architecture named SuperFlow, which is illustrated in Fig.2. In the server section of it, we have more than one controller to respond to numerous flow-control requests from the SuperFlow switches. The scalability in these controllers have been considered through a load-balancing design. At the same time, controllers also possess a unified HTTPs and SOAP-based interface (Simple Object Access Protocol) to provide prompt and secure response to requests. In addition, databases are established to store the system's vital data, like policies, rules, user account, etc. Since no change exists in the rest of parts, such as routers, hosts and firewall, it is thus evident that SuperFlow can facilitate retained network assets to the greatest degree.

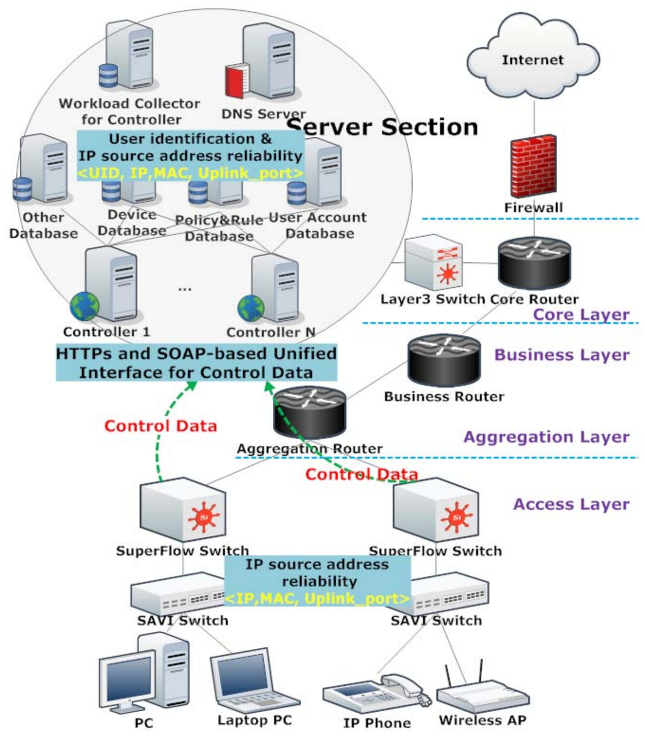

Figure 2. The architecture of SuperFlow

B. System Reliability

We believe that keeping the reliability of hosts and users is the foundation of the system, otherwise, policies cannot have effect on proper users and flows. SAVI has the ability of anti-IP-spoofing in users' access subnets. The SAVI management database (SMD) collects all of hosts binding data from SAVI switches via SNMP. But since SAVI only focus on the IPv6 environment so far, we still progressively to perfect it so as to make it cover both IPv4 and dual-stack, as well as complex IPv4/IPv6 transition scenarios [28]. Meanwhile, keeping user reliability is also very important since a policy is usually associated with user-accounts instead of IP addresses. Thus, the system needs users to authenticate themself before accessing network, which relies on the client software to send out their authentication information to the controller through the security channel. Once the controller has verified these information, SMD will form the quintuple model of "IP, MAC address, uplink-port, user-id and access-time" by integrating these two kinds of authentication data, which is up to the standard of system reliability. In case of users exiting the network accidently, there is also an alive-keeping mechanism to keep the binding information correct which is implemented by the periodically hello message exchanging between end-hosts and controllers.

C. System Controllability

- **Control rule design and description**

In terms of rule granularity, we believe that the rules should combine both coarse and fine granularities, because part of them should be installed in advance into SuperFlow switches for some fixed control, while the rest should be installed based on requests for unknown flows control. Therefore, we divide rules into two categories: **residential** and **temporary**. The former one is basic and coarse-grained, and it will not change within a short period of time. In contrast, the latter is temporary and fine-grained, and it could be changed at any time.

