Protocol Proxy for OSPF Emulation

Shunichi Tsunoda, Nattapong Kitsuwan, Eiji Oki, Takashi Miyamura, and Kohei Shiomoto

***Abstract*—This paper proposes a protocol proxy scheme that can emulate the open shortest path first (OSPF) protocol in an effective and flexible manner. It is implemented in our developed network emulator, which is used to test network controllers for IP optical network management. In the protocol scheme, OSPF protocol emulation is achieved by combining an OSPF protocol proxy (introduced here) and an OSPF peer state manager based on existing OSPF protocol software. The protocol proxy produces OSPF packets holding link state advertisements with customized extensions including MPLS and GMPLS, while the OSPF peer state manager implements neighbor establishment via the proxy. The protocol proxy has two main functions: rewrite OSPF packets originated by the OSPF peer state manager and generate OSPF packets to inform the updated topology to the network controller. To implement the customized OSPF extensions, only the protocol proxy software need be modified; the existing OSPF software for the OPSF peer state manager is not touched. This makes the implementation of the OSPF emulation easy and flexible. Furthermore, the protocol proxy obtains the network topology information from the resource simulator, which is managed in a centralized manner. This reduces the amount of processing resources required and is scalable in terms of network size. We develop a prototype of the network emulator including OSPF protocol emulation with the protocol proxy scheme. The effectiveness of the protocol proxy scheme is confirmed by an experiment on 40 nodes.**

***Index Terms*—OSPF, emulation, protocol proxy**

I. Introduction

Various types of applications are likely to appear and their traffic demands are difficult to predict. Applications that require large bandwidth and high quality, such as high-definition television, video streaming, and high-speed data transfer, place high demands on network resources. The future backbone network is required to support various service networks and must be implemented in a timely manner to satisfy customer demands. In addition, the network resources need to be efficiently utilized in a dynamic manner.

To achieve these requirements, ``network virtualization" was presented in [1],[2]. Network virtualization creates multiple service networks by overlaying IP networks on top of a single optical network. An IP optical Traffic Engineering (TE) server, which is currently under development [1], determines the resource assignments of those service networks with some appropriate optimization algorithm [3]-[5]. The IP optical TE server has two main functions. First, it collects information of network topology, network resources, and traffic. Second, it controls elements such as network paths, links, and bandwidth according to the traffic demands, traffic characteristics, and quality-of-service requirements

To develop a truly practical IP optical TE server, the IP optical TE server's functions and performance must be confirmed for a large-scale network with more than one hundred nodes. To ensure a rigorous examination, the large-scale network must support various topologies and traffic characteristics. However, preparing an actual IP optical network of this scale is not feasible. In addition, it is difficult to generate various traffic streams in an experimental network. Accordingly, it is difficult to achieve these requirements in any experiment on an actual network. Though, it must be used to establish a scalable experimental environment to advance the development of IP optical TE server. A device that emulates an actual network's behavior is needed to confirm the validity and performance of the developed IP optical TE server. This device is called a network emulator.

Several available network simulators are available such as ns-2 [6], ns-3 [7] and OPNET [8]. These simulators provide simulated results to network designers based on the configured network and given traffic conditions, but they do not have sufficient router interfaces to permit the IP optical TE server to behave as in an actual network. The interfaces include CLI (command line interfaces) via telent, SNMP (Simple Network Management Protocol) [9], OSPF (Open Shortest Path First) [10], [11].

An architecture of a network emulator for IP optical network management was presented in [12], [13]. The network emulator mainly consists of three modules: a router interface module, a resource simulator, and a traffic generator. The router interface module supports several protocols, such as telnet, SNMP, OSPF. Each router interface in the network is implemented as a virtual node in the router interface module. The resource simulator module simulates the network resources based on requests of path setups and releases triggered by the IP optical TE server via the router interface module; it judges if the requests can be accepted. The traffic generator creates traffic information, which is retrieved from the IP optical TE server via SNMP, such as traffic volume passing through each link interface [14]. The

Manuscript received August 26, 2011. This work was supported in part by the National Institute of Information and Communications Technology, Japan and the Strategic Information and Communications R&D Promotion Program of the Ministry of Internal Affairs and Communications, Japan.

Shunichi Tsunoda, Nattapong Kitsuwan and Eiji Oki are with Department of Information and Communication Engineering, the University of Electro Communications, Tokyo Japan. (e-mail: {t0610143, kitsuwan, oki} @ice.uec.ac.jp)

Takashi Miyamura and Kouhei Shiomoto are with NTT Network Service Systems Laboratories, NTT Corporation, Tokyo Japan.

network emulator provides an experimental environment for IP optical network management [15], and allows a variety of network control actions to be examined under the various traffic characteristics expected of a large-scale network.

OSPF protocol emulation is a key function in the network emulator. This emulation must meet several requirements as follows. Customized OPSF extensions including MPLS/GMPLS ones [11], [16] should be supported for IP optical network management. Processing resources should be efficiently used to emulate large-scale IP optical networks. In addition, the protocol emulation software should be implemented in a flexible and timely manner. Unfortunately, implementation of an OSPF protocol emulation that can satisfy these requirements for the network emulator presented in [12] remains an open issue.

