International Journal of Future Generation Communication and Networking
Vol. 9, No. 1 (2016), pp. 271-284
http://dx.doi.org/10.14257/ijfgcn.2016.9.1.24

A Method of ForCES Virtualization[1]

Rong Jin[1, 2], Xiongxiong He[1], Ming Gao[2]

[1]*College of Information, Zhejiang University of Technology, Hangzhou, China, 310014*
[2]*College of Information & Electronic Engineering, Zhejiang Gongshang University, Hangzhou, China, 310018*
jinrong@zjgsu.edu.cn; hxx@zjut.edu.cn; gaoming@zjgsu.edu.cn

Abstract

Programmability and virtualization have both become the trends of next generation networks. ForCES (Forwarding and Control Element Seperation) is a new programmable framework for network element. We introduce virtualization into ForCES, propose a method of ForCES virtualization, design its framework, implement a prototype system named vForTER, and test its performance in different scenarios. Test results show that we can flexibly build isolated virtual routers in one physical ForCES network element. And the influences to UDP forwarding rate, TCP out of order packets rate, and RTT are very limited. ForCES network elements supporting with both programmability and virtualization can be used as core infrastructure in future networks.

Keywords: *ForCES, virtualiztion*

1. Introduction

The ossification of traditional Internet has led to difficulties in the deployment of new businesses, so programmability and virtualization ideas are put forward. Network programmability allows network operator to control the network and define its behavior. Network virtualization allows network operator to build virtual or logical networks over a physical infrastructure. Network programmability and network virtualization can meet the needs of new applications and technologies for future network.

As an idea of forwarding and control element separation, IETF ForCES is proposed for breaking the black box of network equipment. It is a new architecture of network equipment. According to the IETF definition of ForCES [1], NE (Network Element) is composed of CEs (Control Element) and FEs (Forwarding Element), FE resources are abstracted into LFBs (Logical Function Block), CEs control the LFBs in FEs by standard ForCES protocols, which provide programmable ability.

This paper introduces virtualization technology into ForCES, and proposes a method of ForCES virtualization. ForCES network elements supporting with both programmability and virtualization can be used as core infrastructure in future networks.

In the research related to ForCES and virtualization, two directions are being studied. One direction focuses on the application of ForCES as virtualization tools and frameworks. Francesco implements an open-source software virtualization tool for programmable modular router, and ForCES is applied in his programmable modular router [2]. Evangelos applies ForCES to NFV (Network Function Virtualization) of ETSI

[1] This work was supported in part by a grant from the National Basic Research Program of China (973 Program) (No. 2012CB315902), the National Natural Science Foundation of China (No.61379120), Zhejiang Leading Team of Science and Technology Innovation (No.2011R50010-11, No.2011R50010-15). Zhejiang Provincial Key Laboratory of New Network Standards and Technologies (NNST)(No.2013E10012).Youth Foundation of Zhejiang Gongshang University(No.QZ13-8)

ISSN: 2233-7857 IJFGCN

(European Telecommunications Standards Institute) [3], his idea is that ForCES model can be applied to abstract and describe network functions, and ForCES protocol can be applied to control network functions. Another direction focuses on the virtualization of ForCES itself. B. Khasnabish submits a draft about IETF ForCES Logical Function Block Subsidiary Management [4]. LFB SM is useful for introducing and supporting virtualization of ForCES Network Element. This draft currently has only finished the definition of FEM (FE Manager) LFB, which can support the virtualization of NE. However, the other components, such as CE visor, FE visor, CE manager, and FE manager, are still not clear. In addition, its test bed platform and reference implementation are still in the state of TBD (To Be Determined). Our work focuses on the second direction. We studied on the virtualization of ForCES itself. To the best of our knowledge, we have not seen an implementation of virtualization of ForCES.

This paper makes the following three contributions. First, a method of ForCES virtualization is proposed. We introduce virtual machine technology into CE and FE, Virtual CEs and virtual FEs form virtual NEs. Second, a prototype system called vForTER (virtual Forces rouTER) is designed. We design a Switching Element to control the virtualization of CEs and FEs in our vForTER architecture, we also propose an algorithm for the resource allocation of FE. Finally, a series of scenarios are designed to test the performance of vForTER. The test results show that we can flexibly build isolated virtual routers in one physical ForCES network element. And the influences to UDP forwarding rate, TCP out of order packets rate, and RTT caused by virtualization are very limited.

The remainder of this paper is organized as follows. Section II proposes a method of ForCES virtualization. Section III presents the vForTER architecture. Section IV proposes an algorithm of resources allocation for FE. Section V illustrates the vForTER prototype system. Section VI discusses experimental results to verify and evaluate our system. Section VII concludes this paper.