Further, We give an XML-based rule description language. Let's check the scripts in figure 3. The first two rules in the residential rule set (row No.3 and No.4) indicate that the current switch should permit HTTP and DNS protocol packets, but drop any other packets since the third rule (row No.5) has higher priority. However, there is a special host who can access all resources because the last rule in the temporary rule set (row No.8) defines its properties and owns the highest priority. Thus, our rule description language supports multiple properties:

```
1<Rd> <Rules>
2  <Residential Rules >
3  <Rule Timeout="0" Priority="100" Prot="TCP" DPort="80" Action=
       "forward"/>
4  <Rule Timeout="0" Priority="100" Prot="UDP" DPort="53" Action=
       "forward"/>
5    <Rule Timeout="0" Priority="1" Action="drop"/>
6  </Residential Rules>
7  <Temporary Rules>
8    <Rule Timeout="10" Priority="999" SIP="192.168.1.10" DIP="*"
     Mask="255.255.255.255"Action="forward" CreateTime="20120720
     170101"/>
9  </Temporary Rules>
10 </Rules></Rd>
```

Figure 3. An example of XML-based rule description language

Different Timeout: Zero value for the timeout property means this rule will not disappear after the switch starts up. Any other positive values indicate the lifetime of these rules.

Different Priorities: When a flow hits multiple rules, the one with the highest priority always takes in charge. These values are designated by network supervisor with system defined reasonable range. Different rules with different priorities can satisfy different demands from diverse application flows.

Various Actions: Rule script supports not only various individual actions, but also their valid combinations, for instance, drop, forward, broadcast, sample, mirror, source routing, encapsulate, cache, payload encrypt, and forwarding & sampling

Wildcard-Supported: Rules can support the wildcard to get a trade-off between accuracy and performance, e.g., the field of Destination IP (DIP) in the last rule of fig.2 (row No.8) is filled by asterisk, which means this host can access any resources.

Confliction-proof: Since different rules have different timeout values and priorities, SuperFlow can prevent rule conflictions from happening. Even if two rules with the same priority, the confliction resolution mechanism will resolve this (see table I).

- **Control-rule installation and match**

SuperFlow switches takes the way of proactive and reactive to install control rules, that is, residential rules should be installed at the moment of switch boot-up to achieve a general control ability, while others for detailed flows' requests should be installed reactively. As to how SuperFlow switches deal with the first unknown packets, it slightly differs from OpenFlow. If a packet misses any rule, the switch will cache it first, then deliver its summary (flow's quintuple form) instead of the whole packet to the controller. Finally, the switch will install the rules and process the packet and its subsequent.

- **Cope with network dynamic**

SuperFlow can cope with network dynamics by means of monitoring both SNMP and OSPF protocols so as to learn the newest topology, and then to respond to network changes from the view of global network at fast speed. Once network change occurs, controllers will evaluate the affection and redistribute new rules to the corresponding switches. This mechanism is more efficiency than the OpenFlow's method that the switches periodically send message to each other.

- **Rule partition strategy and its Consistency**

Generally, network administrators will not give SuperFlow switches any specific rules directly. Instead, they will give controllers policies like "only subnet A can visit the FTP server", and "all the correspondence with destination IP TCP-1433 will be prohibited". On the other hand, switches can only recognize the XML-based rule description language. Thus, controllers should convert the abstract policies into specific rules. Undoubtedly, controllers still need strategies to partition rules into many unique versions for each switch, as showed in Fig. 4. As the questions like how policies can be turned into rules, what kind of partition strategy should be held and how to keep the consistency when rules conflict happen, Table I answers them.

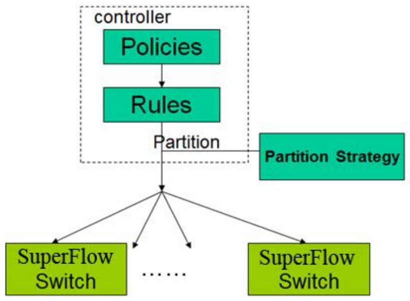

Figure 4. The illustration of rule partition in controller

TABLE I. THE PARTITION STRATEGY AND CONFLICT RESOLUTION

Phase	Method
Policy/rule Requirment	The quintuple form of flow's information (wildcard is available) and policy's classification (residential or temporary), priority, start-time, live time.
Partition Strategy	**by switch:** different rule versions to different switches; **by granularity:** split into Temp. and Residential rules; **by priority:** ordering according to rule priority

Conflict Resolution	Residential rule > Temporary rule;
	high priority > low priority;
	Policy from superior root controller > local policy

D. *System Scalability*

- **Controller's scalability**

In our network management practice, for a campus network with 20K users, the average active number of IP/user is up to 8K, while the average number of new flow request is up to 6K and the average total established flow is around 500K. However, according to the experiment we conducted based on our scheme, a commodity PC server can process less 3K requests per second. Hence, the controller should be scalable instead of solo. DevoFlow has also proved that a single controller is very likely to become the system's bottleneck. We achieve the controller's scalability by designating controller's domain name instead of their IP addresses to the switches. Then, we convert the issue of controller load-balancing into one of DNS load-balancing. Fig. 5 illustrates this consideration.