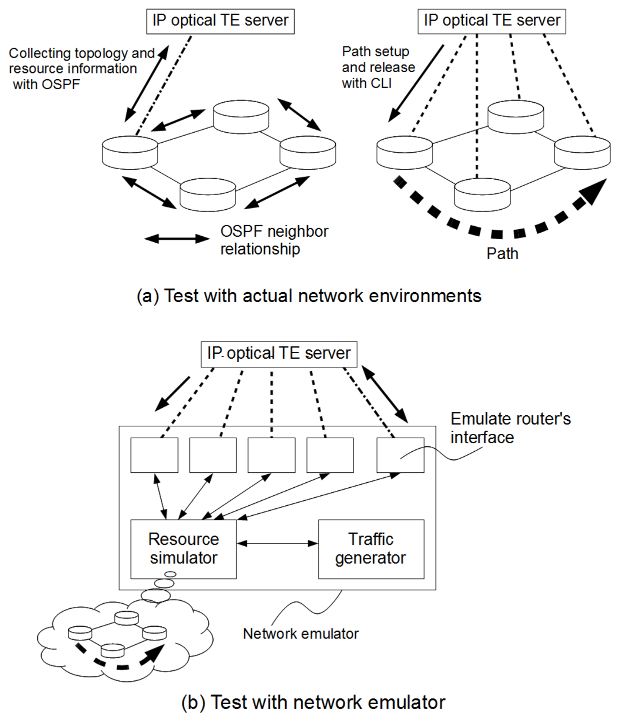
This paper proposes a protocol proxy scheme, which emulates the OSPF protocol in the network emulator and meets our objectives. In the protocol scheme, OSPF protocol emulation is achieved by combining an protocol proxy that we introduce here and an OSPF peer state manager based on existing OSPF protocol software. The protocol proxy produces OSPF packets of link state advertisements with customized extensions including MPLS and GMPLS, while the OSPF peer state manager realizes neighbor establishment via the proxy. The protocol proxy has two functions: rewrite OSPF packets originated by the OSPF peer state manager and generate OSPF packets to inform the updated topology to the IP optical TE server. Implementation of the OPSF customized extensions requires modification of only the protocol proxy software, the existing OSPF software for the OPSF peer state manager is no touched. This makes the implementation of the OSPF emulation easy and flexible. Furthermore, the protocol proxy obtains the network topology information from the resource simulator, which is managed in a centralized manner. This reduces the amount of processing resources required and is scalable in terms of network size. We develop a prototype of the network emulator that includes OSPF protocol emulation via the protocol proxy scheme. The effectiveness of the protocol proxy scheme is confirmed by an experiment on a network with 40 nodes.

The remainder of this paper is organized as follows. Section II describes the network emulator. Section III explains the requirements faced when implementing OSPF emulation. Section IV describes OSPF implementation based on existing schemes. Section V proposes OSPF implementation based on the protocol proxy scheme. Section VI introduces the implementation and the result gained. Section VII presents our conclusions.

II. NETWORK EMULATOR

A scalable network emulator architecture that supports the development of the IP optical server was presented in [12], [13]. The network emulator uses the same router interfaces to communicate with the IP optical TE server as the actual IP optical network, and behaves as an actual IP optical network between the interfaces. Moreover, the network emulator provides a variety of customizable traffic environments.

Figure 1(a) shows a test conducted on an actual network; its cost is excessive and it is difficult to generate realistic traffic due to its unpredictability. The IP optical TE server uses OSPF to collects the topology and resource information. Each OSPF-router exchanges information with neighboring OSPF routers. The IP optical TE server running OSPF has a neighbor relationship with at least one router in the network from which, gets the information necessary. Note that the IP optical TE server does not need to have an OSPF neighbor relationship with all routers in the network. Moreover, it setups or releases a path via CLI in response to a request. Figure 1(b) shows a test conducted on the network emulator, which behaves as the test with actual network.


IP optical TE server
Collecting topology and resource information with OSPF
IP optical TE server
Path setup and release with CLI
OSPF neighbor relationship
Path
(a) Test with actual network environments
IP optical TE server
Emulate router's interface
Resource simulator
Traffic generator
Network emulator
(b) Test with network emulator
Fig. 1. Comparison of test environments for IP optical server.

The network emulator mainly consists of three modules, which are a router interface module, a resource simulator module, and a traffic generator module, and several databases, as shown in Fig. 2. The databases are one TE database (DB) for each service network, one TE DB for the optical network, and the traffic DB. The functions of these modules for the network emulator are as follows.

The router interface module communicates with the IP optical TE server and replicates the behavior of one or more routers. This module consists of *N* sub-modules for Command Line Interface (CLI)/SNMP, where *N* is the number of routers in the emulated service/optical network [1], and one sub-module for OSPF. Sub-modules for CLI/SNMP, which are denoted as virtual nodes 1,…, *N*, are prepared for all routers, which correspond to the emulated network. These sub-modules communicate with the IP optical TE server, to update the router configurations including path setup and release information and to collect traffic information. Virtual node *R* uses OSPF to exchange topology information with the IP optical TE server. If the topology information is updated, the resource simulator notifies the update to virtual node *R*, which then sends it to the IP optical TE server. Since the IP optical TE server uses OSPF to get the topology information from virtual node *R*, the OSPF configuration from virtual nodes 1 to *N* is not required. Virtual node *R* behaves as a neighbor node in the actual network. Although each router runs OSPF in the actual network, it does not use OSPF to communicate with the IP optical TE server with OSPF, only its neighboring router. Therefore, in the network emulator, OSPF communication between the IP optical TE server and virtual node *R* is enough. Using virtual node *R* reduces the complexity of the resource simulator. If all virtual nodes speak OSPF, the resource simulator has to control all virtual nodes. If the network emulator must emulate a large-scale network, it would become to complex. It is more practical if the resource simulator controls only the virtual node *R* since *R* aggregates the OSPF information.