2. For CES Virtualization Method

Our idea is introducing virtual machine technology into the virtualization of CEs and FEs. Isolated CE virtual machines and FE virtual machines can be created on the CE physical machine and FE physical machine. These CE virtual machines and FE virtual machines are virtual ForCES control elements and virtual ForCES forwarding elements, we can call them vCEs (virtual CE) and vFEs (virtual FE) respectively. These vCEs and vFEs can be combined into isolated vNEs (virtual NE). An example is illustrated as Figure 1. The physical ForCES NE is composed of one physical CE and four physical FEs. The CE is virtualized into four vCEs, and every physical FE is virtualized into two vFEs. As shown in Figure 1, one vCE and two vFEs are combined into one vNE, so one physical NE has been virtualized into four vNEs. If these vNEs implement router function, then we can obtain four VRs (Virtual Routers). Because different vCEs and vFEs are isolated, different vNEs are also isolated. These isolated vNEs can support different kinds of virtual networks.

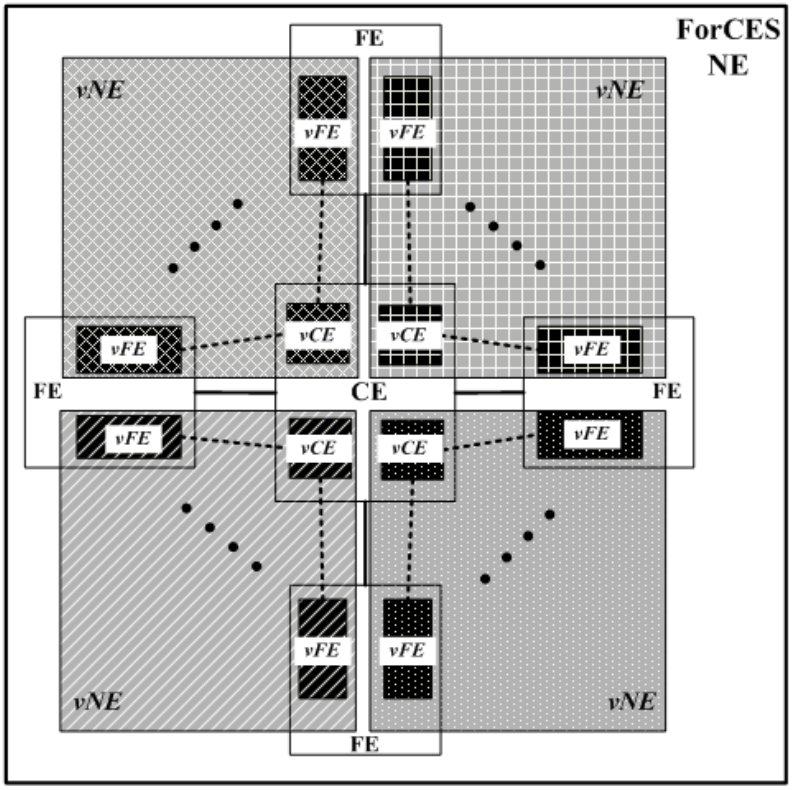


Figure 1. ForCES Virtualization

3. vForTER Architecture

Based on the method of ForCES virtualization, we design a prototype system called vForTER.

3.1 Architecture

Figure 2 shows vForTER's architecture. Our main idea is that a special FE named SE (Switching Element) is introduced in order to control and manage the virtualization of ForCES NE. SE has two major jobs. One job is to manage vCEs and vFEs and to control the creation process of vNEs. The other job is to put all external ports together and to schedule internal communication traffics.

As shown in Figure 2, vForTER carries three virtual networks. So vForTER creates three VRs, each VR is composed of one vCE and several vFEs. vCEs locate on CE, each vCE is a virtual machine and run routing protocols independently. vFEs locate on FEs, the vFEs belong to the same CE can locate on different FEs. The allocation of FE resources depends on the forwarding requirements of VRs. For example, as Figure 2 shows, virtual network 1 and virtual network 2 have high forwarding requirements, so they are both allocated two vFEs. Virtual network 1 is allocated vFE1-1 and vFE1-2, and virtual network 2 is allocated vFE2-1 and vFE2-2. Virtual network 3 has lower forwarding requirement, so it is allocated only one vFE, which is vFE3-1.

