Wang WM, Dong LG, Zhuge B. Analysis and implementation of an open programmable router based on forwarding and control element separation. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 23(5): 769–779 Sept. 2008

Analysis and Implementation of an Open Programmable Router Based on Forwarding and Control Element Separation

Wei-Ming Wang (王伟明), Li-Gang Dong (董黎刚), and Bin Zhuge (诸葛斌)

Institute of Networks and Communications Engineering, Zhejiang Gongshang University, Hangzhou 310018, China

E-mail: {wmwang, donglg, zhugebin}@mail.zjgsu.edu.cn

Received February 12, 2008; revised July 8, 2008.

Abstract A router architecture based upon ForCES (Forwarding and Control Element Separation), which is being standardized by IETF ForCES working group, gains its competitive advantage over traditional router architectures in flexibility, programmability, and cost-effectiveness. In this paper, design and implementation of a ForCES-based router (ForTER) is illustrated. Firstly, the implementation architecture of ForTER is discussed. Then, a layered software model, which well illustrates ForCES features, is proposed. Based on the model, design and implementation of Control Element (CE) and Forwarding Element (FE) in ForTER are introduced in detail. Moreover, security for ForTER is considered and an algorithm to prevent DoS attacks is presented. Lastly, experiments of ForTER are illustrated for routing and running routing protocols, network management, DoS attack prevention, etc. The experimental results show the feasibility of the ForTER design. Consequently, the ForTER implementation basically testifies the feasibility of ForCES architecture and some IETF ForCES specifications.

Keywords computer network, router, ForCES, open programmable, protocol, architecture, network element

1 Introduction

Several demands from Internet service providers and network end-users have made a great impact upon the equipment designs for next generation IP networks[1,2]. The demands for next generation network equipments (formally called Network Element, NE) can be described as: 1) being flexible enough for the deployments of ever growing new services; 2) being open and modular enough so that the equipment market is hard to monopolize, and the equipment prices can be reduced; 3) being capable of providing QoS services in order to support real-time applications such as multimedia transmissions[3].

Open programmable networks[4] are considered as the prospective architecture to meet the above demands. In open programmable networks, an NE (e.g., a router/switch) is systematically separated into a control plane and a forwarding/switching plane. Forwarding/switching plane receives packets from outer networks, processes the packets according to functional requirements of the NE, and then outputs the packets back to outer networks. Forwarding/switching plane usually needs the ability to process packets at line speed. Control plane controls forwarding/switching plane by providing adequate parameter values for the process. More importantly, the interface between the control plane and the forwarding/switching plane is standardized. Moreover, resources at the forwarding/switching plane, which are used for processing packets, are also described in a standardized way. As a result, control plane can access and control the forwarding/switching plane resources in a standard way. This makes it feasible for control plane and forwarding/switching plane to be separated at their product level, i.e., control plane and forwarding/switching plane as separate products from different vendors can work together to form an NE with full interoperability.

Accordingly, researches on open programmable networks are focused on the way to standardize the interface between control plane and forwarding/switching plane and to setup a model to standardize the resources in forwarding/switching plane.

Ideas in early Opensig[5], active networks[6], and virtual networks[7] helped form the basic concept of open programmable networks. Opensig emphasizes the standardization of interfaces, active networks more on the programmability of network functions, and virtual

Regular Paper

Supported by the National Natural Science Foundation of China under Grant Nos. 60573116 and 60603072, the National High Technology Development 863 Program of China under Grant No. 2007AA01Z201, the Zhejiang Provincial NSF China under Grant Nos. Z106829, Y106747, and the Zhejiang Sci & Tech Project under Grant No. 2006C11215.

networks on providing QoS by allocation or reservation of network resources.

Researches in the past years on the standardization of interfaces and resources in open programmable networks include IEEE P1520[8], Multiprotocol Over ATM (MPOA)[9], General Switch Management Protocol (GSMP)[10] for IP Switching[11], Multi-service Switching Forum (MSF)[12], and Forwarding and Control Element Separation (ForCES)[13]. Among them, the ForCES, which is a working group under Routing Area of IETF, has made the most prominent achievements for open programmable networks. Currently, it has been widely accepted as the most typical solution to open programmable networks. ForCES is also supported by Network Processing Forum (NPF)[14], ITUT NGN Focus Group[15,16], and Intel IXA[17].

ForCES proposes that an NE (typically a router) consists of several Control Elements (CEs), among which one acts as an active controller and others as backups and multiple Forwarding Elements (FEs). Multiple FEs in forwarding plane are interconnected together, forming a distributed forwarding plane FE topology to fulfill complex packet forwarding tasks. A standardized interface called ForCES protocol[18] is defined for the communication between CEs and FEs. Up to the present, the ForCES group has completed the work of ForCES requirements (RFC 3654) and ForCES Framework (RFC 3746). The work of ForCES Protocol Specification (i.e., ForCES protocol) and ForCES FE Model[19] is also close to completion.

The concept of ForCES was investigated quite intensively in recent years. R. Haas *et al.*[20,21] proposed Web Service-Based CE architecture, in which Web Service interfaces were mapped to ForCES messages for dynamic service deployment. C. Chrysoulas *et al.*[22] presented a modular node architecture and specified the description interface of nodes. Their aim is to make new services easily added by inserting modules that have the appropriate functionality. The work of [20–22] is part of the FlexiNET IST project. On the ForCES-based distributed router in [23, 24], Hidell *et al.* presented measurements on the distribution of large routing tables in an experimental platform consisting of one CE and up to 16 FEs. J. Fu *et al.*[25] specified a programming model for FE. This programming model attempted to uniform the API of processing blocks in FE. In [26], J. Fu *et al.* used the simulation environment to evaluate throughput, latency, and utilization between pooled and pipelined processing blocks approaches.

In comparison with previous researches, presented in this paper is an implementation of an open programmable router that is based on ForCES architecture and ForCES protocol[27]. The implementation is called ForTER — a ForCES-based rouTER. Key design issues for ForTER are discussed and implementation details of the key elements are presented in this paper. Background of this paper is highly related to IETF ForCES work. As key participants in the ForCES work and authors of the ForCES protocol, we set the motivation of the ForTER implementation as the evaluation of the ForCES protocol and the associated ForCES architecture. ForCES is currently in urgent need of such an implementation to evaluate its protocol designs.

Section 2 describes the architecture of ForTER. Section 3 presents the implementation details of key elements in ForTER. Section 4 introduces some experiments and test results. Section 5 is the conclusion.

2 Architecture

2.1 Forwarding and Control Element Separation

Forwarding and Control Element Separation is the key for the architecture of ForTER. We show the architecture as in Fig.1, which complies with the ForCES Framework defined in RFC 3746. In this architecture, there exists one primary CE. There may also be some redundant CEs for system high availability purpose. There are multiple FEs (may be up to hundreds of FEs for core routers), which are separated from CEs. The interface between CE and FE is to be standardized and the communication protocol upon it is called "ForCES protocol". The CE is responsible for the management of FEs by use of the ForCES protocol. The Fr is the interface connecting CEs, and the Fi connecting FEs. Outer networks connect ForTER via Fi/f interfaces. After the ForCES protocol has been standardized, it will make possible CE and FE being manufactured by different vendors.

