Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 674057, 7 pages
http://dx.doi.org/10.1155/2013/674057

Research Article

An Analysis and Design of the Redirection Schema in ForCES

Ming Gao,[1,2] Shiju Li,[1] and Weiming Wang[2]

[1] *Department of Information Science and Electronic Engineering, Zhejiang University, No. 38, Zheda Road, Hangzhou 310027, China*
[2] *Department of Information and Electronic Engineering, Zhejiang Gongshang University, No. 18, Xuezheng Street, Hangzhou 310018, China*

Correspondence should be addressed to Ming Gao; gaoming19790508@126.com

Received 4 July 2013; Accepted 29 July 2013

Academic Editor: Yoshinori Hayafuji

The idea of Forwarding and Control Element Separation has widely accepted by next generation network researchers, the regain attention of IETF (The Internet Engineering Task Force) ForCES (Forwarding and Control Element Separation) is the best proof. An IP tunnel-based redirection schema was proposed to solve the problem of routing protocol messages interaction between ForCES router and the external merchant routers. The technology of network virtualization is introduced to map network interface from ForCES FE (Forwarding Element) to CE (Control Element) which collaborating with the redirect schema.

1. Background and Related Work

ForCES (forwarding and control element separation) [1] is a working group of routing area in IETF (the internet engineering task force), which devotes itself to study the architecture, standard, and gordian technique. In this paper, for convenience, we always call IETF ForCES ForCES elliptically.

In ForCES, network devices such as ip router, firewall, and ethernet switch will be called by a joint name of NE (network element) [1]. Every NE is composed of two separated plane: forwarding and control. Forwarding plane consists of several FEs (forwarding element) [2] which are dedicated to fast packet routing lookup and forwarding. CE (control element) on the control plane is designed mainly for calculating path, producing routing table and synchronizing them to every FE.

The purpose of ForCES is to provide an open, standard, and modularized platform for network device's design and manufacture to bring more innovation. Although the concept of forwarding and control element separation has been proposed by IETF ForCES for 10 years, there is still no authoritative research result appearing on the design and analysis of ForCES implementation. Another interesting thing is that the word of ForCES never disappears from our sights in the past 10 years, and now the idea of ForCES becomes the common sense for more and more researchers in next generation network area, especially for open network researches.

In open network area, XORP [3] constructs an open and extensible router platform which provides various routing protocols such as OSPF, RIP, and BGP with open interfaces. These features are very interesting for a ForCES researcher because they can greatly reduce the work of developing routing protocols in ForCES implementation. Similar to the ideas of LFB (logical function block) [4] in ForCES, Click [5] partitions route into a number of abstract function modules and packet's content and transmission direction will be changed when traversing these modules; different combinations of these abstraction modules would generate different types of router function, for example, IPv4 and MPLS. The most notable character of Click is the modularity and high flexibility of architecture; however, Click does not show us how to quickly develop new routing protocols.

In the past two years, a large number of open network researchers turn to the next generation network architecture. The most representative one is SDN [6] which is powered by ONF [7] and IRTF [8], and now it becomes the focus and clearly claims that its core idea comes from IETF ForCES. In ONF, different from traditional network devices, OpenFlow [9] considers flows in a network no longer to be individual packets but a bundle of data flows. The processing of data flow can be defined and directed by controller via configuring

flow tables; every entry of a flow table that is maintained by a controller has a strategy. To replace the pattern of per-packet routing, OpenFlow switch utilizes flow strategies to make decision of packet's outgoing, with just the controller being very similar to ForCES CE.

Although OpenFlow has been very successful and obtained great attention, its route map of clean slate will bring a great shock to the existing IP network. Therefore, the NVO3 (network virtualization over 3-layer) [10, 11] that emerged just last year tries another way of IP tunnel to realize the separation of business logical network and physical network. Every VN (virtual network) presenting as a specified business network has independent topology and address space. The routing between VNs can be done by BGP with the direction of controller because only controller has the overview of VN distribution.

From the previous, we can definitely say that the separation of forwarding and control elements separation is a universal phenomenon now in architecture design and network management. This paper will focus on two problems: (1) How can NE exchange routing protocol information with the outside. (2) How to make CE learn FE's physical network interfaces and operate them locally.

For the problem (1) above, the key is to guarantee the integrality of routing protocol information to flow freely between CE and FE. Wang et al. [12] designed a ForCES-based Ip router and told us how to implement an ip-forwarding-enabled FE but didi not mention the exchanging of routing protocol information between CE and FE. In Wang's implementation, routing table is just added manually into FE and CE did not run any routing protocol. So, we propose an IP tunnel-based redirection mechanism as the carrier to hold all of the routing protocol information.