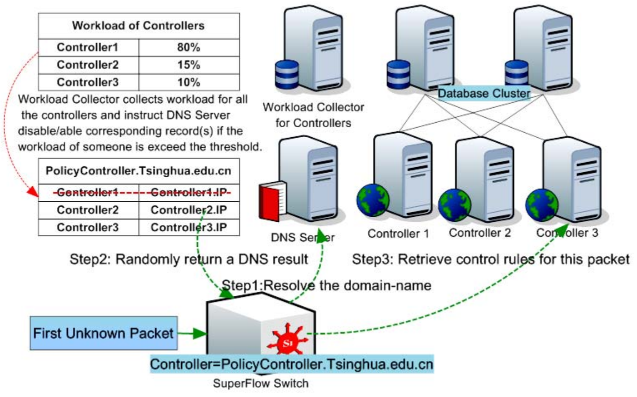
Workload of Controllers
Controller1 80%
Controller2 15%
Controller3 10%
Workload Collector collects workload for all the controllers and instruct DNS Server disable/able corresponding record(s) if the workload of someone is exceed the threshold.
Workload Collector for Controllers
Database Cluster
PolicyController.Tsinghua.edu.cn
~~Controller1 Controller1.IP~~
Controller2 Controller2.IP
Controller3 Controller3.IP
DNS Server
Controller 1
Controller 2
Controller 3
Step2: Randomly return a DNS result
Step3: Retrieve control rules for this packet
Step1:Resolve the domain-name
First Unknown Packet
Controller=PolicyController.Tsinghua.edu.cn
SuperFlow Switch

Figure 5. The DNS load-balancing for controller scalability

Meanwhile, in order to avoid frequent DNS requests which might result in performance deterioration, the switch will not repeat this DNS resolving within a period of time. Moreover, we have a server for workload collector to detect all the controllers' workload within a fixed interval, which can enable/disable the corresponding DNS records according to their pressure-level. Since the interval of detecting controllers' workload is more frequently than the switch's DNS request, we do not need to synchronize them to achieve networking consistency, otherwise simultaneous DNS request will deteriorates system's performance.

- **Controlling-scale's scalability**

Given the fact that a company might own several subsidiary networks, we design a mechanism to cope with this situation. As illustrated in Fig. 6, for the purpose of universal control of these multiple subnets, all the independent branching network should be under the central control of the headquarters' root controller. In this case, a global policy database in the headquarter must be built. When the root controller has a policy change or issue, it will distribute it to branching controllers through a HTTPs-based secure channel. Then, the branching controllers should place these policies into their local policy database with higher priority, so as to avoid the policy confliction. The communication protocol between them takes the approaches of SOAP-based pull and TCP-based trap, which is very similar to the SNMP mode.

We also can expand this mode into multi-level control situation. If the notation (P_D, F_D) refers that all the flows in the network D has corresponding policies to handle, then N levels have $N \cdot \bigcup_{D=1}^{k}\{(P_D, F_D)\}$ policies to manage, and the controller number can be denoted as $\sum_{i=1}^{n}\sum_{j=1}^{k}|C_j^i|$,where $|C_j^i|$ means the controller number in the subnet of the i^{st} level and j^{st} column.

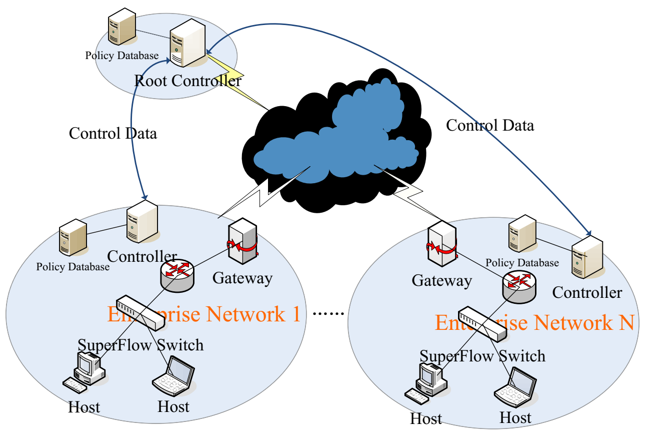

Figure 6. The illustration of two levels unified control for allied-networks

E. System Security

The principle of our designing in system security is that the transmission of the vital information through SSL secure channel, while traffic measurement packets are still delivered by UDP and other communications for coordination are still held by TCP. Since the unified SOAP-based web interface has a very close relationship with XML language, it can adapt with our switch configuration scripts and rule description language well, as what table II shows.