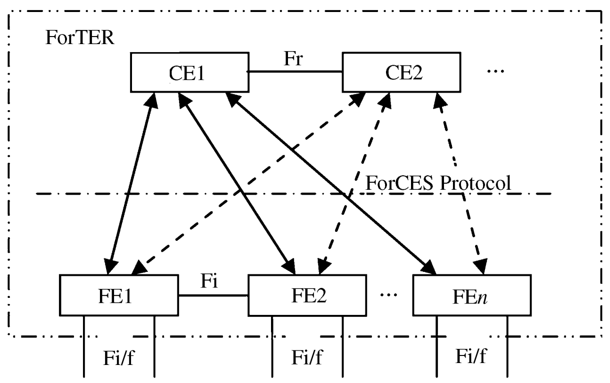


Fig.1. Architecture of ForTER.

FE is further standardized by ForCES FE Model[19]. The resources in an FE are represented by various kinds of standardized Logical Function Blocks (LFBs), each of which has specific functions for IP packet forwarding. Typical LFBs are classifier LFB, scheduler LFB, IPv4 or IPv6 forwarder LFB and so on. Multiple LFBs are interconnected via datapaths to form an LFB topology in FE, so that the FE can carry out a complex process on packets. CE is responsible for the management of LFBs in FE. The management of LFBs includes the configuration and query of LFB attributes, capabilities, or events. ForCES protocol makes it possible for CE to dynamically manage the LFBs and the associated LFB topology, such as to add/remove/modify some LFBs, their attributes, and the associated LFB topology. In this way, CE is able to provide various IP forwarding services in an "on the fly" way by configuring the very bottom resources of FE. Note that this management ability of CE to FE resources is far beyond the management ability that ordinary router console or IOS can provide. This makes ISPs able to configure new services in a very handy, economic, and flexible way.

Mostly, FE forwards packets from some Fi/f to some other Fi/f, but in some cases, FE also have to redirect some packets to CE. Packets of these are usually packets for routing protocols or network management protocols. These packets will be encapsulated as ForCES protocol messages and redirected to CE then. A special security issue called DoS attack from redirect messages is induced by the redirect packets[28]. Subsection 3.3 will describe it and provide a solution to it.

2.2 Layered Model

IETF ForCES proposed the architecture and the protocol for forwarding and control element separation. However, a detailed software model based on the architecture is essential for any ForCES implementations and has not been presented yet. We present here a layered software model which is applied to ForTER and is also fit for other ForCES implementations.

The layered software model of ForTER is shown in Fig.2. As in the figure, the model is coarsely layered as CE and FE, while FE is divided into FE SlowPath and FE FastPath. FE FastPath is responsible for the packet processing that needs running at line speed. The behavior of FE FastPath is flexibly controlled by CE via FE SlowPath.

The Application Module Layer inside the CE contains independent software modules for ForCES applications. Typical application modules include routing protocol modules, network management module, modules for QoS, and modules for other services. Routing

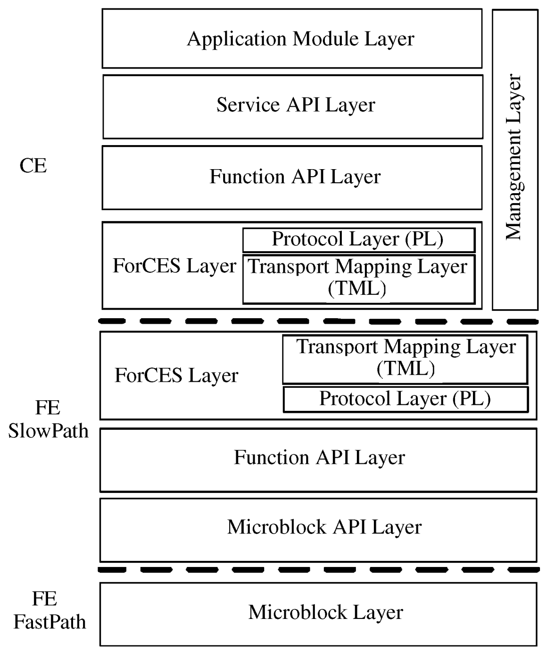


Fig.2. Layered model of ForTER.

protocol module includes routing protocol stack software like RIP, OSPF, etc. Network management module provides core software for network management protocols. The modular design makes it able to widely use third-party software modules as the ForCES application modules. In the ForTER implementation, we adopt Zebra routing module as the routing protocol module, and AgentX++ and NetSNMP as the SNMP module.

Service API Layer in CE is a bridge between upper softwares and ForCES-specific layers. On the one hand, Service APIs in CE convert a service (e.g., DiffServ) from upper application softwares to a sequence of operation on LFBs. On the other hand, Service APIs in CE provide call back functions, which are interfaces to third-party network software, for underlying function APIs in CE. In this way, Service API Layer has hidden the feature of ForCES for upper application and third-party network software. Service API Layer provides sets of function calls and data structures for various upper softwares, e.g., sets of function calls for SNMP, routing, DiffServ, etc. Service API Layer is implemented based on APIs in underlying Function API Layer.

Function API Layer in CE contains function calls and data structures for direct operation (i.e., configuration, inquiry, and event report) on every attribute and capability of all the LFBs. On the one hand, function APIs in CE will call functions in ForCES Layer to realize actual operation on LFBs in FEs. On the other hand, functions in Function API Layer of CE will be called by ForCES Layer when received protocol messages from FE are executed.

The CE ForCES Layer includes PL (Protocol Layer) and TML (Transport Mapping Layer). PL provides generation and execution of ForCES protocol messages for certain operations on LFBs in FEs. TML supports the transportation of ForCES messages over various transport media, like Ethernet, IP, ATM, etc. In most cases, the TML is over an IP-based media.

A CE Management Layer is set for the purpose of the whole CE management. This layer traverses all CE functional layers, providing a management interface for all the layers. In ForTER, we set a GUI interface for CE management purpose, which provides friendly interface for management of the whole ForCES Network element.

In FE, the FE ForCES Layer is identical to the ForCES layer in CE, which also consists of PL and TML sub layers for ForCES protocol.

According to ForCES FE Model[19], LFBs in FE are logical functional blocks that are abstracted and synthesized from FE physical resources. In the layered model of ForTER as in Fig.2, the physical resources in FE are described by microblocks. A microblock is a physical component with a single function, which usually runs at line speed to process packets, hence belongs to FE fastpath layer. To summarize, microblocks are described more from the implementation and physical point of view and not directly related to the ForCES structure, while LFBs in ForCES are modeled more from the logically abstract point of view. LFBs in FE are constructed on the basis of the microblocks.