For the problem (2), the technology of interface virtualization [13] is adopted to number the physical network interfaces on FEs uniformly and then map them to CE. CE will operate these virtual network interfaces just like its own, but in fact they are located in FE.

2. The Design of Redirection Schema

Wang et al. [12] supplies an open and standard implementation of ForCES protocol layer (PL) [14] and transport mapping layer (TML) [15] and packages the implementation as a middle box. The middle box is composed of two parts: *ForCES protocol* and *ForCES function middleware*. The former undertakes the tasks of encapsulation, de-encapsulation, and so forth of ForCES PDU (packet data unit), which is defined by PL. The latter on FE is responsible for registration and attribute management of relevant LFBs. The middleware obtains operating information from *ForCES protocol stack*, calls registered processing functions to manipulate LFB attribute, manages LFB event lists and generates event reports. *ForCES function middleware* on CE is designed mainly for saving all the attribute information of LFBs, analyzing user operating commands from user, and distinguishing redirect messages from messages received by

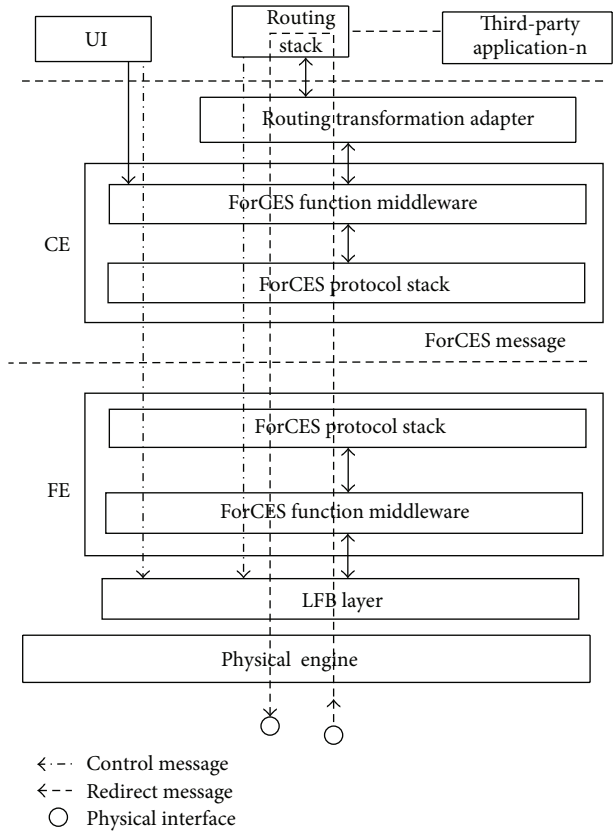

Figure 1: Software structure of ForCES NE.

ForCES protocol stack and forwarding them to a third-party software, for example, Xorp, zebra.

Based on Wang's contribution, we further propose a software framework of ForCES NE shown in Figure 1 and the detailed introduction as follows.

When FEs have received routing protocol messages such as OSPF and SNMP from physical network interfaces and cannot handle them by themselves, then FEs will send these messages to CE to request for further processing. After the succeeded processing, CE sends them back to the corresponding FE. Whether from FE to CE or from CE to FE, all the routing protocol messages must be encapsulated as ForCES PDU. We call the process redirection and the type of ForCES PDU *ForCES redirection message*.

The format of PDU of ForCES *redirection message* is defined and implemented by PL [14]; also PL defines another important message named *ForCES control message* which will be used by CE to control and manage FEs including attribute configuration, query, and report of capacity and event.

For the sake of convenient stating, in the remainder of this chapter the words of message and packet will appear alternatively and have the same meaning.