TABLE II. THE SUMMARY OF COMMUNICATION MECHANISM

Phase	Protocol
Other coordination communication	TCP
Swtich plug-and-play User authentication Rule request/distribute Communication between controllers	SSL/HTTPS
Measurement statistic	UDP

F. Summary

In order to achieve more desirable goals, we introduce many new features and improve some drawbacks of OpenFlow, which will inevitably bring some costs. As to the whole cost of packet forwarding, let's check their individual cost evaluation.

$$\mathrm{Cost(OpenFlow)} = \mathrm{Cost(rule\ request)} + \mathrm{Cost(forwarding)}$$

$$\begin{aligned}\mathrm{Cost(SuperFlow)} = {} & \mathrm{Cost(host's\ IP\ binding)} + \mathrm{Cost(user\ auth.)} \\ & + \mathrm{Cost(IP\ spoofing\ filtering)} \\ & + \mathrm{Cost(rule\ request)} + \mathrm{Cost(forwarding)}\end{aligned}$$

Apparently, SuperFlow incurs more delays with more procedures than OpenFlow. However, we argue that the cost of IP binding and IP spoofing filtering procedures in SAVI switch can be ignored since the former just snoops the IP assignment protocol to establish the binding relationship, while the latter can achieve line rate. The user authentication procedure only repeats once. Though it is the same procedure of rule request, SuperFlow has some tradeoffs in rule designing and forwarding. Consequently, as proved by our evaluation the average packet forwarding delay in SuperFlow outperforms OpenFlow. There are more detailed comparisons between them in table III.

TABLE III. THE COMPARISONS BETWEEN OPENFLOW AND SUPERFLOW

Comparisons	OpenFlow	SuperFlow
Rule installation	Reactive	Proactive & Reactive
Rule granularity	Flow-based fine-grained	fine-grained & coarse-grained
Rule action	Limited four types	Multi-types
Rule description	Complex & Unstructured	Simple & Structured
Control mechanism for the first unknown packet	1. The switch encapsulates the whole packet to the controller. 2. The controller distributes the rules with the packet. 3. The switch installs the rules and processes the packet according to the rules.	1. The switch caches the packet first, and then requests the rules from the controller with the packet's quintuple flow summary. 2. The controller distributes the rules. 3. Same with the third point of OpenFlow.
Path control for data-flows	All the devices along the forwarding path will be distributed related rules	Not care, or use the source routing tech. to add the routing path into packet's header.
Network dynamic adaptability	Periodically send (receive) the topology information to(from) neighbor switches	Using SNMP&OSPF combined way to monitor the whole network dynamics with faster speed.
System Scalability	None	Scalable in the rule action, controller's quantity and scale
Control interface	SSL	Integration platform with HTTPs, TCP and UDP protocols
Rule confliction-proof	None	Different group & Different Priority etc.

V. EVALUATION

We have implemented the SuperFlow's prototype and got it evaluated in a campus network.

A. Prototype Implementation

SuperFlow switch & SAVI switch: We use a commodity PC (Core i3-2120 3.3GHz CPU, 4G 1333Hz Memory, 500GB Hard-disk and one integrated 1Gbps network-card, Linux kernel 2.6.32) to act as the OpenFlow switch, which installs another extra network-card (Intel EXPI9402PT) with total three gigabytes Ethernet interfaces. We create necessary databases in MySQL and establish a firewall based on the IPtable (iptables-1.4.11), as well as a Java-based application to communicate with controllers and to instruct the firewall to process flows. The controller's DNS request interval we set as 5 minutes. We take a commodity SAVI switch product(DCRS-5980) to enable host binding function; **DNS Load-balancing**: We set up two DNS servers and apply Windows DNS Service as our DNS load-balancing solution, which only needs to add the DNS records in the Forward Search Area; **Database:** We create all necessary databases in Microsoft Windows SQL Server, like Policy, Rule, Account, Devices, and etc. However, limited by the paper length, we will not list all the fields of these tables; **Workload collector**: it collects workload of CPU and memory in each controllers with a period of 1 minute, and informs the daemon application in DNS server to add or delete the proper DNS records; **Controller:** The communication interface in controllers is implemented by web services, which includes necessary web-methods and facilities clients remotely invoke. The mainly web-methods are illustrated in Table IV.