Microblock API layer provides management APIs to underlying microblocks. The layer will exploit as much as possible programmability from the microblocks. The programmability of microblocks is an essence for programmability of whole FE.

A Function API Layer is necessary for connecting ForCES Layer and microblock API Layer in FE. The layer implements the converting task between LFB operations and microblock operations.

The layered model in Fig.2 is featured as being layered and modular, which helps exploit the generic features of ForCES as being open, programmable, modular, and standardized.

3 Design and Implementation

ForTER is implemented based on the layered model shown in Fig.2. We further present the design and implementation of elements of ForTER. We also present a security consideration regarding DoS attacks and propose an algorithm to prevent the attacks.

3.1 Control Element (CE)

Fig.3 shows the implementation structure of CE in ForTER. The main platform of CE is based on Windows OS, while third-party network software (including routing protocol software Zebra and SNMP software NetSNMP) run on a separate Linux OS platform, which is called the Routing & SNMP server.

As described in the layered model, CE PL supports the generation, encapsulation, decapsulation, and processing of ForCES protocol messages. The messages and associated modules include: 1) messages for association, e.g., AssociationSetup message, AssociationSetupResponse message, AssociationTeardown message; 2) messages for configure and query, e.g., Config message, ConfigResponse message, Query and QueryResponse message; 3) event notification message; 4) heartbeat message, 5) redirect message. Among the messages above, 1) to 4) messages are called ForCES control messages. Redirect message is the message that contains data packets that are to or from outer networks via FEs but that need to be processed by the CE, including packets for routing protocols and SNMP protocol. ForCES just encapsulates these packets directly in the redirect messages and transport them between CE and FEs. Different sub-modules in PL are used for processing different ForCES messages, like the sub-module for association, and the sub-module for redirection, as shown in Fig.3.

We use the TCP+UDP based TML, which is based on a proposed ForCES TML specification[29] that we have submitted to IETF, as the TML scheme for ForTER. In this scheme, TCP is applied to the transmission of ForCES control messages and UDP to redirect message. According to ForCES requirements in RFC 3654, the transmission of control messages should be reliable and congestion-controllable, while redirect messages do not need to be reliably transmitted. We will discuss the TML security problem in Subsection 3.3.

CE Management Layer as described in Fig.2 is represented by Management GUI (Graphical User Interface), as shown in Fig.3. The GUI is provided in ForTER for management of every part in CE. In the GUI, we designed an FE description tree and an LFB topology diagram for very handy operation of FEs and the LFBs inside. An FE consists of multiple LFBs, and an LFB contains its attributes and capabilities. We use XML to describe LFBs and use a tree-like structure to show them.

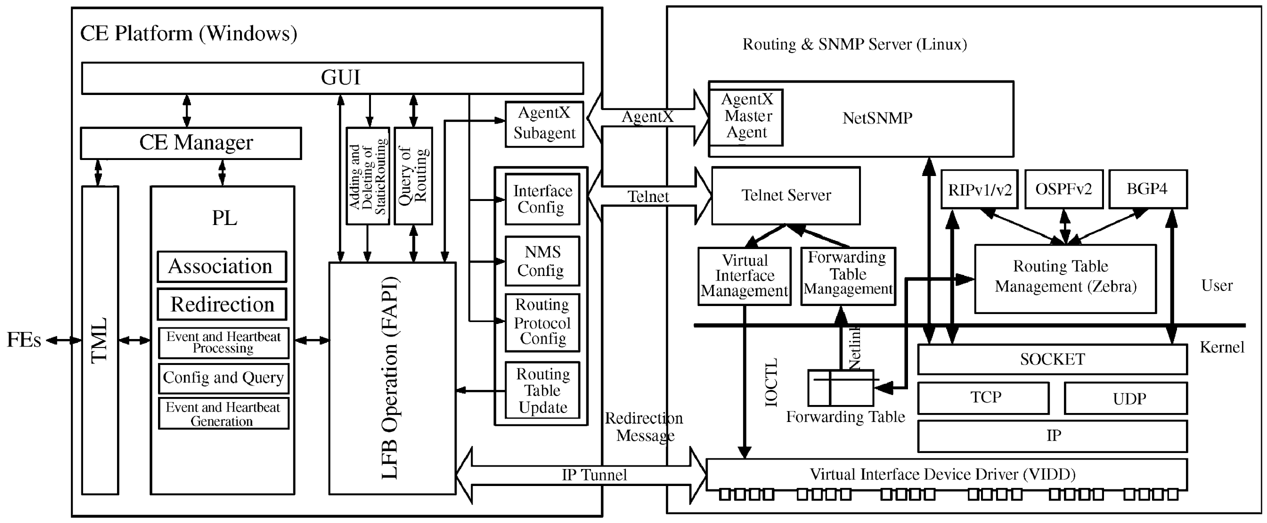
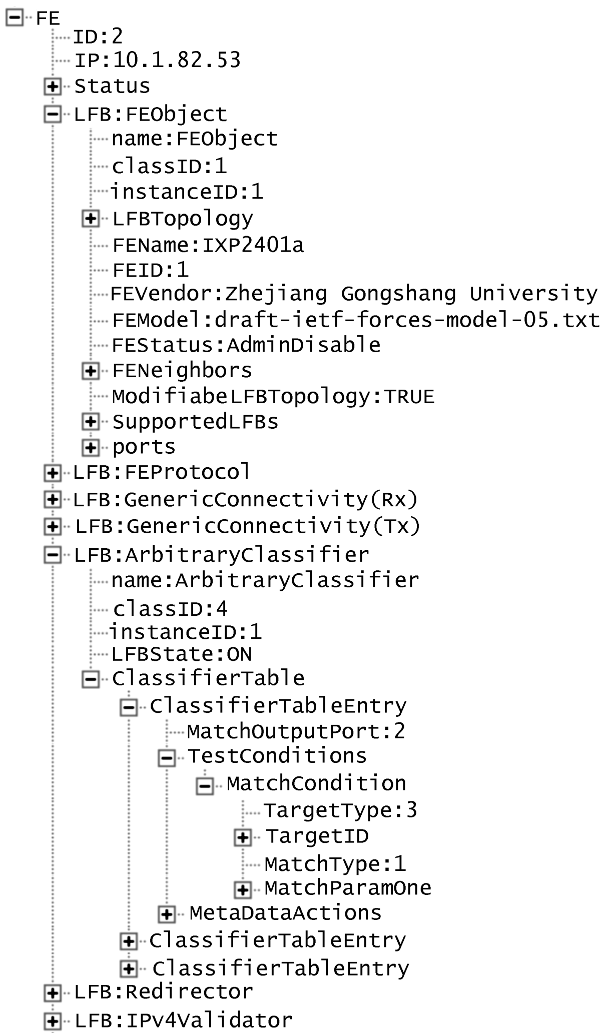


Fig.3. Structure of control element.