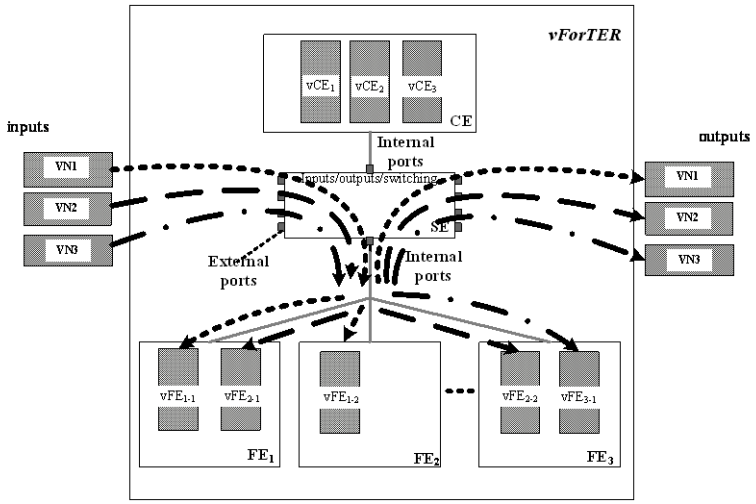


Figure 2. vForTER Architecture

SE is not only responsible for the allocation of vCEs and vFEs, but also responsible for the scheduling of the internal communication traffics. All the communications through vForTER are put together on SE, including vCE-vFE communication and vFE-vFE communication. SE distributes them according to their virtual network tags. For example, when SE receives a packet, it reads the packet's virtual network tag, and distributes it to the corresponding vFE for further processing (for example, routing lookup or traffic shaping). After vFE's processing, the packet added with special virtual network tag is sent back to SE. Then SE reads the tag and distributes it.

3.2 SE Architecture

Figure 3 shows the architecture of SE. SE is the scheduling center of the whole vForTER. It is composed of one classifier, one distributor, and one VN-FE-vFE mapping table. The VN-FE-vFE mapping table is very important, because it records the relationship between virtual network, virtual router, FE, vFE and vCE. It is created when the virtual networks are created.

When packets come into SE, classifier classifies them by their sources. They are classified into two categories: 1) coming through external ports; 2) coming through internal ports.

If the packets are coming through external ports, they can be further classified into two categories: 1) If this packet is a routing protocol packet from external other router, then it will be redirected to vCE for further processing (identified with ① in Figure 3); 2) If this packet is common forwarding packet, then it will be distributed to the corresponding vFE (identified with ②).

If the packets are coming through internal ports, they can be further classified into four categories: 1) If this packet is a ForCES control message, it will be distributed to vFE (identified with ③); 2) If this packet is a routing protocol message from vCE, it will be distributed to other external router through external ports (identified with ④). 3) If this packet is a forwarding packet after vFE's processing, it will be distributed to the corresponding external port according to its tag (identified with ②). 4) If this packet is a ForCES event message produced by vFE, it will be distributed to the corresponding vCE (identified with ⑤).

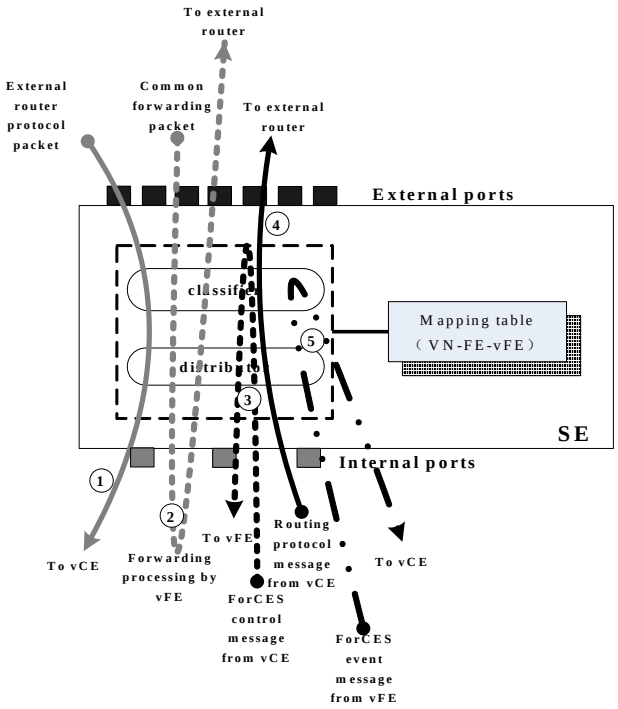


Figure 3. SE Architecture

3.3 FE Architecture

FE is the data plane of vForTER, Figure 4 shows FE architecture, in this example, one FE is virtualized into two vFEs. In our design, we use virtual machine technology to divide FE resources, so these vFEs are isolated. A virtual router's data plane can run on several vFEs. vFEs receive packets from SE and do further processing. These processing functions are provided by ForCES LFBs. These functions include address table lookup, head encapsulation and tag operation, the tag mentioned here is used by SE to classify and distribute packets.

