Design of a Multipurpose Whitebox Networking Platform

Nam-Seok Ko, Hwanjo Heo, Sung-Jin Moon, Sung-Kee Noh, Jong-Dae Park, and Hong-Shik Park

***Abstract*—This paper presents a prototype implementation of a multipurpose network platform which could be easily adapted to accommodate various types of future networks. Our design is motivated by the objective to overcome the drawbacks of existing whitebox networking devices in flexibility and performance points of view. Instead of relying on two extreme ends; ASIC or general purpose CPU, we propose a multipurpose platform which combines the moderate flexibility and rich features for fast packet processing of high-performance network processor, and almost the same flexibility as a general purpose CPU and the offloading functions such as security and deep packet inspection of multicore processor. We also show how we could use the multipurpose platform as a router with deep packet inspection feature by running an open source routing suite as control function, and provide some performance numbers for the system.**

***Index Terms*—Software defined network (SDN), whitebox networking.**

I. Introduction

Most of network devices are provided as an integrated solution; hardware platform and its control software are bundled and tightly coupled together. The inside of hardware system is not disclosed out as well, so it is almost impossible to extend the features or add new features in the system without receiving supports from the manufactures of the products. The closed ecosystem in the network device markets has brought up lots of complaints from the operators of the devices and also from the new manufactures who are willing to seek new opportunities in the area.

The concept of whitebox networking has started to be adopted in networking area as an effort to remove the software's dependency to hardware platform. The term "whitebox" originally was used to call a personal computer or server without a registered brand name which is assembled using off-the-shelf parts. We could run any operating systems on the whitebox which could be bought independently. The operating system could be decoupled from the hardware platform by using the whitebox. The same approach, decoupling of software from hardware, has been tried to the network systems by several startup companies [1], [2]. The software companies could develop their own software without much of dependencies on hardware platform. Pica8 developed Xorplus by extending XORP (eXtensible Open Routing Protocol) and then run it on whitebox systems developed by Pronto Systems which was merged with Pica8 in Feburary 2012. Vyatta also developed Vyatta OS by extending Quagga, another open source routing suites, and then run it on the whitebox systems that they built on their own. The whitebox systems of Pica8 and Vyatta are based on Broadcom ASIC and x86 CPU, respectively.

This whitebox networking approach, however, has its own drawbacks. The whiteboxes based on commodity ASICs are limited in developing new features except the functions which are hardwired when they are manufactured. Therefore, the applications that they could develop on the system are quite limited. The general purpose CPU-based whiteboxes, on the other hand, are very flexible in implementing new features but they have some drawbacks in performance point of view, which makes them be applied to only small size networks.

Software defined networking (SDN) has advanced the whitebox networking model much more by providing a clear isolation of control plane software from the hardware platform with the standardized open protocol between them, OpenFlow [3], [4]. This SDN concept, making switches be dummy and moving most of the intelligence to outside controller, opened the new possibilities to network device market by allowing anyone to develop their ideas and apply them to the real networks; theoretically the control plane does not depend on the specifics of hardware platforms.

The simplicity and flexibility of SDN, however, preclude any advanced features in the data plane on the contrary, which is not easy for one SDN node to have differentiated features from the others. Therefore, big players in networking devices, Cisco, Juniper, HP, etc. are providing the ways to overcome this shortcoming while making the best use of the features that they have in their own devices. For example, Cisco announced its new SDN strategy, Open Networking Environment (ONE) in recent CiscoLive [5]. The ONE provides more open interfaces in both of northbound and southbound directions to fully utilize the existing features of network devices.

When we judge from the recent trends in the networking devices, the whitebox networking model is expected to be evolved into the direction in which the whitebox itself is flexible to include new features while having interfaces with the control functions. We, therefore, present the design of a whitebox system that could be used for various purposes: standalone devices such as firewall, QoS box, and IP router,

Manuscript received October 2, 2012; revised November 5, 2012. This work was supported in part by the the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency. (NIPA-2012-H0301-12- 1004).

Nam-Seok Ko is with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea and Korea Advanced Institute of Science & Technology (KAIST), Daejeon, Republic of Korea (e-mail: nsko@etri.re.kr).