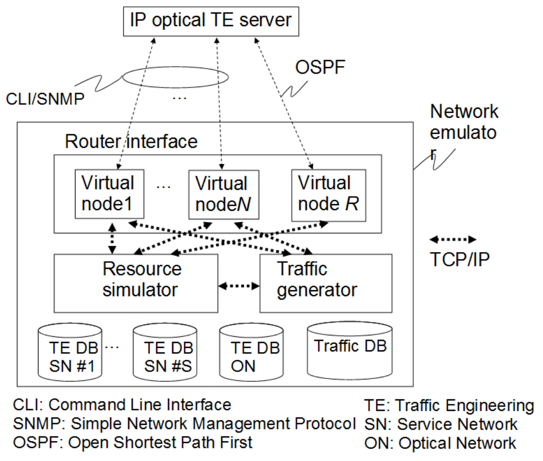


Fig. 2. Scalable network emulator architecture.

The resource simulator module emulates the statuses of the emulated network, such as path setup and release and resource management so as to ensure compliance with MPLS/GMPLS protocols [10], [11], [17]. When the IP optical TE server requests path setup via CLI to the router interface module, the resource simulator module judges whether the path setup request should be accepted or rejected according to the current available resources, the requested bandwidth, and the route. The TE databases are used by the resource simulator to keep the updated topology information.

The traffic generator module generates traffic information to reflect various traffic characteristics with consideration given to the unpredictability of traffic fluctuations [18]. The information so generated is used by the resource simulator and the IP optical TE server via SNMP. The generated traffic information is kept in the traffic DB, which is managed by the traffic generator.

Each virtual node communicates with the resource simulator module and traffic generator modules. In addition, the resource simulator module communicates with the traffic generator module. To make the network emulator scalable, these communications are performed via TCP/IP. Therefore, the proposed architecture allows modules/sub-modules to be apportioned among different computers, so network emulation can be performed in a distributed manner. This makes the network emulator scalable in terms of network size.

III. Requirements of OSPF Emulation in Network Emulator

The network emulator is mainly used to test the functioning of the IP optical TE server as an alternative to creating an actual network as the test environment. OSPF protocol emulation must inform the network topology and resource information to test the functioning of the IP optical TE server. OSPF protocol emulation is provided by virtual node R in the network emulator. Virtual node R obtains the updated topology information from the resource simulator and communicates with the IP optical TE server using OSPF. For OSPF protocol emulation, there are three main requirements. First, customized OPSF extensions, including MPLS/GMPLS extensions, should be easily supported. Second, processing resources should be efficiently used to allow the emulation of large-scale IP optical networks. Third, the protocol emulation software should be implemented in a flexible and timely manner.

The key OSPF functions are neighbor establishment with another OSPF peer and link state advertisement. Figure 3 shows the functions required for OSPF protocol emulation.

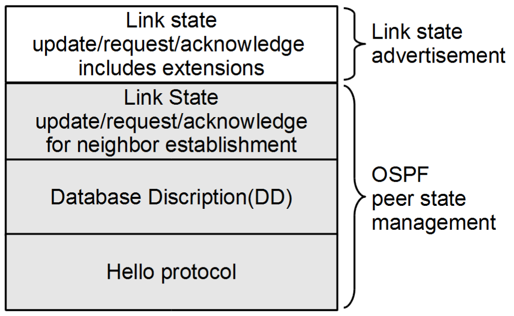

Link state update/request/acknowledge includes extensions
Link State update/request/acknowledge for neighbor establishment
Database Discription(DD)
Hello protocol
Link state advertisement
OSPF peer state management

Fig. 3. OSPF function model.

IV. Conventional Approaches

To emulate the OSPF protocol, there are two conventional approaches. One is to modify an existing OSPF emulator [19], which may be a commercial one, and the other is to modify existing OSPF software.

In the first approach, an existing commercial OSPF emulator, which has limited interfaces, supports various networks and some extensions with MPLS/GMPLS by setting pre-defined configurations. Figure 4 shows a schematic view of the first approach. Whenever the network status is changed, the configurations need to be updated. However, existing commercial OSPF emulators are not easily accept the addition of customized extensions that are not supported. In addition, as the interfaces of existing commercial OSPF emulators available for updating network configurations are limited, it is difficult to ensure integration with the other modules in our network emulator. In the network emulator, the resource simulator module requires virtual node *R* to access the configuration file via TCP/IP.

In the second approach, an OSPF peer that uses existing OSPF software, for example Quagga [20], is processed at each emulated router. Figure 5 shows a schematic view of the second approach. As processing resources are required in proportion to the number of nodes in the emulated network, it is not scalable in terms of network size. In addition, to adding customized OSPF extensions, we have to modify the existing OSPF protocol, which requires substantial development efforts including software debugging.

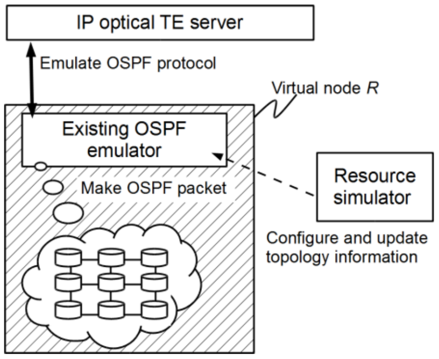


Fig. 4. Approach using OSPF emulator.

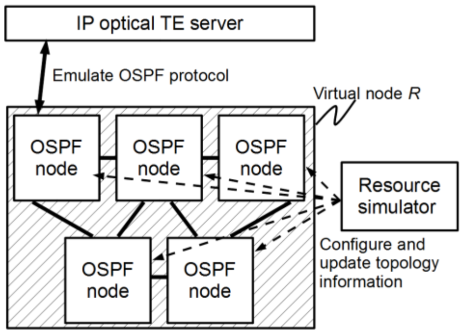


Fig. 5. Approach using existing OSPF software.