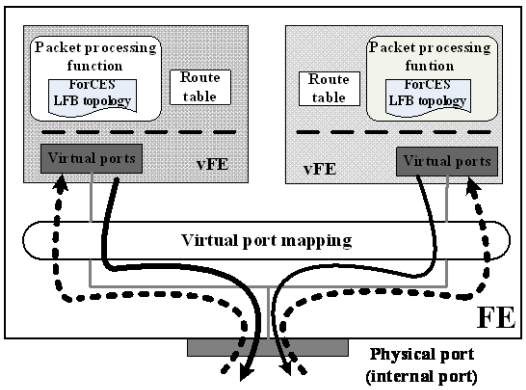


Figure 4. FE Architecture

3.4 CE Architecture

CE is the control plane of vForTER, Figure 5 shows CE architecture, in this example, one CE is virtualized into two virtual CEs. In our design, we use virtual machine technology to divide CE resources, so these vCEs are isolated. vCEs receive ForCES redirect messages from SE, each vCE can run its own routing protocol. vCE needs to add

tag on those ForCES redirect messages created by CE. The tag mentioned here is used by SE to classify and distribute packets.

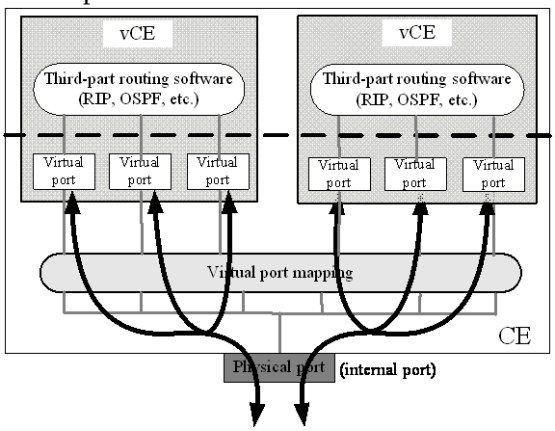


Figure 5. CE Architecture

4 FE Allocation Algorithm

It is important to allocate FE resources to several virtual routers. Parallel packet processing of multiple vFEs in one virtual router will lead to out of order packets, and it will affect the throughput rate of transport layer protocol (for example, TCP). FE allocation algorithm should try to reduce this kind of influence.

In order to reduce the out of order packets rate, the number of vFEs belonging to a same virtual router should not be too big. Suppose that our vForTER needs carry *n* virtual networks, *Ri* is the packet processing ability needed by virtual network *i*, and the processing ability of each FE is *C*. We consider the FE allocation problem as a one-dimensional bin packing problem. If *Ri=kC+ri, (ri<C)*, first, we try our bests to find *k* vFEs and each vFE uses the whole resources of a FE, and then we find minimum number FEs to pack the remaining *ri*. The detailed FE allocation algorithm is described as following.

Input: *R*, the request packet processing ability of a virtual network.
Output: a best FE allocation method.
Definition: *C*—the packet processing capacity of a whole FE.
r—rest packet processing ability.
k—the number of FEs whose abilities are allocated to one virtual network.
FE*—the best FE who can satisfy the rest packet processing ability requirement.
min—The minimum remaining packet processing ability of FEs.
AP(FE_i*)*—the remaining available packet processing ability of FE_i.
BIG-NUM—a big number which is bigger than *C*.

1 *r=R%C*；
2 *k=(R-r)/C*；
3 **for** *i*=0;*i* <*k*;*i*++ **do**
4 find an available FE, create a vFE on the FE, allocate the entire packet processing ability of this FE to this vFE, and allocate the vFE to the virtual network.
end
5 FE*=*null*;
6 *min=BIG_NUM*;
7 **for** （all FE_is that satisfy *AP*(FE_i)>*r*） **do**
8 **if** (*AP*(FE_i)-*r*)<*min* **then**
9 *min*= *AP*(FE_i)-*r*;

10 FE*=FE$_i$;
11 **end**
12 create a vFE with *r* packet processing ability on FE*, and allocate the vFE to the virtual network.

5 vForTER Prototype System

We build a prototype system called vForTER shown as Figure 6. vForTER is composed of one SE, one CE and three FEs. CE is a HP server DL380, SE and FEs are HP servers DL580. We run EXSI [5] of Vmware on all the servers, and run Linux OS on EXSI.