In Figure 1, commonly *routing stack* supplied by third-software Xorp is located in CE. Now we suppose that routing protocol packets (i.e., RIP, OSPF) are originated from outside routers and now have been received in FE; then methods taken by FE must solve problems as follows: (1) settle down which type of ForCES message, redirection, or control to be

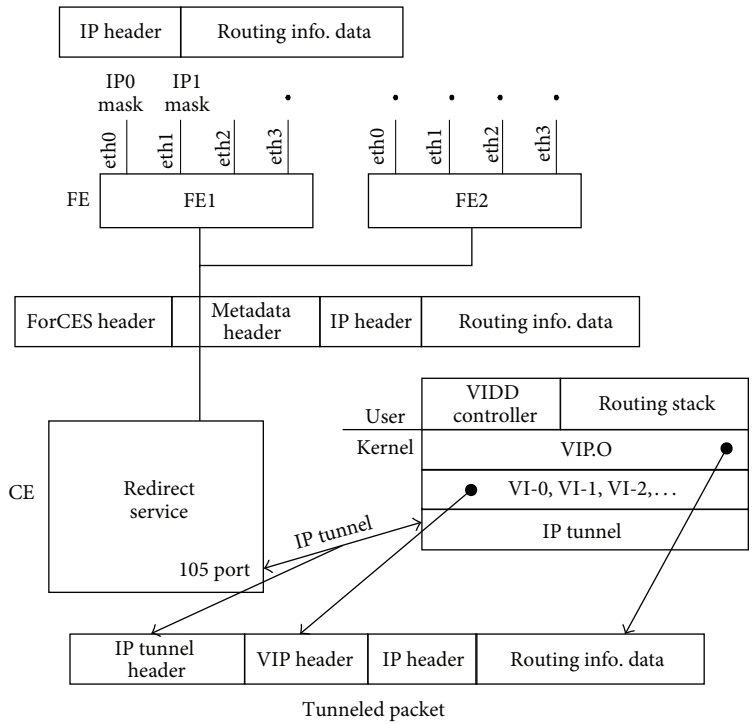

Figure 2: Routing protocol packet transmitting in ForCES NE.

the carrier to transport routing protocol packets from FE to CE and (2) determine what *Metadata* are needed by CE to handle these routing protocol packets properly and then deliver them to CE.

Because of the separation of FE and CE, CE has no way to make sense of where these routing protocol packets come from and go to, so we must try to realize the information synchronization between CE and FE, and the *routing transformation adapter* is introduced which will be carefully detailed in part III. Much information needs to be synchronized; the most important one is the network interface. Physical network interfaces of NE mainly exist in FE, and in order to map these network interfaces from FE to CE, virtual interface (VI) technology [13] is needed. With the help of VI, CE can operate these virtual network interfaces just like its own, but in fact they are located in FE.

According to the definition of ForCES framework [1], an incoming packet in FE will be processed by LFBs from ingress-EtherPHYCop to egress-EtherPHYCop; all of the LFBs must located in datapath defined by FE-Intra topology [16]. Obviously during the processing of LFB, some data maybe produced for next usage. For example, when IPv4UcastLPM LFB is doing the prefix matching, a key data *NexthopID* will be produced which is necessary for subsequent IPv4NextHop LFB is processing and we call these key data *Metadata*. To be noted is that all of the types of LFB mentioned earlier in this paragraph can be found in LFB library [17].

Figure 2 shows an efficient method of how to transfer routing protocol packets between CE and FE. When FE receives the packets which need to be processed by CE, *metadata* will be added to IP packet header and the IP packet is just the routing protocol packet.

To facilitate the encapsulation of *Metadata*, we created a collection of headers named *Metadata header* to contain these *Metadata* in order. *Metadatas header* mainly includes four fields: FEID, Res, portID, and Length. FEID identifies the arrival of packets or destination CE. Res is a reserved field. portID denotes the arrival of packets or destination port number. Length stands for field length. CE analyzes the *ForCES header* which is located in the *redirection* messages and draws out the *Metadata* information and then delivers it to CE. At this time, CE will convert the *Metadata* to virtual interface port (VIP) and attach *VIP header* to the IP packets. After doing this, CE will transmit the packets to VIP module for processing and then add an *IP tunnel header* before *VIP header*. Thus, IP tunnel + VIP + IP + DATA package format is formed. Finally the packets will be accessed and processed by *routing stack* in CE.

After the processing of request packets, *routing stack* generates specific response packets which also need to add specific *VIP header* in virtual network interface card. Then they will be sent to VIP through IP tunnel. When CE receives the packets, *Metadata header* will be added and they will be encapsulated into ForCES redirection message and sent to FE. Subsequently FE determines the outgoing physical network interface and sends the packets out according to the *Metadata header*.

What needs to be particularly pointed out is that *routing stack* described in the paper deals with two types of messages: *control and redirection* as illustrated in Figure 1. When *routing stack* needs to interact with the outer routers, *redirection*

channel will be used. While the routing table has been changed by *routing stack*, *control* channel will be used to synchronize routing table with FE.