V. PROTOCOL PROXY SCHEME

We propose the protocol proxy scheme that meets our requirements. Figure 6 shows a schematic view of the protocol proxy scheme. This scheme achieves OSPF protocol emulation by combining the protocol proxy that we introduced and an OSPF peer state manager based on existing OSPF protocol software, which remains unmodified. The protocol proxy is inserted between the IP optical TE server and the OSPF peer state manager. The protocol proxy produces OSPF packets of link state advertisements using customized extensions including MPLS and GMPLS, while the OSPF peer state manager realizes neighbor establishment via the proxy.

The protocol proxy has two main functions. The first one is to rewrite OSPF packets originated by the OSPF peer, and the second one is to generate OSPF packets to inform the updated topology to the IP optical TE server. The topology information is configured and updated by the resource simulator via TCP/IP. To implement the OPSF customized extensions, only the protocol proxy software is modified, the existing OSPF software for the OPSF peer state manager remains unmodified. This makes the implementation of the OSPF emulation easy and flexible. Furthermore, the protocol proxy obtains the network topology information from the resource simulator module, which is managed in a centralized manner.

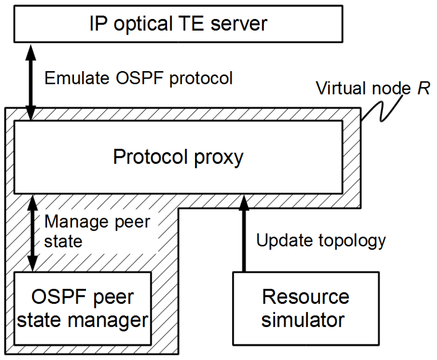


Fig. 6. Concept of protocol proxy scheme.

Figure 7 shows an example of the behaviors of the protocol proxy. The protocol proxy captures all OSPF packets that are exchanged between the OSPF peer state manager and the IP optical TE server, and relays the captured packets to the other side. The protocol proxy rewrites each captured packet before forwarding them to the destination.

First, the OSPF protocol exchanges hello packets between each router. At virtual node *R*, a hello packet is generated by the OSPF peer state manager. The protocol proxy relays the hello packet between the OSPF peer state manager and the IP optical TE server. After exchanging the hello packets, the OSPF protocol determines the adjacency relationships. A database description (DD) packet is used to establish each adjacency relationship, such as neighbor establishment. The DD packet has the neighbor information gained from already established router. A router exchanges the information with its neighbors via DD packets. The protocol proxy captures the DD packets from the OSPF peer state manager, and rewrites the topology information in the DD packet according to the resource simulator status. The protocol proxy sends the rewritten DD packet to the IP Optical TE server. The DD packet from the IP optical TE server, is checked the status by protocol proxy, and relays the DD packet to the OSPF peer state manager. After neighbor establishment is achieved, virtual node *R* notifies the resource information using link state (LS) update/request/acknowledge including MPLS/GMPLS extensions. LS update is used to inform other routers of the new information, LS request is used to get the information from other router, and LS acknowledge is used to confirm the packet.

The protocol proxy generates LS packets including MPLS/GMPLS extensions using the resource simulator's output.

The protocol proxy offers two advantages. First, the existing software used to implement extended OSPF functions does not need to be modified. The protocol proxy provides the basic OSPF functions to establish adjacency relations and extended functionally to notify the resource information. The basic OSPF functions are provided by the existing OSPF software, but the extended functions are provided by the protocol proxy. The protocol proxy scheme can implement the extended functions by modifying the protocol proxy. Second, the protocol proxy scheme reduces the complexity of the resource simulator. In using the existing OSPF software scheme, the resource simulator has to control status of all OSPF nodes to pass the OSPF information to the IP optical TE server. In contrast, the protocol proxy scheme aggregates the OSPF information of all virtual nodes at the protocol proxy. Hence, the resource simulator controls only the protocol proxy.

The network emulator does not support the emulation of convergence time. Convergence time is time taken for all router to recognize the information representing a change in network topology. Since the emulation of convergence time is not supported, some topology characteristics, such as link delay, are omitted. The convergence time is generally needed for performance testing. Since the purpose of the network emulator is to test functions of the IP optical TE server, the convergence time is not necessary in OSPF protocol emulation. To obtain the convergence time via emulation, setting some packet delay at the protocol proxy is one candidate.

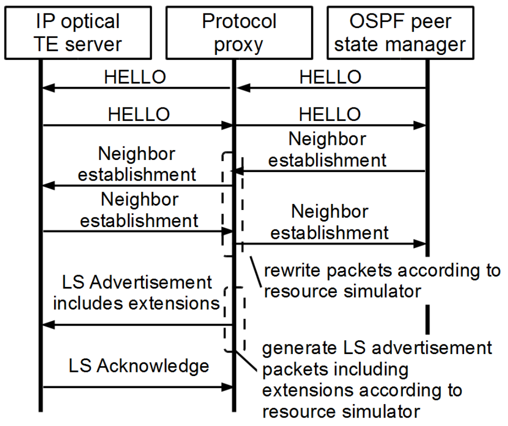


Fig. 7. An example of behaviors of protocol proxy scheme.

VI. IMPLEMENTATION AND RESULTS

A. *Behavior of network emulator*

We developed a prototype to confirm the effectiveness of the protocol proxy scheme. Figure 8 shows the prototype of our developed network emulator; the number of emulated nodes is 40. Three computers were used to implement the network emulator. Each computer was equipped with Software "Xen" [21] was used to realize the VMs. 20 virtual nodes for CLI/SNMP were installed on computers 1 and 2. Virtual node R, resource simulator, and databases were installed on computer 3. Virtual node *R* consists of our developed protocol proxy and the OSPF peer, where Quagga [20] is the existing OSPF software. Quagga is a routing software suite: General Public License (GPL) licensed IPv4/IPv6 routing software.