We run a third-part software called open vSwitch on SE. By using open vSwitch, we implement a software switch and add programs to implement classifier, distributor, and mapping table (shown as Figure 3). All external ports of vForTER are collected in SE. We redefine the "EtherType" in MAC head as a tag, which indicates the input/output ports in vForTER. SE reads the tag to distribute packets.

We use vSphere [6] to create virtual FEs. Each vFE uses Click [7] to implement the functions of packet processing. Click is a software routing system developed by Dr. Eddie Kohler of MIT. Click implements abstraction and modular processing of packet processing in routers. Click can run as a forwarding engine on common PC, server or some MIPS structures.

We use a third-part software called XORP [8] to implement route protocols in vCEs, such as RIPv1, RIPv2, OSPFv2, OSPFv3, BGP.

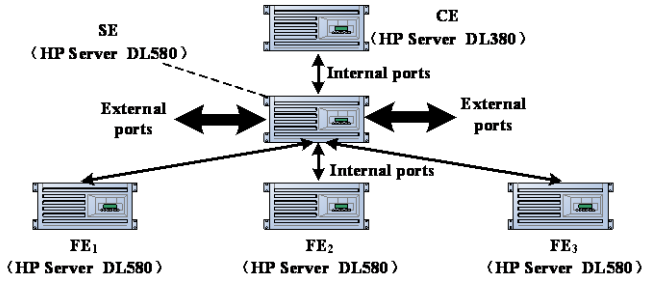


Figure 6. vForTER Prototype System

6 Test and Evaluation

vForTER implements the virtualization of ForCES, the virtualization greatly enhanced its flexibility. vForTER supports to create different virtual network to carry on different business. It can flexibly construct different VRs with different performances according to user's requirements. At the same time, the virtualization is inevitable to pay some cost of overall system performance. We build a test bed and construct four test scenarios for tests. We test UDP forwarding rate and TCP forwarding rate. The purpose of these tests is to evaluate the flexibility of ForTER. We also test total UDP forwarding rate of one FE when it is virtualized to several vFEs, TCP out of order rate, and internal packet forwarding delay in ForTER. The purpose of these tests is to evaluate the performance influence brought by the virtual method proposed in this paper.

6.1 Test Environment

Figure 7 shows our vForTER test bed. A professional SmartBits600 is used to produce TCP and UDP traffic. SmartBits600 is a professional network performance testing platform produced by Spirent Company, SmartFlow and TeraRouting can run on this platform. SmartFlow is designed for traffic measurement. It can act as a source to send

TCP or UDP traffic to external network. At the same time, it can act as a destination to receive traffic from external network. SmartFlow obtains throughput, delay, jitter and other data of external network by comparing its input and output traffic. TeraRouting can simulate various functions of router (such as OSPF, RIP, BGP, etc.). It can act as a router to interact with the tested router. We can configure the FIB information of TeraRouting to control the FIB scale of the tested router.

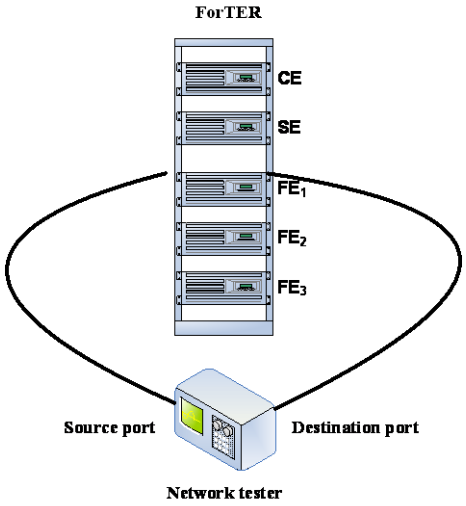


Figure 7. Test Bed

Table 1 shows all the equipments configurations of the test bed.