Hwanjo Heo, Sung-Jin Moon, Sung-Kee Noh and Jong-Dae Park are with ETRI, Daejeon, Republic of Korea (e-mail: {hwanjo, sjmoon, sknoh, jdpark@etri.re.kr).

Hong-Shik Park is with the Department of Electrical Engineering, Korea Advanced Institute of Science & Technology (KAIST), Daejeon, Repbulic of Korea (e-mail: hspark@ee.kaist.ac.kr).

DOI: 10.7763/IJFCC.2013.V2.175

or SDN switch node which could be controlled by SDN controller.

The remainder of this paper is organized as follows. In Section II, we present our multipurpose network platform and discuss distinguishing features of the architecture. Then, we discuss a router example running XORP [6], an open source routing suite, on the proposed multipurpose network platform and show performance results of our implementation in Section III. We finally conclude in Section IV.

II. Multipurpose Network Platform

A. Design Goals

We came up with our design goals according to our discussion in section I as follows.

Flexibility and Programmability: It should be possible and not be difficult to add new features or to extend existing features.

Performance: Performance of the system should be affected as little as possible by adding new features.

We also considered the following requirements more specifically to satisfy the goals:

- Packet classification and SDN extensibility by equipping with TCAM
- General purpose processor level high programmability
- Half-duplex 40 Gbps of basic packet forwarding performance (64-byte packet size)
- Deep packet inspection (DPI) ability for application awareness
- Half-duplex 10 Gbps of DPI performance through Regular Expression (RegEx) engine (64-byte packet size)

B. Architecture

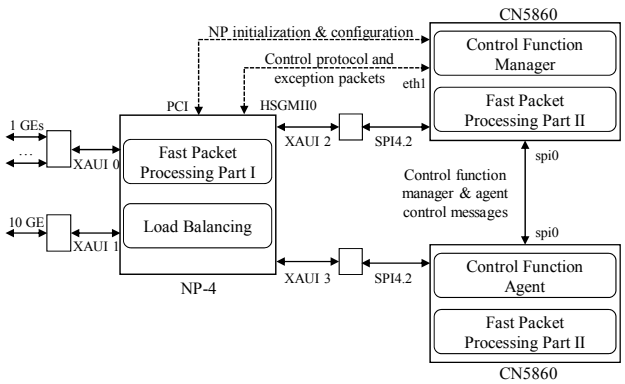

Fig. 1. Proposed multipurpose platform block diagram.

The proposed multipurpose network platform is composed of three major processors as shown in Fig. 1, one network processor (NP) and two multicore processors. The combination of NP and multicore processor is the best solution when we consider the design goals and requirements discussed in Subsection II-A since NP is generally fit for fast packet processing and multicore processor can augment the NP in fast packet processing and also the other specific functions such as security processing and DPI.

There are several commercially available NPs and multicore processors, so several combinations could be viable to work as a multipurpose network platform. The extensive system testing is needed to prove which combination is the best but it requires lots of efforts, time and money. We leave the more extensive comparison of different combinations for further study. Among various off-the-shelf NPs and multicore processors, we chose EZchip's NP-4 [7] and Cavium's CN5860 [8] for NP and multicore processor, respectively. The main features which we considered in NP-4 are as follows:

1) Powerful Search Engine: NP-4 provides a search engine supporting four different types of data structures:direct access table, hash table, tree and multibit trie. So programmers do not need to implement their own search algorithms, which could be drawbacks of the NP when we want to have our own specific search algorithm but the data structures are enough to cover most of existing network applications.
2) Microcode Development Environment: NP-4 is provided with an integrated microcode development environment called EZdesign which supports the cycle accurate simulation capability. Microcode developers could develop new functions without having real target systems with the help of the tool. The tool also makes the debugging easy by providing complete views of system internals such as the values of registers and memory.
3) Easy Programming Model: Single-image programming model with no parallel programming and multithreading could drastically reduce the burden of NP-4 programmers.
4) High Performance: The packet processing performance of NP-4 is enough with integrated traffic management functions: half duplex 100 Gbps. There are several other NPs which have more processing power [9] and EZChip is also planning half-duplex 200 Gbps NP, NP-5 [10], but the performance of NP-4 was enough at the time of system design and is still high enough for general network systems.
5) TCAM Interface: NP-4 supports a TCAM interface which is useful for fast lookups through large tables with wildcards such as multi-field packet classification tables and access control list (ACL).