3. The Synchronization from CE to FE

Routing table is made up of route entries and there are two kinds of routing tables existing in current routing system: integrated and distributed [17].

For edge router, a single table is usually used to store routing entries for the convenience of management, which is called integrated routing table. Each entry of the table includes *Destination*, *Mask*, *NHIP* (next hop IP), OID (outport ID), *Flag*, *MTU,* and *Metric*.

For core routers, as the number of route entries is too large (generally around 20000), enormous storage space will be wasted while using a single table. The reason is that in an integrated routing table, different destination networks may point to the same next hop. In that case, the amount of routing table can absolutely be compressed by reasonable designing. Thus distributed routing table is proposed. It maintains two subtables: *PT* (prefix table) and *NHT* (next hop table). Every entry of PT consists of fields of *Destination*, *Mask,* and *NHI* (next hop index). *NHT* contains *NHI*, *OID* (outport ID), *Flag*, *MTU,* and *NHIP* (next hop IP). The two tables correlate to each other by NHI. By using this type of design of routing table, the repetitive storage of information in a table may be greatly reduced.

Here a calculating model is set up to calculate the compression efficiency when an integrated routing table is converting into a distributed one. For integrated routing table we define every route entry as $RTEntry = \{Destination, Mask, NHIP, OID, Flag, Metric\}$. Let $Length_0$ denote the total bytes of all of the fields. Suppose that the number of route entry is N and store each field's value of *RTEntry* into arrays $Destination[N]$, $Mask[N]$, $NHIP[N]$, $OID[N]$, and $Metric[N]$. Suppose, K is the times when different values appear in $NHIP[N]$; then we can get a set $NHIP_{\text{val}} = \{NHIP_1, NHIP_2, \ldots, NHIP_K\}$ and any NHIP_i in NNIP_{val} is defined as follows:

$$NHIP_i = \{NHIP[i+j], NHIP[i+k], \ldots, NHIP[i+l]\} \tag{1}$$

subject to

$$1 \le i+j \le i+k \le i+j \le N,$$
$$NHIP[i+j] = NHIP[i+k] = \cdots = NHIP[i+l]. \tag{2}$$

Let $NHIPCount[i]$ be the number of items in set $NHIP_i$; we will get $\sum_{i=1}^{K} NHIPCount[i] = N$ and obviously $Size_0 = Length_0 * N$ can be used to denote the total bytes of an integrated routing table.

For distributed routing table we define entry of PTas $PTEntry = \{Destination, Mask, NHI\}$ and $Length_1$ as the total bytes of all of the fields in *PTEntry*; then the total bytes of PT can be calculated as

$$Size_1 = Length_1 * \sum_{i=1}^{K} NHIPCount[i]. \tag{3}$$

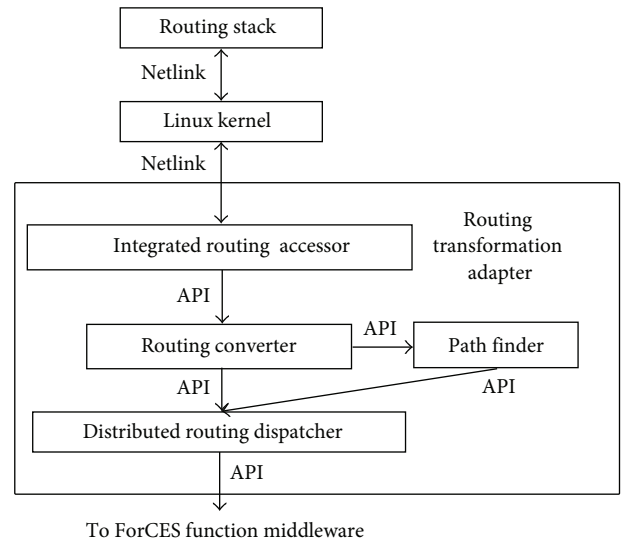

Figure 3: The structure of routing transformation adapter.

In distributed routing table, entry of NHT is defined as $NHTEntry = \{NHI, OID, Flag, MTU, NHIP\}$ and $Length_2$ denotes the total bytes of all of the fields in NHTEntry; then the total bytes of NHT are calculated as $Size_2 = Length_2 * K$.