Table 1. Configuration List of Vforter Prototype System

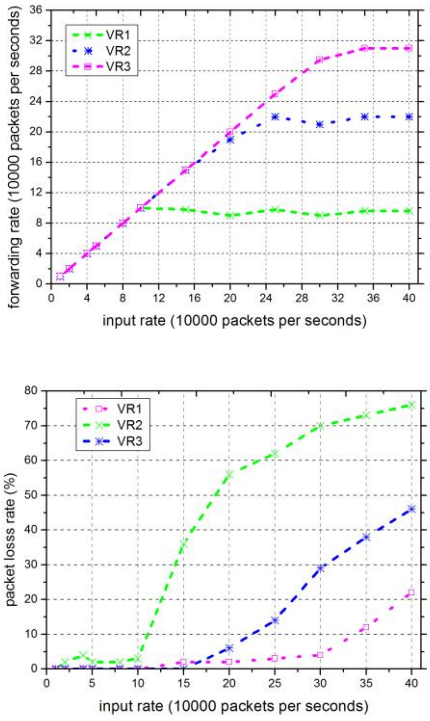
Role	Equipment type	Hardware	Software
CE	HP DL380	Processor: 2 Intel® Xeon® E5-2650 Memory: 8G DDR3 Interface: 8 1GB Ethernet interfaces	Vmware EXSI + Vmware vSphere + XORP
FE_1 FE_2 FE_3	HP DL580	Processor: 2 Intel® Xeon® E5-2650 Memory: 16G DDR3 Interface: 16 1GB Ethernet interfaces	Vmware EXSI + Vmware vSphere + Click
SE	HP DL580	Processor: 2 Intel® Xeon® E5-2650 Memory: 32G DDR3 Interface: 16 1GB Ethernet interfaces	Vmware EXSI + Open vSwitch
Network tester	Spirent SmartBits600	Machine frame: SmartBits600 portable Test board: LAN 3101B 1G	SmartFlow TeraRouting

We build three scenarios for tests. They are called VR1, VR2 and VR3. They are virtualized routers based on vForTER with different forwarding performance. VR1 is composed of one vCE and one vFE, VR2 is composed of one vCE and two vFEs, and VR3 is composed of one vCE and three vFEs. VForTER consists of one CE, one SE and three FEs with the same packet processing performance. The data plane packet processing performance of the VR which created based on vForTER depends on the performance and the number of its vFEs. Here we simplify the configuration of vFE. We suppose each vFE has the same performance, and one FE is only allocated to one vFE. Thus VR's packet processing performance will only depend on its number of vFE.

Firstly we test vForTER's performance when it forwards UDP traffic. We test the UDP forwarding rate and the packet loss rate of the three scenarios, and we also test the total performance cost when one FE is virtualized to several vFEs. Then we test vForTER's performance when it forwarding TCP traffic. We test the TCP throughput and the out of order packet rate of the three scenarios. Lastly we test the extra RTT delay brought by the introduction of SE in vForTER.

6.2 UDP Performance Test

We test UDP packet forwarding rate and packet loss rate in those three scenarios mentioned above. In our test, we use 64 Bytes UDP packets, input rate ranges from 10 thousands packets per second to 400 thousands packets per second. And we use a small FIB with 2 routing entries. Test result is shown in Figure 8, it shows that with the increase of input rate, VR3 has the best forwarding rate, VR1 is the worst. At the same time, VR3 has the lowest loss rate, and VR1 has the maximum loss rate. It shows that more vFEs bring better performance. So our virtualization method can flexibly allocate vFEs to build VRs with special performances according to user's special requirements.



(a)forwarding rate (b)packet loss rate

Figure 8. UDP Performance Test

We also change UDP packet length (512 bytes, and 1500 bytes) and FIB size (20 thousands routing entries) for testing. The test results show the similar trend, that is, with the increase of vFE number, the performance is better.

In order to study the influence of virtualization, we test the total performance cost when a FE is virtualized into multiple vFEs. We change vFE number from 1 to 8. Figure 9 shows the total forwarding rates of the FE, it shows that with the increase of vFE number, the total forwarding rate declines slightly. The reason is that FE's CPU needs to switch frequently between those vFEs, and this kind switching brings extra system overhead and thus results in a slight decline in the whole FE forwarding rate. Test data shows that the whole FE forwarding rate will decrease by 10% when the FE is virtualized into 8 vFEs.

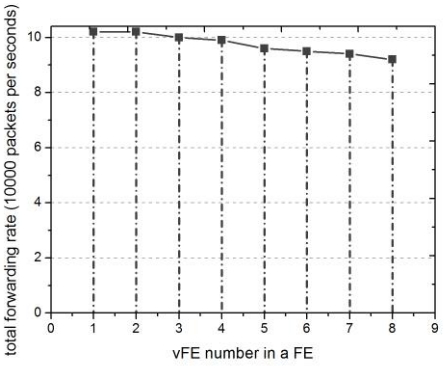


Figure 9. The Total Forwarding Rate of One FE When it is Virtualized to Several Vfes

UDP performance tests show that vForTER can flexibly construct different VRs with different performances according to user's requirements, at the same time it unavoidably pays a little performance cost on the whole.