The main features which we considered in CN5860 are as follows:

1) Flexible Allocation of Multicores: CN5860 has 16 cnMIPS64 cores, Cavium's custom implementation of MIPS64, which could be flexibly allocated in different ways. Since each core has its own memory management unit (MMU) and translation lookaside buffer (TLB), each cnMIPS64 core could be flexibly allocated for an operating system or data plane code.
2) Simple Executive API: CN5860 could be used for fast packet processing without running operating system using the Simple Executive application programming interface (API). It is a Hardware Abstraction Layer (HAL) in the form of an API to the underlying hardware units.
3) Hardware Acceleration Units: There are multiple hardware acceleration units which offload the

cnMIPS64 cores reducing software overhead and complexity. These acceleration units include:

- Per-core security unit
- 32 RegEx search engines for deep packet inspection
- Compression/decompression engine

We could use NP-4 for most of fast packet processing: search of various lookup tables including TCAM, traffic management and load balancing data traffic to two CN5860s. The uses of CN5860s are two fold: both of control plane and data plane processing. The control plane includes the system initialization and application programs such as the routing suite for a router system. The data plane in CN5860 includes DPI and security encapsulation/decapsulation. The CN5860, which is connected to NP-4 through PCI bus, works as a control plane processor as well as a data plane processor. We call the CN5860, the upper CN5860 in Fig. 1, as the master CN5860 and the other as the slave CN5860.

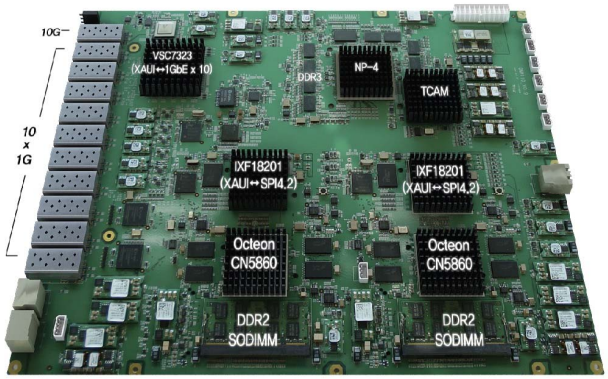

Fig. 2. Picture of multipurpose platform.

The NP-4 and CN5860s are connected with each other as shown in Fig. 1 and Fig. 2. There are four types of main interconnections among them: PCI bus between upper CN5860 and NP-4 for initialization and configuration of NP-4 (type 1), 1 Gbps Ethernet interface between upper CN5860 and NP-4 for exchanging control and exception packets (type 2), 10 Gbps interface between NP-4 and each CN5860 via Cortina's IXF18201 10 Gbps Ethernet MAC controller [11] for delivering data packets (type 3), and 10 Gbps direct SPI 4.2 channel between two CN5860s for transferring both of data packets and control messages (type 4). NP-4 is also connected to one 10 Gbps Ethernet interface and ten 1 Gbps Ethernet interfaces, and each CN5860 has 1 Gbps management Ethernet interface.

The NP-4 does the first fast packet processing from network interfaces and could distribute the packets into the two CN5860s after balancing the load between them only if any further fast packet processing is needed. For example, the basic IPv4/IPv6 packet forwarding is processed in NP-4 and DPI processing is processed in CN5860s as shown in Section III-A. There are three possible packet processing paths in the platform as follows.

- Local Turnaround Traffic Path: 1 Gbps or 10 Gbps Ethernet input interface → ingress and egress fast packet processing in NP-4 → 1 Gbps or 10 Gbps Ethernet output interface
- Full Data Traffic Path: 1 Gbps or 10 Gbps Ethernet input interface → ingress fast packet processing part I in NP-4 → ingress fast packet processing part II in one of two CN5860s → egress fast packet processing in NP-4 → 1 Gbps or 10 Gbps Ethernet output interface
- Control Traffic Path: 1 Gbps or 10 Gbps Ethernet input interface → packet exception to master CN5860 → slow packet processing in controller core in master CN5860 → basic header manipulation processing in NP-4 to send the packets out → 1 Gbps or 10 Gbps Ethernet output interface

Normal data traffic that does not need any further fast packet processing such as DPI, security processing, etc. would take the Local Turnaround Traffic Path and the data traffic that needs those further packet processing would go through the Full Data Traffic Path. In case of this Full Data Traffic Path, NP-4 should distribute packets between the two CN5860. In addition, the cores in each CN5860 could be load balanced using an explicit instruction header that CN5860 requires to do that. The instruction header could be used to send additional information to indicate special packet treatment in CN5860 as well. The control packets such as routing protocol packets and error packets would take the Control Traffic Path.