So far we can deduce the compression efficiency which is denoted by η when an integrated routing table converts into a distributed one. Equation (3) is the formula of η:

$$\eta = \frac{Size_0}{Size_1 + Size_2} = \frac{Length_0 * N}{Length_1 * \sum_{i=1}^{K} NHIPCount[i] + Length_2 * K}. \tag{4}$$

From (4) we can clearly see that the value of η is determined by a function composed of three variables: N, K, and distribution of the *NHIP*.

The reality is that the routing stack in CE uses integrated routing table while FE uses the distributed. In order to make up the difference, *routing transformation adapter* is introduced in the paper and the software structure is depicted in Figure 3. The primary functions of *routing transformation adapter* are dedicated to transform integrated routing table to distributed one and dispatching it. In the former, integrated routing entries generated by the *routing stack are* located in Linux kernel; then *integrated routing accessor* in *routing transformation adapter* will communicate with Linux kernel bidirectionally through *Netlink* to read and write them. The integrated routing entries provided will be converted to distributed ones by *routing converter*. In our design for every distributed routing entry, the operating will be mapped to the modification of *PTEntry* and *NHTEntry*. Both *PTEntry* and *NHTEntry* are attributes of LFBs such as LPM (longest prefix match) and NextHop [4, 17]. In Figure 3, *path finder* calculates to get the ID of attribute in the format of ForCES path. While calculating, it is very difficult to locate the destination (i.e., FE) that the routes need to be sent to. For this problem, FE ID and Port ID should be used simultaneously

while numbering virtual interface. In this way, FE IDs can be extracted from the outgoing virtual interface-OID included in *NHTEntry*. Based on the new calculated ForCES paths, *distributed routing dispatcher* maps the configuration of *PTEntry* and *NHTEntry* in distributed routing table to the modification of LFB's attributes [2]. Figure 3 interacts with *ForCES functional middleware* in Figure 1; *distributed routing dispatcher* sometimes sends ForCES *control* message to FE to modify LFB's attributes via ForCES *function middleware*. Now we give the detailed example of adding a routing entry as follows.

(1) Searching for *NHIP* in *NHT* to get the corresponding *NHTEntry*. If it is found, return the *NHI* contained in *NHTEntry*. If not, allocate a new *NHI* and create an *NHTEntry*.

(2) Searching for the result of *Destination* and *Mask* to get the corresponding *PTEntry*. If it is found, end the procedure and return. If not, create a new *PTEntry* according to *Destination*, *Mask,* and *NHI*.

4. Evaluation

To evaluate the effect imposed by the mechanisms of redirection, we build a testbed shown in Figure 4. Both CE and FE are implemented by PC with Linux installed and configured with Intel core (TM) 2 Quad 2.8 GHz CPU and 2.96 GB RAM. Connect one port of FE to SMB1-2 port of SmartBits 600 which is manufactured by Spirent Network Ltd, USA. In SmartBits, TeraRouting simulates network topology and tracks the exchanging of OSPF link state via SMB1-2 port; OSPF link state packets going through the redirect channel will be encapsulated and decapsulated over and over, especially the routing scale increasing to a certain degree, extra cost of computation time will be unavoidably introduced into system. In our testing-scenario design, when TeraRouting changes the network topology, the network scale and number of route entry will change accordingly. Here we only have OSPF network LSAs [18] custom made to generate route entries. In OSPF specification every OSPF network LSA corresponds to a route entry and several LSAs will be encapsulated in one OSPF update packet entering the ForCES redirection channel. As a result, we can envisage that overhead of CPU of CE and FE will increase, and more bandwidth will be occupied by redirection process, and also the convergence time of OSPF with state "FULL" will be delayed. Here network scale mentioned in the behind is only by the number of route entries, by monitoring these factors, we capture the view of results from Figures 5 to 9.

Adding an additional redirection process between CE and FE adds overhead to the system. However, as a result of our design, the redirection mechanism does not add overhead to the data path of FE. To quantify this overhead, we measure the increased CPU utilization and FE's forwarding latency and bandwidth cross between CE and FE for network scale changing with and without the redirection.

Figure 5 shows the variation of overhead of CPU, obviously seeing that with redirection mechanism, CE and FE are more sensitive to network scale.