Fig.4 shows an instance for such an FE and LFB description tree in the GUI, where the FE attributes like ID, IP address, and associated LFBs like FE Object LFB and FE Protocol LFB[18] are shown. LFBs in an FE are interconnected and form an LFB topology. We represent the LFB topology in the CE GUI with an LFB topology diagram. This diagram looks like a circuit diagram, where an LFB is like an electronic element. Fig.5 shows an example of the LFB topology diagram, where every node represents an LFB in the FE and the lines represent the datapaths connecting them. By the diagram, we can clearly see the LFBs and their interconnection relationship inside the FE, hence provide the ability to see the functions and services of a router, which is very hard to do for traditional routers. Moreover, via ForCES configuration messages, CE is also able to dynamically add or remove some LFBs in the diagram, in order to modify the functions or services that FE provides. This makes FE quite open at its resources level and very flexibly programmable at the functional level. For example, by configuring and modifying the LFBs and the topology, we can change the ForTER from IPv4 routers to IPv4/IPv6 double stack routers in a very handy way. Management GUI also provides the operation on Service API Layer, e.g., start/ stop/ restart of routing, SNMP, and QoS service, parameter configuration of interfaces to routing software, SNMP software, etc. We also provide dialog boxes for TML management like TML parameter (e.g., transport protocols, service ports) configurations. This actually provides the functions as a CE manager does.

As described before, ForTER is flexible to include third-party software modules as the CE application module. In the implementation, Zebra is taken as the



```
⊟ FE
   ID:2
   IP:10.1.82.53
   ⊞ Status
   ⊟ LFB:FEObject
      name:FEObject
      classID:1
      instanceID:1
      ⊞ LFBTopology
      FEName:IXP2401a
      FEID:1
      FEVendor:Zhejiang Gongshang University
      FEModel:draft-ietf-forces-model-05.txt
      FEStatus:AdminDisable
      ⊞ FENeighbors
      ModifiabeLFBTopology:TRUE
      ⊞ SupportedLFBs
      ⊞ ports
   ⊞ LFB:FEProtocol
   ⊞ LFB:GenericConnectivity(Rx)
   ⊞ LFB:GenericConnectivity(Tx)
   ⊟ LFB:ArbitraryClassifier
      name:ArbitraryClassifier
      classID:4
      instanceID:1
      LFBState:ON
      ⊟ ClassifierTable
         ⊟ ClassifierTableEntry
            MatchOutputPort:2
            ⊟ TestConditions
               ⊟ MatchCondition
                  TargetType:3
                  ⊞ TargetID
                  MatchType:1
                  ⊞ MatchParamOne
            ⊞ MetaDataActions
         ⊞ ClassifierTableEntry
         ⊞ ClassifierTableEntry
   ⊞ LFB:Redirector
   ⊞ LFB:IPv4Validator
```

Fig.4. Example of an FE and LFB description tree in CE GUI.

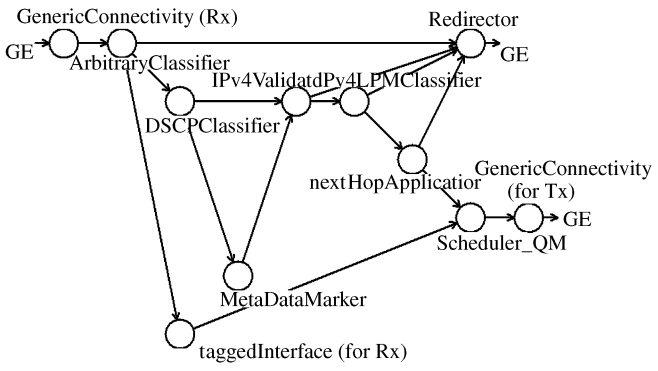


Fig.5. Example of LFB topology diagram in CE GUI.

routing protocol software module, and NetSNMP as the SNMP module. As mentioned before, routing protocol packets and SNMP packets are encapsulated in ForCES redirect messages so as to be transported between FE and CE. The redirect messages are further transported between CE main platform and Routing & SNMP server by use of IP tunnels, as shown in Fig.3.

A VIDD (Virtual Interface Device Driver) is used in the Routing & SNMP server to mirror FE real network interfaces (described in Fig.1 as Fi/f) to the server, in which the routing protocol and SNMP protocol softwares run. It makes all FE network interfaces just like network interfaces locally on this server. Redirect messages that contain routing protocol or SNMP protocol messages are transported over IP tunnel to the VIDD, and the VIDD recovers them back to routing or SNMP protocol messages and outputs them to routing protocol or SNMP software modules for use. In this way, the third-party software modules can just treat VIDD as network interfaces that they interact.

3.2 Forwarding Element

ForTER implements FEs based on Intel IXP network processors. In this case, FE SlowPath and FE FastPath in Fig.2 layered model are exactly referred to Intel XScale Layer and MicroEngine (ME) Layer. Structure of FE in ForTER is shown in Fig.6.

The FE Management module in Fig.6 is a manual interface to provide some necessary manual management for modules in the FE. The FE Management is based on CLI (Command Line Interface).

Upon receiving control messages from CE, the FE may be triggered to call LFB operations in the Functions API Layer. The operations on LFBs are further converted to the operations on microblocks, which are carried out by the Microblock API Layer.

We have built Microblock API layer and Microblock layer in ForTER based on Intel IXP environment, where Microblock API Layer matches Core Component (CC) Layer, Core Component Infrastructure Layer, and Resource Manager Layer, while Microblock layer to IXP ME layer.

A set of typical LFBs have been accordingly constructed in ForTER. The LFBs include IPv4 LPM forwarding LFB, DSCP classifier LFB, Tagged Interface LFB, scheduler LFB, etc.