6.3 TCP Performance Test

Most of network applications use TCP as their underlying transport protocol. We test TCP throughput in the three scenarios, Figure 10 shows the result, VR3 has the best throughput, and VR1 is the worst. It shows more vFEs bring better performance. So our virtualization method can flexibly allocate vFEs to construct VRs with special performances according to user's special requirements.

We also change FIB size (from 2 routing entries to 20 thousands routing entries) for testing. The test results show the similar trend, that is, with the increase of vFE number, the performance is better.

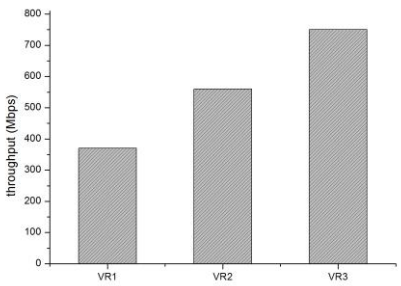


Figure 10. TCP Throughput Test

vForTER can flexibly configure vFEs to meet user's forwarding performance requirement, but out of order packets will occur when a VR uses multiple FEs, and packets reordering is a huge challenge for a parallel processing system. So we test the out of order TCP packets rate of VR1, VR2, and VR3. Test result is shown as Figure 11. VR1 has a very small out of order packets rate because it uses only one FE, while VR2 and VR3 have obvious bigger out of packets rate because they both use multiple FEs. The out of packets rate of VR2 is about 10%, and the VR3's is about 11%. There is no obvious difference between VR2 and VR3.

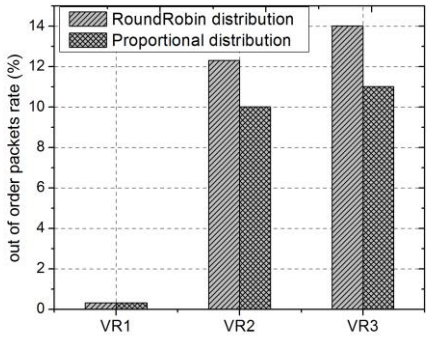


Figure 11. TCP Out of Order Packets Rate Test

The influence of out of order packets rate can be reduced by using appropriate scheduling strategy in SE. We test the out of order packets rate under two different scheduling policies, as shown in Fig 11. One scheduling strategy is called RoundRobin distribution. If a VR has three vFEs, the RoundRobin strategy will distribute equal packets to these three vFEs round in turn. The other scheduling strategy is called Proportional distribution. If a VR has three vFEs, the Proportional strategy will distribute different numbers of packets to these three vFEs according to their CPU capacities. Figure 11 shows test result. In the VRs containing more than one vFE, the Proportional distribution can more effectively reduce out of order packets rate than the RoundRobin distribution.

TCP performance tests show that, vForTER can flexibly construct different VRs with different performance according to user's requirements. At the same time it brings out of order packets phenomenon when VR use multiple FEs, we can use the Proportional distribution scheduling strategy in SE to reduce the impact of packet reordering.

6.4 Internal Packet Forwarding Delay in Vforter

Our virtualization method of ForCES introduces a SE to manage virtualization procession. When packets pass through VR, firstly they need to go through SE, then they are distributed to FE by SE, finally they come back from FE to SE and are forwarded from SE to outside of VR. The packets moving forth and back between SE and FE is bound to increase processing delay. So we test the packet forwarding delay of vForTER. As shown in Figure12, we test RTT in four scenarios, which are VR1, VR2, VR3, and R1. R1 is a ForCES router without virtualization, and it is composed of one CE and one FE. There is no SE in R1. We compare VR1, VR2 and VR3 with R1 to evaluate the increase of RTT brought by the virtualization method. As shown in Figure12, the results show that the average increase of RTT is about 0.17 milliseconds. Reference [9] finds that the distance between most of hosts in the Internet is about 14 hops, and the average RTT is about 80 milliseconds. Thus, if our VR is deployed into the Internet, each node will increase the RTT of 0.17 milliseconds, and then the overall RTT will increase by 2.4

milliseconds. Comparing with the RTT of the original 80 milliseconds, the RTT increase of 2.4 milliseconds is small. So the internal packet forwarding delay in vForTER is very small.