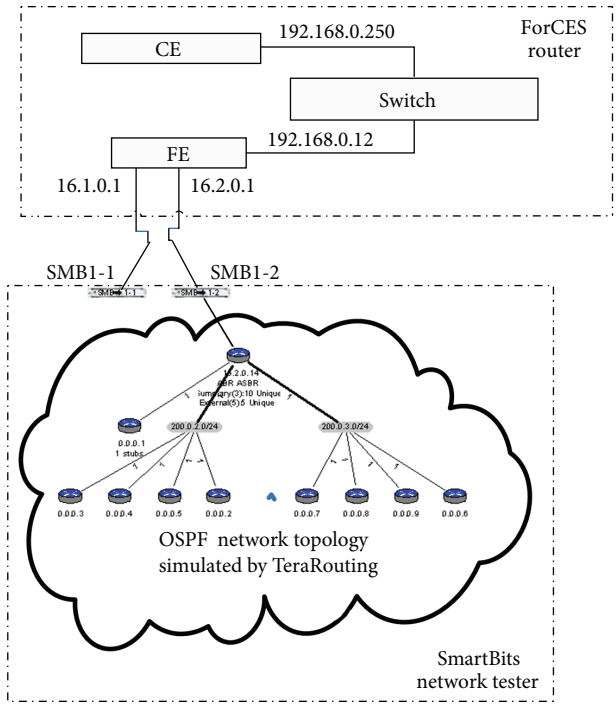

Figure 4: The construction of testbed.

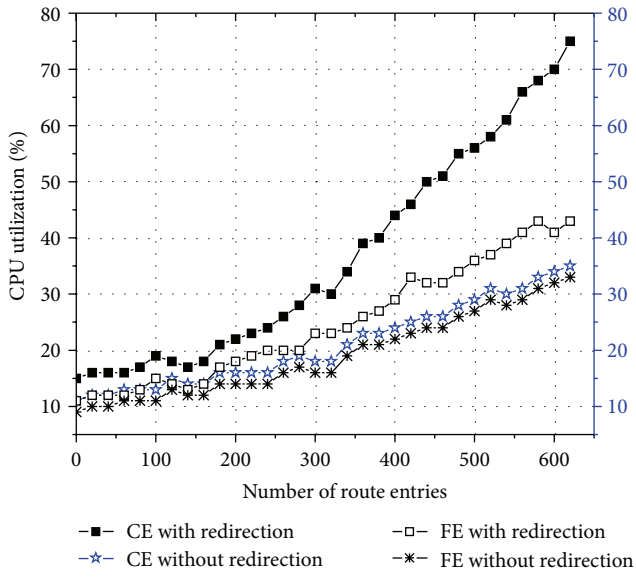

Figure 5: The view of overhead appended to CPU.

Our results (Figure 6) show that the redirection increases FE's forwarding latency, that is, the additional transfer time from SMB 1-1 to SMB 1-1, by about 16 ms on average. For latency sensitive applications, for example, web services in large data centers, 16 ms may be too much overhead.

All OSPF protocol messages including network LSA will be encapsulated as ForCES redirection message during redirection channel. However one determinative factor is the "*Length*" field in ForCES protocol message header [14]; it is a 16-bit-long integer, which means that the total ForCES

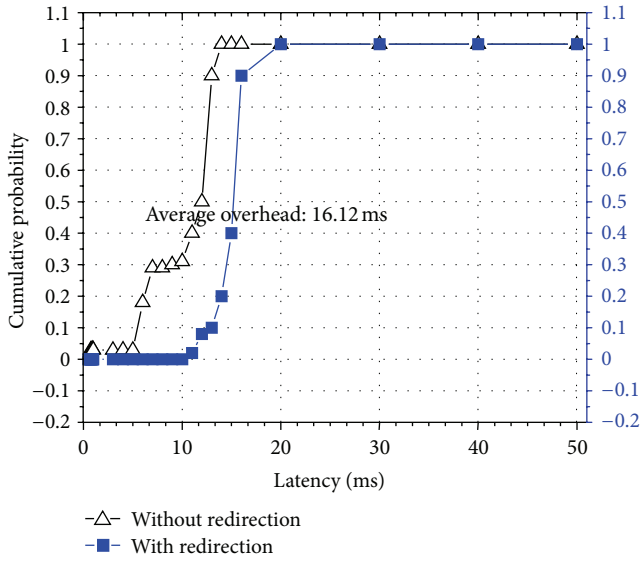

Figure 6: CDF of redirection overhead for FE's forwarding rate.