III. A Router Prototype Based on the Open Source Routing Suite

In this section, we will show a prototype IP router with deep packet inspection capability based on the multipurpose platform presented in section II.

A. Architecture Overview

The first thing we should do is the functional allocation among the NP-4 and two CN5860s when we want to implement a new system based on the proposed multipurpose platform. Table I shows the functional allocation for our prototypes design. Then we also need to decide how to allocate the cores in each CN5860. Both of CN5860s need at least one core for running control plane softwares. The two routing control planes are not equal but rather the one in the master CN5860 runs the main programs of control plane and the other in the slave CN5860 runs some agent programs. Then the rest of cores except the control plane cores are used for data plane processing. The resulting core allocation for each CN5860 in our prototype is as follows: 1 core for control plane software and 15 cores for deep packet inspection. Fig. 3 shows the software architecture of the prototype IP router system.

TABLE I: Functional Allocation of the Prototype IP Router

Components	Functions
NP-4	Data plane processing part I: IPv4/IPv6 forwarding lookup, packet classification and load balancing
Master CN5860	Router control plane software: the main part of XORP Data plane processing part II: DPI
Slave CN5860	Router control plane software: a few XORP processes Data plane processing part II: DPI

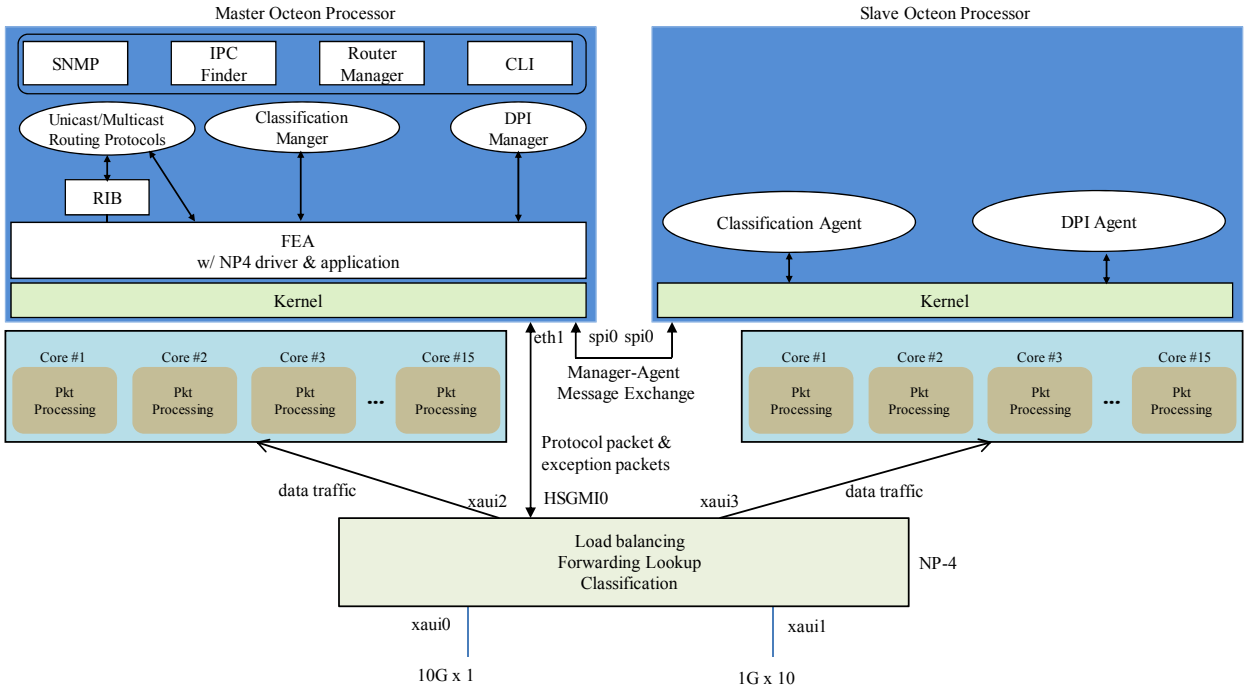

Fig. 3. Prototype IP router based on the open source routing suite, XORP.