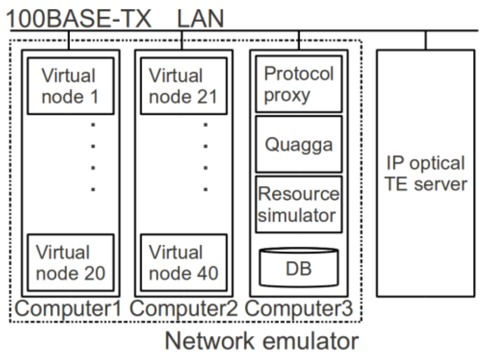
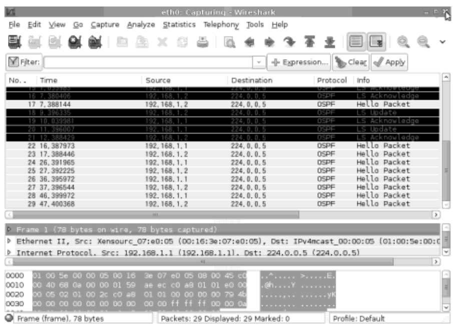


Fig. 8. Prototype of network emulator.

Our goal is to test the functions of the IP optical TE server in a network with several hundred nodes. In general, one machine can be dedicated as a router to implement a virtual node that speaks CLI/SNMP/OSPF. However, this approach is not cost-effective, so we employ the virtual machine (VM) approach. VM technology allows multiple machines to run independently on one computer. This means that we do not take the other approach, which integrates the resource simulator and traffic generator with the virtual nodes. This is because the VM technology enables us to develop the network emulator more easily, a significant benefit since router interfaces are frequently upgraded. Our approach allows the use of several available software packages as the required protocol suites. However, using VM technology prevents the direct emulation of the propagation delay. Because the virtual nodes work in the same computer, they are unable to replicate the real delay in the network.

Figure 9 shows captured packets passed between the IP optical TE server and the protocol proxy. We confirmed that the OSPF protocol was successfully emulated by the protocol proxy.


eth0: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression...
Clear
Apply

```
No. . Time        Source         Destination   Protocol  Info
   16 7.380406    192.168.1.2    224.0.0.5     OSPF      LS Acknowledge
   17 7.388144    192.168.1.2    224.0.0.5     OSPF      Hello Packet
   18 9.396335    192.168.1.2    224.0.0.5     OSPF      LS Update
   19 10.039981   192.168.1.1    224.0.0.5     OSPF      LS Acknowledge
   20 11.396007   192.168.1.1    224.0.0.5     OSPF      LS Update
   21 12.388429   192.168.1.2    224.0.0.5     OSPF      LS Acknowledge
   22 16.387973   192.168.1.1    224.0.0.5     OSPF      Hello Packet
   23 17.388446   192.168.1.2    224.0.0.5     OSPF      Hello Packet
   24 26.391965   192.168.1.1    224.0.0.5     OSPF      Hello Packet
   25 27.392225   192.168.1.2    224.0.0.5     OSPF      Hello Packet
   26 36.395972   192.168.1.1    224.0.0.5     OSPF      Hello Packet
   27 37.396544   192.168.1.2    224.0.0.5     OSPF      Hello Packet
   28 46.399972   192.168.1.1    224.0.0.5     OSPF      Hello Packet
   29 47.400368   192.168.1.2    224.0.0.5     OSPF      Hello Packet

▷ Frame 1 (78 bytes on wire, 78 bytes captured)
▷ Ethernet II, Src: Xensourc_07:e0:05 (00:16:3e:07:e0:05), Dst: IPv4mcast_00:00:05 (01:00:5e:00:0
▷ Internet Protocol. Src: 192.168.1.1 (192.168.1.1). Dst: 224.0.0.5 (224.0.0.5)

0000  01 00 5e 00 00 05 00 16  3e 07 e0 05 08 00 45 c0   ..^..... >.....E.
0010  00 40 68 0a 00 00 01 59  ae ec c0 a8 01 01 e0 00   .@h....Y ........
0020  00 05 02 01 00 2c c0 a8  01 01 00 00 00 00 79 4b   .....,.. ......yK
0030  00 00 00 00 00 00 00 00  00 00 ff ff ff 00 00 0a   ........ ........

Frame (frame), 78 bytes    Packets: 29 Displayed: 29 Marked: 0    Profile: Default
```

Fig. 9. Packet capture.

Figure 10 shows an example of path setup. In Fig. 10, the network emulator has two planes. One is the control and management plane. This plane connects to the IP optical TE server, and provides the CLI and OSPF interface. This plane is visible to the IP optical TE server. The other one is the internal plane. The internal plane is used for communication among the modules in the network emulator. The internal plane is invisible to the IP optical TE server. Table I shows the interface IP addresses in the network emulator. The address of Table I (a) represents the connection the control and management plane. M1 to M40 is the router interface for CLI of the IP optical TE server. C1 is the OSPF interface for the IP optical TE server. The address of Table I (b) represents the connection to the internal plane. I1 to I40 is the virtual node's interface that communicates with the resource simulator. PP is the protocol proxy's interface which communicates with the resource simulator. RS is the resource simulator's interface that communicates with the virtual node and the protocol proxy. OP1 and OP2 are particular interfaces. They are invisible and are not connected to any plane. Instead, they are to provide the basic OSPF functions. We demonstrated optical path setup and release in the optical network. The procedure for optical path setup on the network emulator as follows.