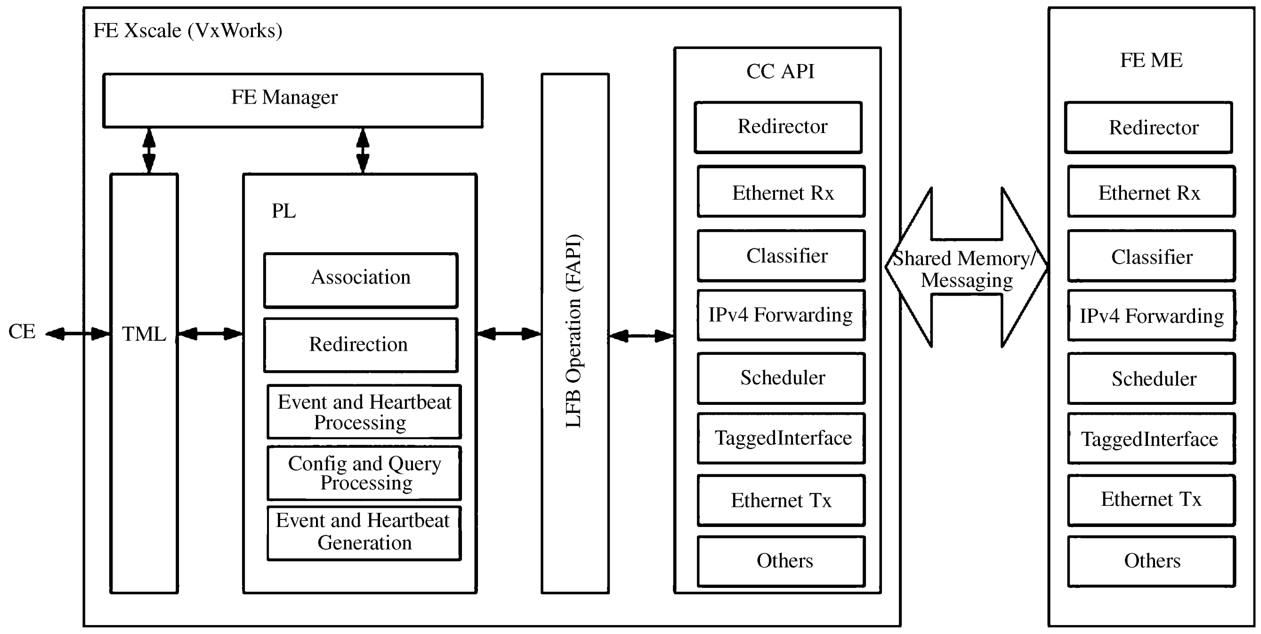


Fig.6. Structure of forwarding element.

3.3 Security Consideration

Because of the separation of control and forward elements, we must consider a special security problem for a ForCES router. As described before, FE may receive some packets from outer networks that have to be delivered to CE for processing. In this case, FE just forwards the packets to the CE over the ForCES channel and via ForCES redirect messages. This leaves a hole for malicious attackers. Attackers may try to send a huge amount of such packets to FE and make FE forward them to CE. As a result, bandwidth of the ForCES channel that connects FE and CE is easy to exhaust, making the CE loses control on FEs, and subsequently making the ForCES router totally down. We call such an attack as a DoS attack from redirect data.

In Subsection 3.1, we proposed adopting the TCP + UDP TML for ForTER. In the TML, TCP is for PL control messages and UDP for redirect messages. As is known, UDP is more aggressive than TCP, which leads to unfair bandwidth allocation and TCP congestion when TCP and UDP are in the same link. In the ForCES case, this will leave space for DoS attacks from redirect data.

We propose an algorithm here to avoid the DoS attack from redirect data. In the algorithm, we try to suppress UDP stream in some way that the UDP will not completely block TCP control message transmission. Note that in the algorithm, we do not try and do not need to setup a very strict and full congestion control for the UDP stream, and the main purpose of the algorithm is only to guarantee a minimum bandwidth for control message transmissions between CE and FE.

The key idea of the algorithm is to try to control UDP stream with reference to TCP stream regarding its congestion status, i.e., to roughly bind UDP to TCP for the congestion control. We call TCP and UDP, binding each other in this way, a TCP-UDP pair. In this way, we expect the UDP stream not going into a status that the UDP stream completely blocks the TCP stream.

Fig.7 shows the diagram for the algorithm. Two parallel threads are established in the ForCES TML to send ForCES messages from FE to CE, one for control message sending and another for redirect message sending. Control message sending thread uses TCP as the transport protocol, and redirect message sending thread uses UDP. A sending buffer is respectively set in the threads to temporarily store sending messages. A flag called ControlMsgSent flag is used as a global variable for these two threads, indicating the state for control message sending. In the control message sending thread, whenever the sending buffer is checked

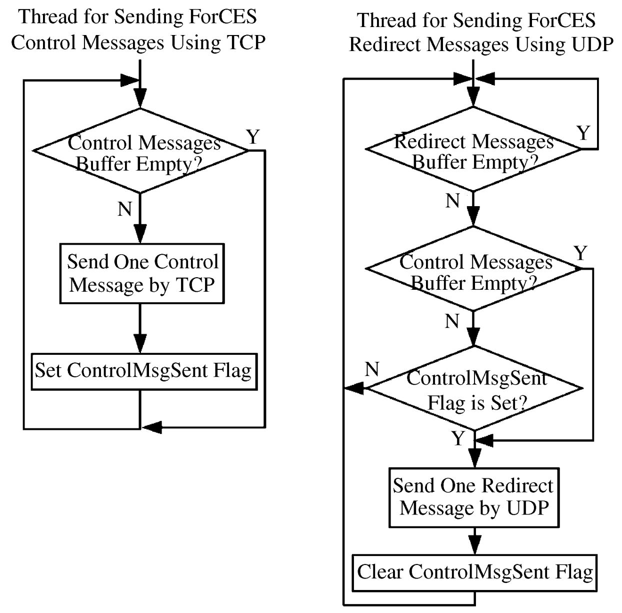

Thread for Sending ForCES Control Messages Using TCP
Control Messages Buffer Empty?
Y
N
Send One Control Message by TCP
Set ControlMsgSent Flag
Thread for Sending ForCES Redirect Messages Using UDP
Redirect Messages Buffer Empty?
Y
N
Control Messages Buffer Empty?
Y
N
N
ControlMsgSent Flag is Set?
Y
Send One Redirect Message by UDP
Clear ControlMsgSent Flag

Fig.7. Diagram of algorithm to protect redirect data DoS attack.

unempty, a control message is sent using TCP, and the ControlMsgSent flag is set. In the redirect message sending thread, the local redirect message buffer is first checked to see if there is any redirect message to send. If so, the sending buffer in the control message thread is further checked to see if, at the moment, there are control messages waiting for sending. If no, the redirect message sending thread just picks one redirect message and sends it by UDP. If yes, the thread must further check the ControlMsgSent flag state to decide if it is allowed to send a redirect message right now. When the flag is set, it actually means at least one control message has been sent before last time the redirect message was sent. In this case, the thread is allowed to send one redirect message over UDP at the moment. Otherwise, when the ControlMsgSent flag is not set, it actually means no control message has been sent since the thread sent one redirect message last time. In this case, the thread is not allowed to send any redirect message over the UDP channel and has to wait for the control message sending thread to send at least one control message. Whenever a redirect message is sent, the ControlMgSent flag is cleared so as to block the redirect message sending, till the flag has been set again by the control message sending thread.

4 Experiments

4.1 Purposes

The nature of the ForTER implementation is for the evaluation of the IETF ForCES architecture and the associated protocol, so we specifically list the purposes of the experiments we have made based on ForTER as follows.

1) To see the feasibility of the ForCES architecture as defined in RFC 3654 and RFC 3746.

2) To see the feasibility of ForCES protocol as defined in ForCES Protocol Specification[18].

We go with our experiments for above purposes by means of constructing several basic applications on CE. The applications include:

1) routing function and associated routing protocols, which are fundamental for routers;

2) SNMP network management.

Besides the purposes above, we also made an experiment on the algorithm presented in this paper for the prevention of DoS attacks from redirect data.