B. Control Plane Software Architecture

We used XORP for our base control plane software. XORP supports most of existing IPv4/IPv6 unicast and multicast routing protocols but not all though. XORP also has some more useful features such as XORP inter-process communication (XIPC) mechanism based on the XORP resource locator (XRL) and IPC finder process, which makes it possible for XORP processes communicate each other even when they are located in remote sites. The detailed information about XORP can be found in [6].

NP-4 driver is integrated into forwarding engine abstraction (FEA) process of XORP so that forwarding entries which learned by routing protocols could be added, modified or deleted from the forwarding table which resides in NP-4. Then two new processes, classification manager and DPI manager, were added for managing packet classification rules and DPI rules as XORP processes. The classification manager and the DPI manager manage packet classification rules of the tables in NP-4 and signature tables of CN5860s.

We also implement a virtual ethernet driver on eth1 interface of the master CN5860. The virtual ethernet driver manages logical interfaces of the system. Since the master CN5860 is indirectly connected to the interfaces, one 10 Gbps interface and ten 1 Gbps interfaces, through one Ethernet interface, eth1, a virtual interface per each logical interface of the external interfaces should be created so that XORP could know that the logical interfaces are directly connected.

C. Data Plane Software Architecture

IPv4 packet forwarding and deep packet inspection are processed in the data plane software. More advanced packet processing is possible by adding more features in NP-4 and CN5860 but we just assume NP-4 handles all the IPv4 forwarding and then CN5860 processes DPI when a packet is delivered to them.

When a packet enters NP-4, NP-4 decides how it will treat the packet by looking up the classification table first. It is also decided whether the packet is a control packet or not. If the packet is a control packet, the packet is delivered to the master CN5860, or more exactly the controller core of it, through the type 2 interface which is described in section II-B. When the packet is taking the Control Traffic Path, an additional header is needed to inform the input port information to the virtual ethernet driver.

For the data packets, the decision whether the packet needs DPI processing in CN5860 or not is made with a flag bit in the result of packet classification. If a packet does not need DPI processing, then the packet is taking the Local Turnaround Traffic Path which includes IPv4 multibit-trie forwarding table and ARP table lookups. Otherwise, the packet will take the Full Data Traffic Path.

When the packets are transferred to a CN5860, the packets should be load balanced with the total of 30 cores in two CN5860s, or 15 cores in each CN5860. The load balancing algorithm is not the focus of our prototype system, we use simple hash-based load balancing algorithm. So according to the hash value, the core of each CN5860 is decided.

Then the DPI is performed for the incoming packets in each core of the two CN5860s. In order to test the worst case performance, every packet is going through the DPI process. The performance measure is discussed in the next subsection.

D. Performance Measurement

The performance of the system is restricted by the two CN5860s' performance since the processing power of NP-4, full duplex 50 Gbps, is way higher than the total sum of network interfaces, full duplex 20 Gbps. Therefore, we give only the performance result of the DPI processing in this paper and more in-depth performance tests are left for further study.

We use one 10 Gbps interface to generate 60 different

traffic flows at the rate of 10 Gbps with the packet size of 64 bytes. The headers of 60 different flows are tweaked to have different hash values so that the flows could be load balanced equally to all 30 cores. We also insert different text signatures per each flow and we have a signature table, deterministic finite automaton (DFA) table, with 100 signatures in each CN5860.We could have about a total of half duplex 10 Gbps throughput performance in the system with the above test environment.

IV. Conclusions

We present a design of multipurpose platform which is composed of one EZchips NP-4 and two Caviums CN5860 multicore processors. An example IP router design is also provided with DPI feature. We show that our system can process 5 Gbps input traffic with DPI processing. We argue that the proposed design could be used for various purposes utilizing its flexibility and processing power. We will study further about the possible applications on the platform.