- Step 1: The IP optical TE server makes the adjacency relation with virtual node R via interface C1(10.0.0.254).
- Step 2: The protocol proxy receives the topology and resource information from the resource simulator via PP (192.168.1.50) interface and RS (192.168.1.60) interface.
- Step 3: The protocol proxy informs the topology and resource information to the IP optical TE server via C1 interface.
- Step 4: The IP optical TE server logins to the virtual nodes, which correspond to ingress and egress router interfaces such as M1 (10.0.0.1) to M40 (10.0.0.40), using CLI over telnet. The IP optical TE server sends a path-setup request to the ingress and egress virtual nodes. This emulates the router configurations. The path-setup request information includes several attributes such as interfaces with physical ports and IP addresses, required bandwidth, required route, and switch type. The received information is saved at the ingress and egress virtual nodes.
- Step 5: After the received information is confirmed, the virtual nodes activate the configurations by sending the received information to the resource simulator module from I1 (192.168.1.1) to I40 (192.168.1.40) interface to RS (192.168.1.60) interface.
- Step 6: The resource simulator judges if the path setup request can be accepted by comparing the requested attributes to the available resources in the associated TE DB.
- Step 7: If the path-setup request is accepted by the resource simulator, the associated TE DBs are updated based on the newly accepted path attributes. The acceptance is notified to the ingress and egress virtual nodes by the resource simulator. The information updated at the TE DBs is notified to virtual node R. Otherwise, the resource simulator notifies the rejection to the ingress and egress virtual nodes.
- Step 8: The ingress and egress virtual nodes notify the acceptance or rejection to the IP optical TE server via M1 interface to M40 interface.
- Step 9: If network topology or resource information is changed, the resource simulator sends the updated topology or resource information to the protocol proxy from the resource simulator via RS interface to PP interface.
- Step 10: When the protocol proxy receives the information, the protocol proxy informs the topology and resource information to the IP optical TE server by OSPF via C1 interface.

TABLE I
IP ADDRESS FOR EACH INTERFACE IN NETWORK EMULATOR.

(a) Visible to IP optical TE server

Interface	IP Address
M1	10.0.0.1
M2	10.0.0.2
…	…
M40	10.0.0.40
C1	10.0.0.254

(b) Invisible to IP optical TE server

Interface	IP Address
I1	192.168.1.1
I2	192.168.1.2
…	…
I40	192.168.1.40
PP	192.168.1.50
RS	192.168.1.60
OP1	192.168.2.1
OP2	192.168.2.2

B. *Effects of protocol proxy scheme*

The protocol proxy makes the implementation of the OSPF emulation easy and flexible, and reduces the amount of processing resources required.

First, we examine the effectiveness of the protocol proxy scheme in terms of the software development time. It is difficult to quantify how much the developing time is reduced. Therefore, the number of lines in the source code is considered to be an indication of development time. Figure 11 compares the number of lines in the source codes needed to implement the protocol proxy scheme to that in the existing OSPF software of Quagga. There are 74,000 lines, approximately, in the existing OSPF software. While the source code of the protocol proxy has only 1,000 lines, 74 times fewer lines. In the protocol proxy scheme, we do not need to touch the Quagga source code. Since the protocol proxy is modified to implement extended functions, the modification of Quagga's source code is not required. This means that the protocol proxy scheme reduces the code needed to implement some functions. Therefore, it is expected that the development time is reduced. Furthermore, if the Quagga's source code is modified, we have to confirm that the modification satisfies the standard RFC.

The protocol proxy does not work by its self. To establish adjacency relation, the protocol proxy still needs an OSPF peer state manager such as Quagga. Therefore, the advantage of the

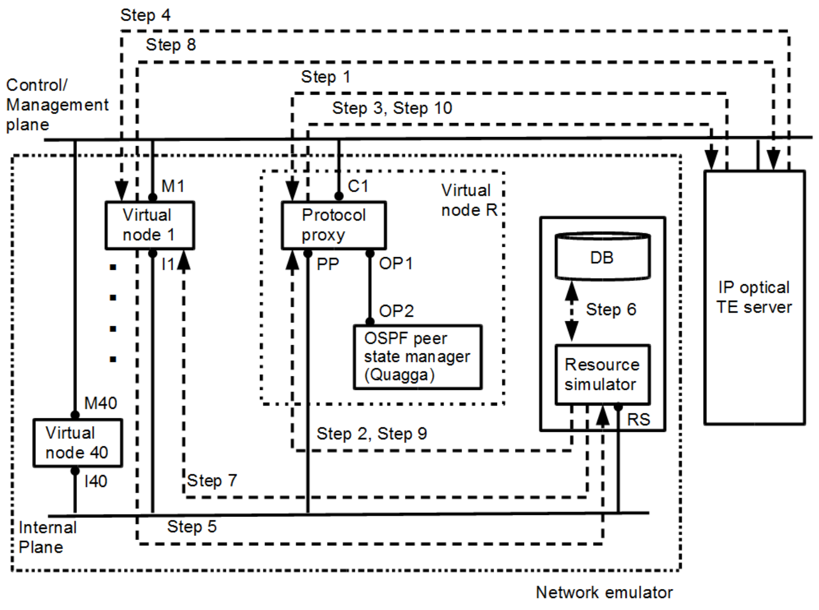


Fig. 10. Illustrative example of path setup.

protocol proxy is that modification of the Quagga in the protocol proxy scheme is not required. Hence, we do not need to test the functions provided by Quagga. This is because it is guaranteed to offer the standardized functions specified in RFC [10]. Quagga is used to establish the adjacency relation. If Quagga is modified, the standardized functions provided by the modified Quagga must be tested.