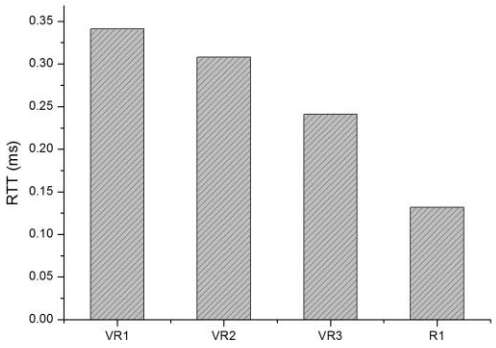


Figure 12. RTT Test

7. Conclusion

This paper introduces virtualization into ForCES, and proposes a method of ForCES virtualization. A SE is introduced into ForCES architecture to manage its virtualization. This paper also implements a prototype system named vForTER according our method of ForCES virtualization. vForTER can flexibly construct isolated VRs based on one physical ForCES NE. Tests show that vForTER can flexibly construct different VRs with different performances such as UDP forwarding rate and TCP throughput. The performance costs of virtualization such as the degradation of overall UDP forwarding rate, TCP out of order packets rate, and the increase of RTT are limited. So our method can realize virtualization and guarantee isolation with limited performance cost. It is a practical method of ForCES virtualization. Our method of ForCES virtualization makes ForCES have both programmability characteristic and virtualization characteristic, thus makes ForCES equipments can be used as core equipments in future networks.

There are three types of virtualization, full virtualization, Para virtualization and OS-level virtualization. We use full virtualization in our method of ForCES virtualization. The next work can use Para virtualization or OS-level virtualization to improve performance.

References

[1] A. Doria, J. Hadi Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal and J. Halpern, "IETF RFC 5810: ForCES Protocol Specification", http://tools.ietf.org/search/rfc5810, **(2010)**.

[2] F. Apollonio, C. Caini, W. Cerroni, D. Lacamera and C. Raffaelli, "Open-Source Software Virtualization Tools for Programmable Modular Router Implementation and Testing", http://www.telematica.polito.it/oldsite/Italian_Networking_Workshop_2012/Pdf/s10_p2.pdf, the 9th Italian Networking Workshop, Courmayeur, Italy**, (2012)**, pp. 11-13.

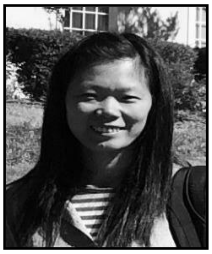
[3] E. Haleplidis, S. Denazis, Odysseas, D. Joachimpilla and J.H. Salim, "ForCES Applicability to SDN-Enhanced NFV", 2014 Third European Workshop on Software-Defined Networks, Budapest, Hungary, **(2014)**, pp. 1-3.

[4] B. Khasnabish, E. Haleplidis and J.H. Salim, "IETF ForCES Logical Function Block (LFB) Subsidiary Management", draft-khs-forces-lfb-subsidiary-management-00.txt. http://zinfandel.levkowetz.com/pdf/draft-khs-forces-lfb-subsidiary-management-00.pdf, **(2014)**.

[5] A. Qasim, K. Vladimir, S. Josh and Z. Puneet, "Performance evaluation of HPC benchmarks on VMware's ESXi server", Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7155 LNCS (PART 1), **(2012)**.

[6] Nishikiori and Masaaki, "Server virtualization with VMware vSphere 4", Fujitsu Scientific and Technical Journal, vol. 47, no. 3, **(2011)**.
[7] K. Joongi, H. Seonggu, J. Keon and P. KyoungSoo, "The power of batching in the click modular router", Proceedings of the Asia-Pacific Workshop on Systems, **(2012)**.
[8] T. Xiuhui, W. Jianping, C. Yong and X. Mingwei, "MPBGP extensions on open source routing software XORP", Qinghua Daxue Xuebao/Journal of Tsinghua University, vol. 49, no. 11, **(2009)**.
[9] A. Fei, G. Pei, R. Liu and L. Zhang, "Measurements on delay and hop-count of the internet", IEEE GLOBECOM'98-Internet Mini-Conference, Sydney, Australia, **(1998)**, pp. 8-12.

Authors



Rong Jin, She received her B.S. degree in physics from Shaanxi Normal University, China in June 2000 and her M.S. degree in communication and information system from Zhejiang University of Technology, China in March 2003. She is currently working towards his Ph.D. degree in control theory and control engineering at Zhejiang University of Technology, China. Her current research interest includes open programmable networks and Software Defined Network.

Xiongxiong He, He received the M.S. degree from Qufu Normal University in 1994 and the Ph.D. degree from Zhejiang University in 1997. From 1998 to 2000, he held a postdoctoral position in Harbin Institute of Technology. He joined Zhejiang University of Technology in 2001, and since then he has been working as a professor in the College of Information Engineering of ZJUT. His research areas include control theory and signal processing.

Ming Gao, He received his M.S. degree in communication and information system from Zhejiang Gongshang University in 2005 and his Ph.D. degree from Zhejiang University in 2014. His current research area is Software Defined Network.