References

[1] Pica8 Xorplus OS. [Online]. Available: http://www.pica8.com/products/.

[2] Vyatta OS. [Online]. Available: http://www.vyatta.com.

[3] Openflow switch specification. [Online]. Available: http://www. openflowswitch.org/documents/openflow-spec-v1.1.0.pdf.
N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.
The cisco open networking environment. [Online]. Available: http://www.cisco. com/web/solutions/trends/open network environment/index.html.

[4] M. Handley, O. Hodson, and E. Kohler, "XORP: an open platform for network research," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 53–57, Jan. 2003.

[5] C. Albrecht, R. Hagenau, E. Maehle, A. doring, and A. Herkersdorf, "A comparison of parallel programming models of network processors," *PARS*, vol. 1, march 2004, pp. 390–399.

[6] Cavium Octeon cn58xx hardware reference manual. [Online]. Available: http://support.cavium.com/.

[7] FP3: The 400g network processor. [Online]. Available: http://www.alcatellucent. com/fp3/.

[8] EZchip's 200-gigabit network processor with integrated traffic management. [Online]. Available: http://www.ezchip.com/products.htm.

[9] IXF18201: 10 Gbps Ethernet media access controller. [Online]. Available: http://www.cortina-systems.com/products/.

Nam-Seok Ko received the B.S. degree in computer engineering from Chonbuk National University, Jeonju, South Korea, in 1998 and the M.S. degree in information and communications engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2000. In 2000, he joined Electronics and Telecommunication Research Institute (ETRI), Daejeon, South Korea. Currently he is working towards the Ph.D. degree with the Department of Information and Communications Engineering, KAIST, Daejeon, South Korea. His research interests include network architecture, network protocols, traffic engineering and software defined networking.

Hwanjo Heo received the B.S. degree in electrical engineering from Korea University, Seoul, South Korea in 2004, and M.S. degree in computer science from Purdue University, West Lafayette, IN, in 2009. He is currently a researcher with Electronic and Telecommunications Research Institute (ETRI), Daejeon, South Korea. His research interests include network measurement, network protocols, and software defined networking.

Sung-Jin Moon received the B.S. and M.S. degrees in electrical engineering from Hanyang University, Seoul, South Korea, in 1992 and 1994, respectively. In 1994, he joined Electronics and Telecommunication Research Institute (ETRI), Daejeon, South Korea. His research interests include network architecture, network protocols, traffic engineering and software defined networking.

Sung-Kee Noh received the B.S. degree in electrical engineering from Hanyang University, Seoul, South Korea in 1990, M.S. degree in Industrial Engineering from Pohang University of Science and Technology (POSTECH), Pohang, South Korea, in 1992, and Ph.D. degree in Chungnam National University, Daejeon, South Korea, in 2006. He is currently a researcher with Electronic and Telecommunications Research Institute (ETRI), Daejeon, South Korea. His research interests include network measurement, network protocols, and software defined networking.

Jong-Dae Park received his B.S, M.S. and Ph.D. degrees in electronic engineering from Yeungnam University, South Korea, 1985, 1987 and 1994, respectively. He was a research fellow in the department of electrical and electronics, Toyohashi University of Technology, Japan from 1995 to 1996. In 1997, he joined Electronics and Telecommunication Research Institute (ETRI), Daejeon, South Korea, where he is currently with the Net-computing Convergence Team as a team leader of the engineering staff. His current research includes packet forwarding engine and software defined networking.

Hong-Shik Park received the BS degree from Seoul National University, Seoul, South Korea, in 1977, and the M.S. and Ph.D. degrees from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in electrical engineering in 1986 and 1995, respectively. In 1977, he joined Electronics and Telecommunication Research Institute (ETRI) and worked on the development of the TDX digital switching system family, including TDX--1, TDX--1A, TDX--1B, TDX--10, and ATM switching systems. In 1998, he moved to Information and Communications University, Daejeon, South Korea as a member of faculty. Currently he is a professor of the Department of Electrical and Electronics Engineering, KAIST, Daejeon, South Korea. His research interests are network architecture, network protocols, and performance analysis of telecommunication systems. He is a member of the IEEE, IEEK, and KICS of South Korea.