TABLE IV. THE MAINLY WEB-MEMHODS IN CONTROLLER

Function	Proto.	WebMethod
Switch Verification	HTTPs	//Return switch configuration script String SwitchVeri(String MAC, String serialNo)
Rule request	HTTPs	//Return rule description to switch String RuleReq(String srcIP, String dstIP, int srcPort, int dstPort, int prot, string switchIP)
Sub-contr. request policy from root contr.	HTTPs	//Root controller distributes policies boolean DownloadPolicy(String rootIP,string usrName, String password, String policy)
User Verification	HTTPs	//Return verification result to host Boolean UserVeri(String userName, String Password, String hostIP, String md5)

B. Performance

Our evaluation environment is a college campus network, which owns about 27,000 network users and 20,000 cable slots. The average number of online users and the number of traffic loads in the core routers are about 3,000 and 600Mbps, respectively. The SuperFlow prototype was tested in a dormitory building of this campus network which contains nearly 1K students, and our SuperFlow switch uplinks the buildings access router and downlinks many SAVI switches so as to provide Internet accessing services for these students.

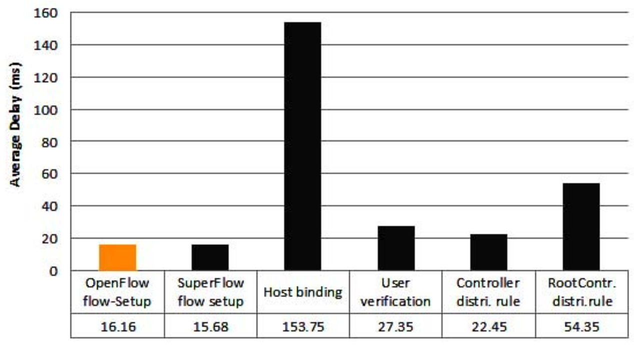

Figure 7. The average delay of the different precedures

First, we test the SuperFlow switch's throughput with professional IAIA traffic generator, we can learn that the maximal forwarding throughput is about 350Mbps. But we believe that taking the non-x86 hardware architectures will achieve much better performance, and in the future work we plan to use the NetFPGA[29] card to redevelop it. Then we carefully evaluate the time cost of each procedure which is showed in Fig. 7, we can learn that we have reduced 3% delay in flow-setup than OpenFlow. As the longest delay in the host binding procedure, it depends on the detailed IP address assignment protocol since SAVI switch just listens the whole interactive courses, and in our experiment we take the DHCP.

Further, we test the average forwarding throughput of SuperFlow switch with different number of traffic and rule combination, the result in Fig.8 tells us that the major constraint comes from the switch's performance, instead of the number of control rules. Meanwhile, in order to investigate the impact from the flow number and packet field matching, we also evaluate the packet forwarding delay with various flows and matching properties in Fig.9, which states that it is the flow number rather than the matching items that is the major factor.