Second, we examine the processing time on the protocol proxy. We measure the packet rewriting time and LS packet generating time. The packet rewriting time starts from when the protocol proxy receives a packet that needs rewriting, and stops when the modified packet is sent out. The packet generating time starts when the protocol proxy receives the information from the resource simulator, and stops when the packet is sent out. To measure the processing time, we use the getrusage() function, a system call, provided by the Linux operating system. The getrusage() function returns the CPU processing time in millisecond.

The processing time of the protocol proxy is less than one milli-second. This means that the processing time of the protocol proxy has little impact on the network emulator. We note that the processing time of the protocol proxy is not related to network size, because the protocol proxy supports only packet rewriting and generation. It takes less than one millisecond from receiving the information to rewrite and generate the packet. In the protocol proxy scheme, the resource simulator manages the OSPF information. When the network emulator emulates large-scale network, the processing time of the resource simulator is significant. In contrast, the protocol proxy takes less than one millisecond. This means that the protocol proxy is not a bottleneck in the network emulator.

Third, we examine the reduction in the amount of processing resources required. In the conventional approach based on existing OSPF software, all virtual nodes have to install the OSPF software. The existing OSPF software approach demands that the processing resources must be proportional to the number of nodes. On the other hand, the protocol proxy scheme needs only one OSPF peer to perform neighbor establishment. Thus the superiority of the protocol proxy scheme relative to the existing OSPF software approach increases with network size. Figure 12 compares the used memory amount of the protocol scheme with that of the conventional approach. In Figure 12, the used memory of the conventional approach is proportional to the number of nodes. The conventional approach requires 2.3MB memories per node, where OSPF processes run. On the other hand, the protocol proxy scheme requires only 2.8MB memories for a network, which does not depend on the number of nodes in the network. The breakdown is OSPF peer 2.3MB and protocol proxy 0.5MB. This result shows the protocol proxy scheme reduces the required memory amount. We also investigated the required CPU resource by using a command of “pidstat”, which is included in “sysstat version 8”. However, the CPU resource was not measurable, because it is too small. This means that the CPU resource is not a bottleneck for the network emulator.

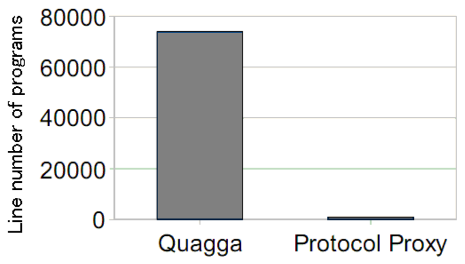


Fig. 11. Comparison of program lines.

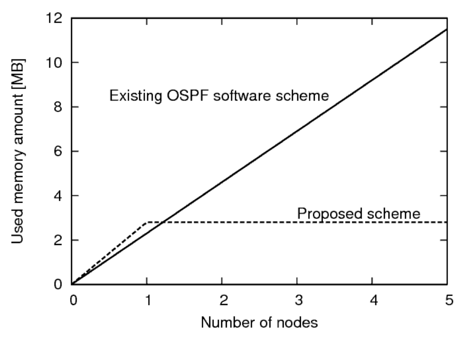


Fig. 12. Illustrative example of path setup.

VII. Conclusions

This paper proposed a protocol proxy scheme to emulate the OSPF protocol in an effective and flexible manner. The protocol proxy scheme reduces the processing resource requirements as well as development time. The protocol proxy scheme realizes OSPF emulation by combining the protocol proxy and OSPF peer state management.

The protocol proxy emulates the OSPF protocol including customized extensions. In implementing the OSPF peer state manager, the existing OSPF software does not need to be modified. The protocol proxy captures packets that are transmitted between the OSPF peer state manager and IP optical TE server.

We developed a prototype of the network emulator. The effectiveness of the protocol proxy scheme reduces the development time. The protocol proxy does not need Quagga to be modified to implement the extended functions. Hence, the standardized functions specified in RFCs provided by Quagga are guaranteed without testing. Moreover, the processing time of the protocol proxy has little impact on the performance of the network emulator.

References

[1] K. Shiomoto, I. Inoue, and E. Oki, "Network virtualization in high-speed huge-bandwidth optical circuit switching network," High-Speed Networks 2008 (HSN 2008), IEEE INFOCOM 2008, Apr. 2008.

[2] K. Matsui, M. Kaneda, and K. Matsuda, "Evaluation of a Server-based Traffic Engineering Architecture Suitable for Large-Scale MPLS Networks," in Proc. *8th Asia-Pacific Symposium on Information and TelecommunicationTechnologies (APSITT)*, 2010.

[3] Y. Koizumi, T. Miyamura, S. Arakawa, E. Oki, K. Shiomoto, and M. Murata, "Stability of virtual network topology control for overlay routing services," Journal of Optical Networking, vol. 7, no. 7, pp. 704-719, Jul. 2008.

[4] K. Shiomoto, E. Oki, W. Imajuku, S. Okamoto, and N. Yamanaka, "Distributed Virtual Network Topology Control Mechanism in GMPLS-Based Multi-Region Networks," IEEE Journal on Selected Areas in Communications, vol. 21, no. 8, pp. 1254-1262, Oct. 2003.

[5] E. Oki, K. Shiomoto, D. Shimazaki, N. Yamanaka, W. Imajuku, and Y. Takigawa, "Dynamic Multilayer Routing Schemes in GMPLS-based IP+Optical Networks," IEEE Commun. Mag., pp. 108-114, vol. 43, no. 1, Jan. 2005.