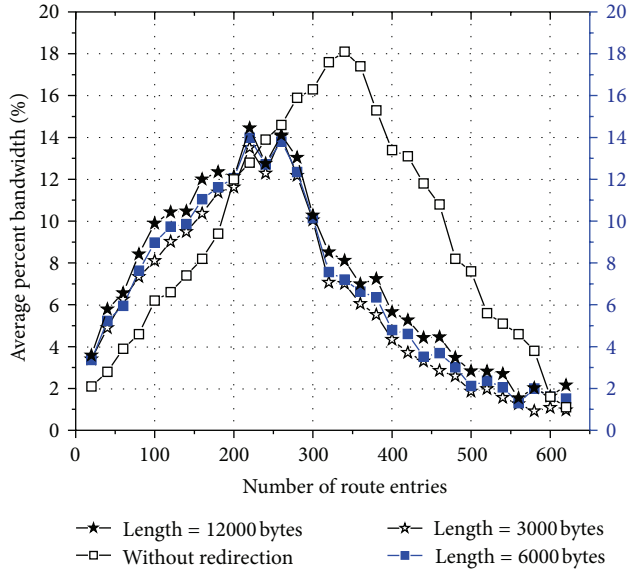

Figure 7: Effect of bandwidth occupation.

message length can reach the maximum, 2^{16}. Obviously much more length can increase the transmission efficiency of redirection message, namely, the bandwidth cross between CE and FE, also to say that the ratio of bandwidth occupation is direct to *Length* to some extent. However when the network scale sharply goes upward, very large quantity of Network LSA appears and more exchanging time will be needed, finally the average bandwidth occupation will decrease. To validate the effect of bandwidth cross between CE and FE, the view of Figure 7 just reflects the analysis above.

Figure 8 shows the convergence time when redirection mechanism is used or not. It is obviously seeing that the latter needs much more convergence time. When the number of routing entries is no more than 300, the two curves are close to each others which indicates that the cost caused by redirection mechanism is trivial. However, when the scale of routing entries continues to increase, the influence of redirection gradually appears and the latter curve becomes more and more steep. In particular, when the number of routing entries reaches 640, the former's convergence time is 18 seconds while the latter's is up to about 50 seconds.

After the complete convergence of OSPF on CE, CE begins to synchronize routing entries with FE via routing transformation adapter module, and the routing synchronization time will be changed with the scale of testing routing entries. Figure 9 gives the variation of synchronization time. When the number of routing entries is no more than 500, the curve represents the character of linearity which is approximated to be a 40 degrees line and each increment of 20 routing entries brings about about 8 ms cost. However, with the continuous increment of testing routing entries, the linearity disappears gradually and the increment of synchronization time speeds up influenced by CPU, memory load, and so forth. When the scale reaches 620, synchronization time is 618 ms which is still tolerant and does not exert great influence on the overall performance.

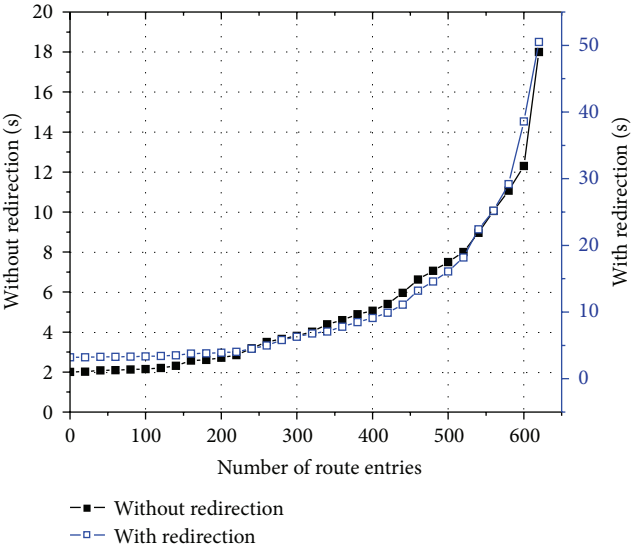

Figure 8: Convergence time of OSPF on CE.

5. Conclusion

We originally set out to answer two specific problems in ForCES architecture: (1) How can we make ForCES router exchanging routing protocol message (OSPF for example) with external in the manner of separation between forwarding and Control Element? (2) How can we realize the synchronization of routing from CE to multiple FEs?