The ForTER used for the experiments consists of one CE and three FEs. CE includes a CE platform on Windows and a routing and SNMP server on Linux, structured as Fig.3. FE is structured as in Fig.6. Among the three FEs, two FEs run on Intel IXMB2401, and one FE on IXMB2851, which contain IXP2400 and IXP2850 Intel Network Processors, respectively. CE and FEs are interconnected by the ForCES protocol which runs over TCP+UDP TML. ZX5000 — an ATCA-based switch board is used as the switch fabric to connect these three FEs.

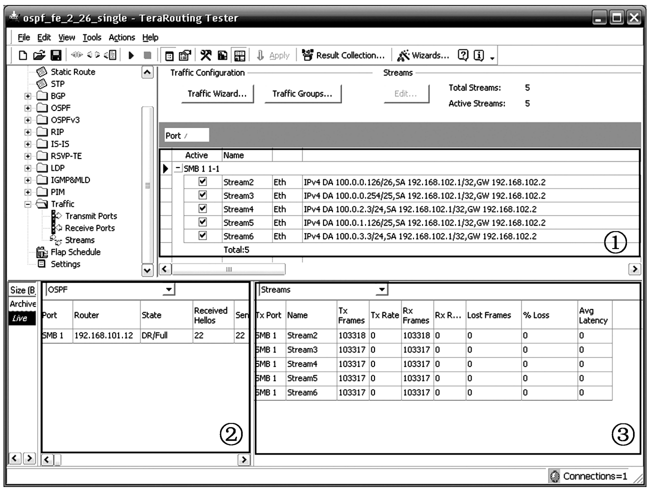
A SmartBits600 and the TeraRouting Tester software are used as the test flow generator and the measure tool.

The following subsections explain the individual experiments and show the results.

4.2 Routing and Running Routing Protocols

As described before, routing protocols run on the routing server. Via the ForCES protocol, the routing protocol software cooperatively works with resources on FEs to make ForTER a router. In this way, we have implemented RIP and OSPF routing protocols on the ForTER.

A TeraRouting Tester based on SmartBits600 with a LAN3321A port module is used for testing ForTER for its running of RIP and OSPF. TeraRouting Tester establishes a simulated network topology, which contains many simulated routers to send the messages of routing protocol via SmartBits600 ports. ForTER exchanges the routing protocol messages with the SmartBits600. The CE of ForTER learns the topology of the simulated network and generates the routing table. Then the routing table is distributed to all FEs of ForTER. In the test scenario, another interface of SmartBits600 sends IP traffic, whose destination IP address is in the range of the simulated network to ForTER. According to the generated routing table, FEs of ForTER forward the IP traffic to its output port which is connected back to another interface of SmartBits600. If ForTER correctly runs the routing protocols, SmartBits600 then can receive all IP traffic which is sent by itself. Moreover, through TeraRouting Tester, lost IP packets along the paths can be shown. In such a ForCES-based architecture, it is expected the TeraRouting Tester should show the returned traffic with no lost packets.



Total Streams: 5
Active Streams: 5

	Active	Name		
SMB 1 1-1				
	✓	Stream2	Eth	IPv4 DA 100.0.0.126/26,SA 192.168.102.1/32,GW 192.168.102.2
	✓	Stream3	Eth	IPv4 DA 100.0.0.254/25,SA 192.168.102.1/32,GW 192.168.102.2
	✓	Stream4	Eth	IPv4 DA 100.0.2.3/24,SA 192.168.102.1/32,GW 192.168.102.2
	✓	Stream5	Eth	IPv4 DA 100.0.1.126/25,SA 192.168.102.1/32,GW 192.168.102.2
	✓	Stream6	Eth	IPv4 DA 100.0.3.3/24,SA 192.168.102.1/32,GW 192.168.102.2
		Total:5		

Port	Router	State	Received Hellos	Sen
SMB 1	192.168.101.12	DR/Full	22	22

Tx Port	Name	Tx Frames	Tx Rate	Rx Frames	Rx R...	Lost Frames	% Loss	Avg Latency
SMB 1	Stream2	103318	0	103318	0	0	0	0
SMB 1	Stream3	103317	0	103317	0	0	0	0
SMB 1	Stream4	103317	0	103317	0	0	0	0
SMB 1	Stream5	103317	0	103317	0	0	0	0
SMB 1	Stream6	103317	0	103317	0	0	0	0

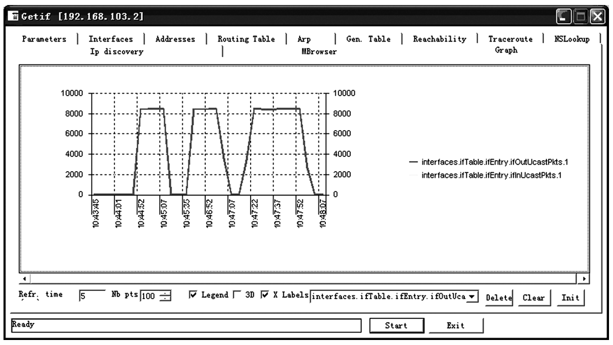
Fig.8. Result of routing and routing protocol experiment.

Fig.8 shows the final results for testing the routing protocol and the routing ability of the ForTER. Area ① of Fig.8 shows information of all IP traffic streams that are generated. Area ② selects the tested routing protocol type (OSPF as in the experiment), and shows the information associated with the routing protocol. At last, area ③ shows the state of IP traffic streams that are making round trips between SmartBits600 and ForTER, in which the lost frames and the % loss show the loss state for the streams. Our experiment has shown no packet loss here. This can only be achieved when the ForTER has run the routing protocol well, and has correctly produced routing tables and forwarded packets according to the tables in the FEs. ForCES protocol is the basis for the CE and FEs to interactively run the whole routing protocol. The ForCES protocol messages like configure message, query message, association message, and redirect message are frequently used in the process. As a result, this routing ability and routing protocol experiment has also well evaluated the feasibility of ForCES protocol and ForCES architecture for routers.

4.3 SNMP-Based Network Management

As Fig.3 explains, SNMP agent runs on the SNMP server. The server collects information on CE and FEs to form MIB information. Information on FEs is actually collected by means of the ForCES protocol.

The experiment is to show the ability of ForTER to support SNMP. It has been tested that the SNMP agent in ForTER can configure and query the information on the FEs to correctly form various MIB information, and it can also form SNMP trap messages to report events that happened on FEs. We use Getif software module as an SNMP management browser, which is located in the outer network of ForTER. Fig.9 is the experimental result on the Getif that shows the IP traffic of a specific port in an FE of the ForTER. This is done by means of SNMP to GET the ifOutUcastPkts and ifinUcastPkts MIB on the FE. We still use SmartBits600 as the traffic generator of the port, and use SmartBits600 start/stop button to switch the traffic. Then, we monitor the traffic got by the Getif. Fig.9 shows a snapshot of the traffic diagram, where ifOutUcastPkts and ifinUcastPkts are recorded.



Refr. time 5 Nb pts 100 ☑ Legend ☐ 3D ☑ X Labels interfaces.ifTable.ifEntry.ifOutUca

Ready

Fig.9. Result of testing SNMP.