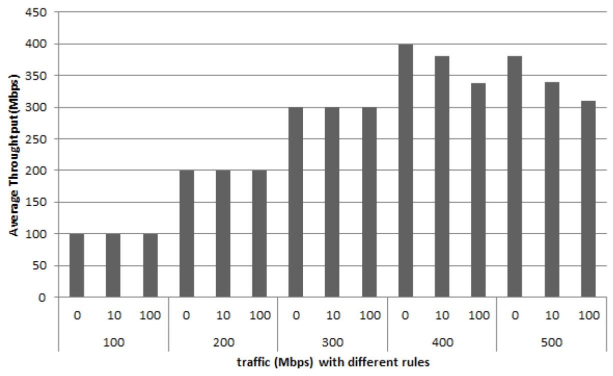

Figure 8. The throughput of Superflow switch with different traffic and rules

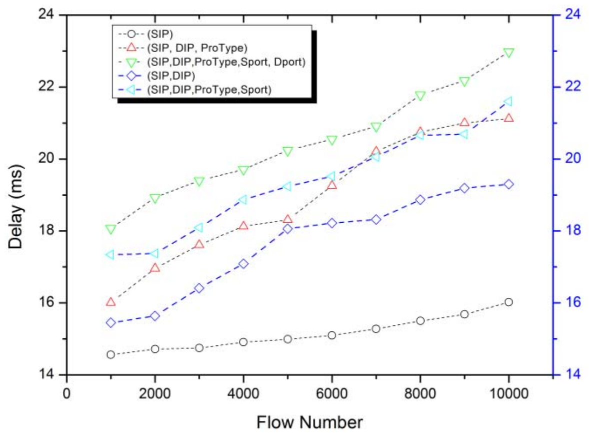

Figure 9. The packet forwarding delay with different number of flows

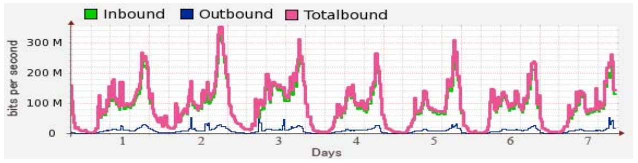

Figure 10. The traffic of Superflow switch in seven days

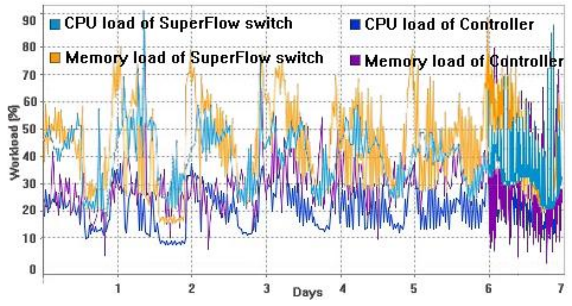

Figure 11. The CPU&Mem. load in SuperFlow and controller for seven days

We put our prototype running for seven consecutive days, the switch's traffic status and the controller's CPU/memory workload are illustrated in Fig. 10, Fig. 11, respectively, which indicate that system works very steady. Fig. 12 depicts that the average flow-setup requests to the controller reduce sharply as time pass by, while the flow-setup number (the switch's performance) climbs up rapidly to a steady status, which we believe it results from the rule accumulation/localization in the switch, as well as the fine- and coarse-grained combined rule property.

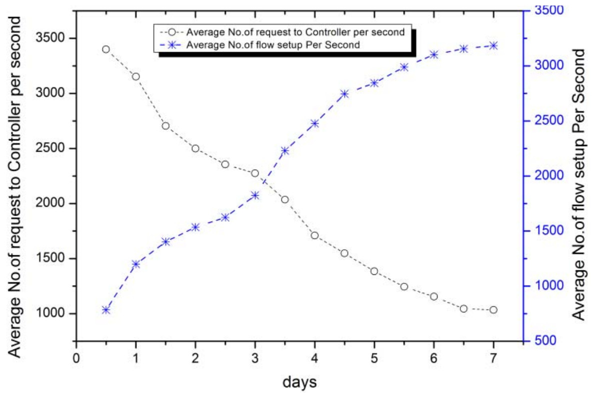

Figure 12. The average number of flow setup and request per second in SuperFlow switch for seven days

VI. CONCLUSION

Because of a large number of data-flows, policies and users, it brings enterprise networks with numerous challenges in various aspects. OpenFlow, serving as a classical example for the architecture of SDN, gives us lots of inspirations to overcome these challenges. However, OpenFlow still exists some flaws in the aspect of scalability, efficiency and security. After a comprehensive investigation of the enterprise networks requirements, as an improvement of OpenFlow, we present a new architecture named SuperFlow that offers distinctive features include reliable identity to network hosts and users, scalable and dynamic control to data-flows, flexible system scalability in controller-numbers, rule-actions and controlling-scales. We have implemented the prototype of SuperFlow and got it evaluated in a real campus network. The result has also proved its stability, controllability, scalability and efficiency, which makes it satisfied the management and control requirements of large-scale enterprise networks.

ACKNOWLEDGMENT

This work was supported by the NSFC Project (61170292), 973 Project of China (2009CB320501, 2012CB315803), 863 Project of China (2013AA013302), new generation broad-band wireless mobile communication network of the National Science and Technology Major Projects (2012ZX03005001) and the National Science and Technology support Program (2011BAK08B05-02).