We concluded earlier that if external merchant router wants to exchange OSPF packets with ForCES, then ForCES CE needs a way to make OSPF packets go across ForCES FE with no information lost. So we concluded that we should (1) give: IP tunnel-based redirection mechanism, a way to encapsulate OSPF packet as ForCES redirection message, (2) map physical ports in FE into virtual ports in CE,

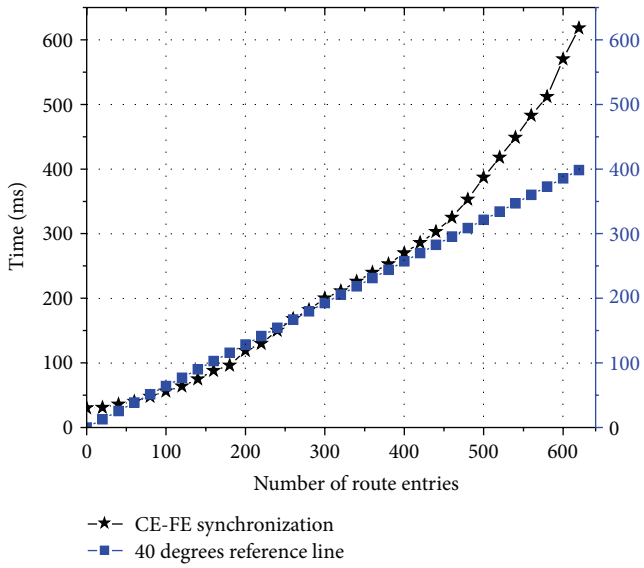

Figure 9: Synchronization time between CE and FE.

and (3) manage and number virtual ports in a special manner with FE ID included, this manner of number will be helpful to locate the outgoing FE during routing synchronization from CE to FE.

Acknowledgments

This work was supported in part by a Grant from the National Basic Research Program of China (973 Program) (no. 2012CB315902) and the National Natural Science Foundation of China (nos. 61102074 and 61170215).

References

[1] L. Yang, R. Dantu, and T. Anderson, "Forwarding and Control Element Separation (ForCES) Framework," http://datatracker.ietf.org/doc/rfc3746/.

[2] J. Halpern and J. H. Salim, "Forwarding and Control Element Separation (ForCES) Forwarding Element Model," http://tools.ietf.org/html/rfc5812.

[3] C. Kim, M. Caesar, and J. Rexford, "Seattle: a scalable ethernet architecture for large enterprises," *ACM Transactions on Computer Systems*, vol. 29, no. 1, article 1, 2011.

[4] L. Dong, B. Zhuge, and W. Wang, "Research on logical function blocks in ForCES-based routers," in *Proceedings of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD '07)*, pp. 172–177, Qingdao, China, August 2007.

[5] A. Ivan, C. Raul, and C. Walter, "Optical switch emulation in programmable software router testbed," *Photonic Network Communications*, vol. 25, no. 1, pp. 10–23, 2013.

[6] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *IEEE Internet Computing*, vol. 17, no. 2, pp. 20–27, 2013.

[7] Open Network Foundation, https://www.opennetworking.org/.

[8] Software-Defined Networking Research Group (SDNRG), http://irtf.org/sdnrg.

[9] S. Xing, C. Y. Zhang, and G. M. Jiang, "Research on OpenFlow-based SDN technologies," *Journal of Software*, vol. 24, no. 5, pp. 1078–1097, 2013.

[10] M. F. Bari, R. Boutaba, R. Esteves et al., "Data center network virtualization: a survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 2, pp. 909–928, 2013.

[11] "Framework for DC Network Virtualization," http://datatracker.ietf.org/doc/draft-ietf-nvo3-framework/.

[12] W. Wang, L. Dong, B. Zhuge et al., "Design and implementation of an open programmable router compliant to IETF forces specifications," in *Proceedings of the 6th International Conference on Networking (ICN '07)*, p. 82, Martinique, France, April 2007.

[13] R. Shea and J. Liu, "Network interface virtualization: challenges and solutions," *IEEE Network*, vol. 26, no. 5, pp. 28–34, 2012.

[14] "ForCES Protocol Specification," http://datatracker.ietf.org/doc/rfc5810/.

[15] C. Li, S. Zhang, and W. Wang, "Network calculus based performance analysis of ForCES SCTP TML in congestion avoidance stage," *Information Technolodgy Journal*, vol. 12, no. 4, pp. 584–593, 2013.

[16] B. Xiao, W. Wang, and M. Gao, "The research and implementation of the layout for ForCES FE-Intra topology," *Journal of Convergence Information Technology*, vol. 7, no. 19, pp. 487–496, 2012.

[17] ForCES LFB Library, http://datatracker.ietf.org/doc/rfc6956/.

[18] J. Doyle and J. Carroll, *Routing TCP/IP*, vol. 1, Pearson Education Press, Upper Saddle River, NJ, USA, 2nd edition, 2007.