The experiment shows that SNMP is well supported in the implementation of ForTER. It also shows the feasibility of ForCES architecture and ForCES protocol from another point of view, as ForTER has ForCES-based architecture.

4.4 DoS Attack Prevention Algorithm

The algorithm to prevent the redirect data DoS attack, which is to bind UDP to TCP regarding its congestion control, as proposed in Fig.7, is tested, and its feasibility of using a platform is shown in Fig.10. There is only one CE and one FE simulated. Connecting CE and FE is a 10Mbps Ethernet link.

The send rate of packets from SmartBits600 can be easily configured, and the receive rate of packets can be accurately counted in SmartBits600. Also, the send rate and receive rate of packets can be detailedly compared in SmartBits600. Therefore SmartBits600 is used for generating two rate-adjustable UDP flows (called Flow I and Flow II) that are input to FE. Once FE receives *one* packet of UDP Flow I from SmartBits600, FE would send *one* simulated ForCES control message using the TCP socket. When FE receives *one* packet of UDP Flow II from SmartBits600, FE would send *one* simulated ForCES redirect message using the UDP socket. In this way, the UDP Flow I triggers ForCES control message flows that have to be sent from FE to CE via a TCP TML, and the UDP Flow II triggers ForCES redirect messages that also have to be delivered to CE from FE via the UDP TML. For simplicity, simulated ForCES control messages have the same size as simulated ForCES redirect messages. We have tried different-sized (ranging from 64 bytes to 1500 bytes) ForCES messages. The experimental results are very similar.

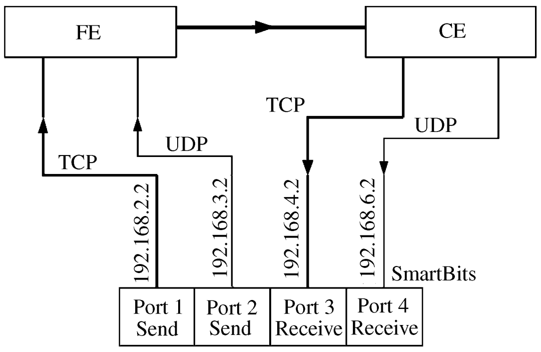


Fig.10. Dos attack testing platform.

By changing rates from SmartBits600, we can then adjust the control and redirect messages rates respectively. At the CE side, it is designed to recover two UDP flows again when CE has received the flows, and then returns the flows back to SmartBits600. In this way, in SmartBits600, we then can monitor how the TCP flow or the UDP flow has been transported from FE to CE, or in other words, how the throughput is over the CE-FE link for the two flows, to see if some of them are blocked or not.

We test the feasibility of the DoS attack algorithm by means of comparing the throughput status of the flows when the algorithm is applied and not. TCP sending traffic keeps 5Mbps, and UDP sending traffic changes from 1Mbps to 20Mbps for simulating the DoS attack from redirect data. The throughput is represented by a percentage, indicating the relative throughput.

Fig.11 shows the throughput status when no DoS attack prevention means is applied. In this case, the

UDP flow is independent of the TCP flow regarding the congestion. Fig.11 shows the throughput changes of the two flows following the UDP traffic changes. A little more than 1.0 of the relative throughput is from the system rounding error. From the figure, we can clearly see the TCP throughput decreases along with the UDP traffic increases. This means, in the ForCES case, the ForCES control messages become harder and harder to deliver to CE when ForCES redirect message traffic floods.

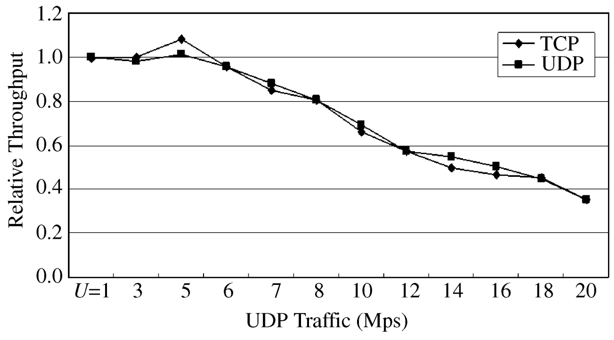


Fig.11. UDP independent of TCP.

Fig.12 shows the different result at the time when the redirect DoS attack prevention algorithm applies. As described, the algorithm binds UDP traffic to TCP traffic. Fig.12 shows that when the UDP traffic increases from 1M to 20M, the TCP relative throughput can roughly keep at 100% status, meaning the ForCES control messages are not blocked. This also means attackers are unable to destroy the communications between CE and FE.

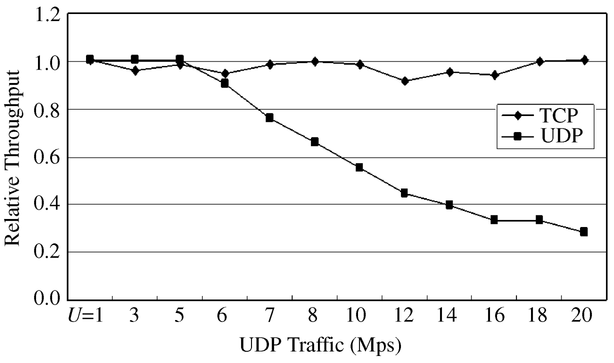


Fig.12. UDP binding to TCP.

5 Conclusion

In this paper, we introduce the design and implementation of the ForTER, an IETF ForCES-based router. The architecture and the layered software model for ForTER are described. The detailed design schemes for the Control Element and the Forwarding Element are presented, and an algorithm to prevent DoS attacks from ForCES redirect data is proposed.

Experiments have shown the feasibility of ForTER with its designs regarding the ability to develop routing and to run routing protocol, and the ability to support SNMP network management. More importantly, the experiments have, as a result, actually illustrated the feasibility of IETF ForCES architecture as defined by RFC 3654 and RFC 3746, and the IETF ForCES protocol, which is near to be standardized as an RFC protocol.

IETF ForCES working group is now working hard and is moving towards its completion. Implementations based on ForCES are highly expected from vendors as well as the ForCES working group to make one important step forward towards completion of ForCES specifications, and ForCES applications. As the only one full ForCES implementation ever presented till now, it is expected ForTER implementation introduced in this paper can contribute to IETF ForCES work for its progress.

More efforts on ForTER are required to improve ForTER from an experimental router into a practical router. For example, we shall carry out the DoS prevention algorithm experiments in environment of multiple FEs and observe whether different FEs will affect each other. Also, our TCP + UDP TML should be compared with other TML schemes (e.g., SCTP or DCCP).

Acknowledgments The authors are grateful for contributions from the following colleagues: Ming Gao, Feng-Gen Jia, Rong Jin, Jin Yu, Xiao-Chun Wu and Chuan-Huang Li. The authors also thank the reviewer for comments which were very helpful to improve the quality of this paper.