REFERENCES

[1] Websense. Deployment Guide Supplement for Very Large Enterprise Network.http://www.websense.com/content/support/library/web/v70x/en/v70x_Deploy_VL_Enterprise.pdf

[2] OpenFlow. http://www.openflow.org

[3] J.Wu, J.Bi. Source Address Validation Improvement Framework. http://datatracker.ietf.org/doc/draft-ietf-saviframework, October 2010

[4] R. Sherwood, G. Gibb et.al. McKeown, and G. Parulkar. Can the production network be the testbed? In Proceedings of OSDI 2010:1-14

[5] Andrew R. Curtis, Jeffrey C. Mogul, Jean T. and et al. DevoFlow: Scaling Flow Management for High-Performance Networks. In Proceedings of SIGCOMM 2011: 254– 265

[6] Tavakoli, Arsalan, et al. "Applying NOX to the Datacenter." HotNets. 2009.

[7] Minlan Yu,Jennifer Rexford et al, "Scalable Flow-Based Networking with DIFANE," In Proceedings of SIGCOMM 2010: 351–362

[8] Amin Tootoonchian,Yashar Ganjali. HyperFlow: A Distributed Control Plane for OpenFlow. In Proceedings of INM 2010

[9] Cai, Zheng, Alan L. Cox, and TS Eugene Ng Maestro. A system for scalable OpenFlow control. Technical Report TR10-08, Rice University, 2010.

[10] Voellmy, Andreas, and Junchang Wang. "Scalable software defined network controllers." Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication. ACM, 2012

[11] Gude, Natasha, et al. "NOX: towards an operating system for networks." ACM SIGCOMM Computer Communication Review 38.3 (2008): 105-110.

[12] HONG Y, MALTZ D A, EUGENE T S. Tesseract: a 4D network control plane. In Proc 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI). Cambridge, MA, USA, 2007, 382

[13] L.Yang, R. Dantu, T. Anderson, R. Gopal. Forwarding and control element separation (ForCES) framework. IETF RFC 3746, September 2004

[14] Interface to the Routing System (i2rs). http://datatracker.ietf.org/wg/i2rs

[15] A. Nayak, A. Reimers, N. Feamster, and R. Clark. Resonance:Dynamic access control for enterprise networks. In Proceedings of Workshop on Research in Enterprise Networks, 2009.

[16] C. Kim, M. Caesar, and J. Rexford. Floodless in SEATTLE: A scalable Ethernet architecture for large enterprises. In Proc. of SIGCOMM 2008

[17] Handley M, Hodson O, Kohler E. XORP: An open platform for network research. ACM SIGCOMM Computer Communication Review, 2003, 33(1): 53-57.

[18] T. Koponen, M. Casado, N. Gude, J. Stribling,et.al. Onix: a distributed control platform for large-scale production networks. In Proceedings of OSDI, 2010.

[19] Jianpin Wu, Gang Ren, Xin Li. Source Address Validation: Architecture and Protocol Design design. In Proc. of IEEE ICNP 2007:276-283

[20] Guangwu Hu, Jianping Wu, Ke Xu et.al. SAVT: A Practical Scheme for Source Address Validation and Traceback in Campus Network. In Proceedings of IEEE ICCCN 2011

[21] Bi Jun,Liu Bingyang,Wu Jianping etl. Preventing IP Source Address Spoofing: A Two-Level, State Machine-Based Method.Tsinghua Science and Technology. 2009，4(14): 413-422

[22] Understanding URPF. http://www.cisco.com/web/about/security/intelligence/unicast-rpf.html.

[23] F.Baker,P.Savola. Ingress Filtering for Multihomed Networks, IETF RFC 3704, 2004

[24] J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang. SAVE: Source Address Validity Enforcement Protocol.In Proceedings of INFOCOM 2002, June 2002.

[25] T. Aura. Cryptographically generated addresses (CGA). RFC 3972, March 2005

[26] CERNET. http://www.cernet.edu.cn

[27] CERNET2. http://www.cernet2.edu.cn

[28] K.Xu, G.Hu, J.Bi, M.Xu et.al. The A General Framework of Source Address Validation and Traceback for IPv4/IPv6 Transition Scenarios. http://tools.ietf.org/html/draft-xu-savi-transition-02, November 2012

[29] NetFPGA. http://netfpga.org/