[6] http://nsnam.isi.edu/nsnam/, 2011.

[7] http://www.nsnam.org/, 2011.

[8] http://www.opnet.com/, 2011.

[9] D. Harrington, D. Harrington, and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," RFC 3411, Dec. 2002.

[10] J. Moy, "OSPF Version 2," RFC 2328, Apr. 1998.

[11] D. Katz, K. Kompella, and K. Kompella, "Traffic Engineering (TE) Extensions to OSPF Version 2," RFC 3630, Sep. 2003.

[12] E. Oki, N. Kitsuwan, S. Tsunoda, T. Miyamura, A. Masuda, and K. Shiomoto, "Scalable Network Emulator Architecture to Support IP+Optical Network Management," IEEE/IFIP Network Operations and Management Symposium (NOMS 2010), Apr. 2010.

[13] E. Oki, N. Kitsuwan, S. Tsunoda, T. Miyamura, A. Masuda, and K. Shiomoto, "Scalable Network Emulator Architecture for IP Optical Network Management," IEICE Transactions on Communications, vol. E93-B, no. 7, pp. 1931-1934, Jul. 2010.

[14] C. Srinivasan, A. Viswanathan, and T. Nadeau, "Multiprotocol Label Switching (MPLS) Traffic Engineering (TE) Management Information Base (MIB)," RFC 3812, Jun. 2004.

[15] E. Oki, T. Takeda, JL. Le Roux, A. Farrel, "Framework for PCE-Based Inter-Layer MPLS and GMPLS Traffic Engineering," RFC 5623, Aug. 2009.

[16] E. Mannie (Ed.), "Generalized Multi-Protocol Label Switching (GMPLS) Architecture," RFC 3795, Oct. 2004.

[17] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209, Dec. 2001.

[18] E. Oki and A. Iwaki, "Load-Balanced IP Routing Scheme Based on Shortest Paths in Hose Model," IEEE Trans. Commun., vol. E93-B, no. 5, pp. 31180-1189, May. 2010.

[19] M. Nathansen, B. Stilling, "Emulation of routing and signaling processes in automatically switched optical networks," in Proc. *Optical Fiber Communication Conference and Exhibit, 2002. (OFC 2002)*, 2002.

[20] http://www.quagga.net/, 2011.

[21] http://www.xen.org/, 2011.

Shunichi Tsunoda received B.E. ain Information and Communications Engineering from The University of Electro-Communications, Tokyo Japan, in 2009. He is with the Department of Communication Engineering and Informatics, Graduate School of Informatics and Engineering, The University of Electro-Communications.

Nattapong Kitsuwan received the B.E. and M.E. degree in Electrical Engineering (Telecommunication) from Mahanakorn University of Technology, King Mongkut's institute of Technology, Ladkrabang, Thailand, and a Ph.D. in Information and Communication Engineering from the University of Electro-Communications, Japan, in 2000, 2004, and 2011 respectively. From 2002 to 2003, he was an exchange student at the University of Electro-Communications, Tokyo Japan, where he did a research about optical packet switching. From 2003 to 2005, he worked for ROHM Integrated Semiconductor, Thailand, as an Information System Expert. His research focus is on IP optical network, optical burst switching, optical packet switching, and scheduling algorithms.

Eiji Oki is an Associate Professor of The University of Electro-Communications, Tokyo Japan. He received the B.E. and M.E. degrees in instrumentation engineering and the Ph.D. degree in electrical engineering from Keio University, Yokohama, Japan, in 1991, 1993, and 1999, respectively. In 1993, he was with Nippon Telegraph and Telephone (NTT) Corporation Communication Switching Laboratories, Tokyo, Japan, where he was engaged in research in developing high-speed optical IP backbone networks. From 2000 to 2001, he was a Visiting Scholar at the Polytechnic University, Brooklyn, NY, where he was involved in designing tera-bit switch/router systems. He is the coauthor of two books Broadband Packet Switching Technologies (Wiley, 2001) and GMPLS Technologies (CRC Press, 2005). His research interests include multimedia-communication network architectures based on asynchronous transfer mode (ATM) techniques, traffic-control methods, and high-speed switching systems. Dr. Oki was the recipient of the 1998 Switching System Research Award and the 1999 Excellent Paper Award presented by the Institute of Electronics, Information and Communications Engineers, and the 2001 Asia-Pacific Outstanding Young Researcher Award presented by IEEE Communications Society, for his contribution to broadband network, ATM, and optical IP technologies.

Takashi Miyamura graduated from Osaka University in 1997 and obtained a master's degree in operations research in 1997 He is currently research engineer at NTT network service systems labs. His main interests are with the management and engineering of Future Networks. He is thus concerned with issues like new routing and transport network architecture, network virtualization, autonomic network control and design in the framework of future networks.

Kohei Shiomoto is a Senior Research Engineer, Supervisor, Group Leader at NTT Network Service Systems Laboratories, Tokyo, Japan. He joined the Nippon Telegraph and Telephone Corporation (NTT), Tokyo, Japan in April 1989. He has been engaged in R\&D of high-speed networking including ATM, IP, (G)MPLS, and IP+Optical networking in NTT labs. From August 1996 to September 1997 he was a visiting scholar at Washington University in St. Louis, MO, USA. Since April 2006, he has been leading the IP Optical Networking Research Group in NTT Network Service Systems Laboratories. He received the B.E., M.E., and Ph.D degrees in information and computer sciences from Osaka University, Osaka in 1987 1989, and 1998, respectively. He is a Fellow of IEICE, a member of IEEE, and ACM.