References

[1] Struck B, Weingarten M. Next Gen Switch/Router design issues: Tomorrow's LANs, SANs and WANs will need more coprocessors, plus bigger buffers and programmable-on-the-fly, non-blocking switch/routers. *Business Communication Review,* 2004, 34(10): 44–50.

[2] Jian-Ping Wu, Ke Xu. Research on next-generation Internet architecture. *Journal of Computer Science and Technology,* 2006, 21(5): 723–731.

[3] Guan-Qun Gu, Jun-Zhou Luo. Some issues on computer networks: Architecture and key technologies. *Journal of Computer Science and Technology,* 2006, 21(5): 708–722.

[4] Wang W, Dong L. Design and performance evaluation of general router management protocol (GRMP). In *Proc. IEEE TENCON*, Chiang Mai, Thailand, Nov. 2004, pp.37–40.

[5] Calvert K L, Campbell A T, Lazar A A, Wetherall D, Yavatkar R. Active and programmable networks. *IEEE Journal*

on Selected Areas in Communications, 2001, 19(3): 401–403.

[6] OPENSIG. http://comet.ctr.columbia.edu/opensig/.

[7] El-Darieby M, Rolia J, Petriu D C. Performance modeling for virtual network-based service provisioning. In *Proc. IFIP/IEEE International Symposium on Integrated Network Management,* Seattle, USA, May 2001, pp.139–152.

[8] Biswas J *et al.* The IEEE P1520 standards initiative for programmable network interfaces. *IEEE Communications Magazine*, 1998, 36(10): 64–72.

[9] Multi-protocol over ATM version 1.0. The ATM Forum Technical Committee. *AF-MPOA-0087.000*, July 1997.

[10] General switch management protocol. http://www.networksorcery.com/enp/protocol/gsmp.htm.

[11] Newman P, Minshall G, Lyon T, Huston L. IP switching and gigabit routers. *IEEE Communications Magazine*, 1997, 35(1): 64–69.

[12] Multiservice switching forum. http://www.msforum.org/.

[13] Forwarding and control element separation (forces). http://www.ietf.org/html.charters/forces-charter.html.

[14] Network processing forum (NPF). http://www.npforum.org/.

[15] Woo T, Faynberg I. The softrouter concept and the proposal for the study of separation of forwarding and routing in the next generation network. http://www.itu.int/ITU-T/ngn/fgngn/docs/FGNGN-004-June04.doc.

[16] SoftRouter. http://www.bell-labs.com/org/11347A/projects.htm#SoftRouter.

[17] Naik U *et al.* IXA portability framework: Preerving software investment in network processor applications. *Intel Technology Journal*, 2002, 6(3): 50–60.

[18] Doria A, Wang W, Dong L *et al.* ForCES protocol specification. Internet-Draft, work in progress, http://www.tools.ietf.org/html/draft-ietf-forces-protocol/.

[19] Halpern J *et al.* ForCES forwarding element model. Internet-Draft, work in progress, http://www.tools.ietf.org/html/draft-ietf-forces-model/.

[20] Haas R, Suzuki T. Architecture of the flexinet forCES-based control point. In *Presentation in 63rd IETF Meeting*, Paris, France, Aug. 2005, http://www.ietf.org/proceedings/05aug/slides/forces-1/forces-1.ppt.

[21] Haleplidis E, Haas R, Denazis S, Koufopavlou O. A web service- and ForCES-based programmable router architecture. In *Presentation in International Working Conference on Active and Programmable Netwoks* (*IWAN*), Sophia Antipolis, France, Nov. 2005, http://www.ens-lyon.fr/LIP/RESO/iwan2005/slides/IWAN05-Haleplidis.ppt.

[22] Chrysoulas C *et al.* A distributed router's modeling and implementation. ESRGroups Research Report, Dec. 2006.

[23] Hidell M, Hagsand O, Sjodin P. Distributed control for decentralized modular routers. In *Proc. SNCNW,* Karlstad, Sweden, Nov. 2004, pp.9–13.

[24] Hidell M *et al.* Control and forwarding plane interaction in distributed routers. In *Proc. 4th International IFIP-TC6 Networking Conference*, Waterloo, Canada, May 2005, pp.1339–1342.

[25] Fu J, Hagsand O. A programming model for a forwarding element. In *Proc. SNCNW,* Karlstad, Sweden, Nov. 2004, http://www.ee.kth.se/php/modules/publications/reports/2004/IR-EE-LCN_2004_002.pdf.

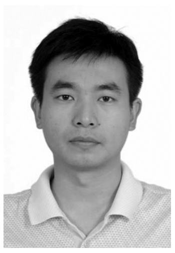
[26] Fu J, Hagsand O. Designing and evaluating network processor applications. In *Proc. IEEE Workshop on High Performance Switching and Routing,* Hong-Kong, May 2005, pp.142–146.

[27] Wang W *et al.* Design and implementation of an open programmable router compliant to IETF ForCES specifications. In *Proc. ICN*, Martinique, France, April 2007, p.82.

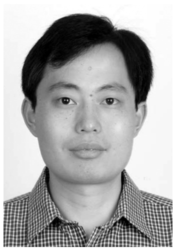
[28] Lakkavali S, Khosravi H. ForCES protocol design analysis for protection against DoS attacks. In *Proc. ICCCN,* Oct. 2004, pp.550–554.

[29] Wang W, Dong L, Zhuge B. TCP and UDP based ForCES protocol TML over IP networks. Internet-Draft, work in progress, http://www.tools.ietf.org/html/draft-wang-forces-iptml/.

Wei-Ming Wang is a professor in the College of Information and Electronic Engineering at Zhejiang Gongshang University. He obtained his Ph.D. degree in electrical engineering from Zhejiang University, China, in 1992, M.A. degree in electrical engineering from National University of Defence Technology, China, in 1989, and B.E. degree in telecommunications from Zhengzhou Institute of Information Engineering, China, in 1983. He worked as a research engineer in the Internet Technology Lab, The University of Arizona, from March 2001 to May 2002. His active research area covers open-arch networks and its applications in Internet, specializing in IETF protocols on forwarding and control element separation (ForCES).



Li-Gang Dong received his B.Eng. and M.Eng. degrees both from Zhejiang University, China, in 1995 and 1998, respectively. In 2003, he obtained the Ph.D. degree from the Department of Electrical and Computer Engineering at the National University of Singapore, Singapore. He worked as a research fellow in Institute for Infocomm Research, Singapore, from July 2002 to July 2003. Currently he is an associate professor in the College of Information and Electronic Engineering of Zhejiang Gongshang University, China. His research interests include various topics in computer networks and distributed systems.



Bin Zhuge is an associate professor in the College of Information and Electronic Engineering at Zhejiang Gongshang University, Hangzhou, China. He received his B.Eng. degree in detection technique and instrument from University of Electronic Science and Technology of China in 1998. He obtained his Ph.D. degree in biomedical engineering from University of Science and Technology of China in 2003. His current research interests are in the areas of architectures of network equipments, network security, and routing issues.