Multicore Flow Processor with Wire-Speed Flow Admission Control

Kyeong-Hwan Doo, Bin-Yeong Yoon, Bhum-Cheol Lee, Soon-Seok Lee, Man Soo Han, and Whan-Woo Kim

We propose a flow admission control (FAC) for setting up a wire-speed connection for new flows based on their negotiated bandwidth. It also terminates a flow that does not have a packet transmitted within a certain period determined by the users. The FAC can be used to provide a reliable transmission of user datagram and transmission control protocol applications. If the period of flows can be set to a short time period, we can monitor active flows that carry a packet over networks during the flow period. Such powerful flow management can also be applied to security systems to detect a denial-of-service attack. We implement a network processor called a flow management network processor (FMNP), which is the second generation of the device that supports FAC. It has forty reduced instruction set computer core processors optimized for packet processing. It is fabricated in 65-nm CMOS technology and has a 40-Gbps process performance. We prove that a flow router equipped with an FMNP is better than legacy systems in terms of throughput and packet loss.

Keywords: Flow-based network processor, flow admission control, OpenFlow, flow management, multicore processor.

Manuscript received Apr. 24, 2012; revised July 27, 2012; accepted Sept. 10, 2012.

Kyeong-Hwan Doo (phone: +82 42 860 5374, khdoo@etri.re.kr), Bhum-Cheol Lee (bclee@etri.re.kr), and Soon-Seok Lee (sslee@etri.re.kr) are with the Advanced Communications Research Laboratory, ETRI, Daejeon, Rep. of Korea.

Bin-Yeong Yoon (byyun@etri.re.kr) is with the Creative & Challenging Research Division, ETRI, Daejeon, Rep. of Korea.

Man Soo Han (mshan@mokpo.ac.kr) is with the Department of Information and Communications Engineering, Mokpo National University, Mokpo, Rep. of Korea.

Whan-Woo Kim (wwkim@cnu.ac.kr) is with the Division of Electrical and Computer Engineering, Chungnam National University, Daejeon, Rep. of Korea.

http://dx.doi.org/10.4218/etrij.12.1812.0046.

I. Introduction

In packet-switching networks, a packet flow is a sequence of packets from a source host to a destination [1], [2]. A flow can be uniquely identified by the following parameters within a certain time period: source and destination IP address, source and destination port, and layer-4 protocol, that is, a transmission control protocol (TCP), a user datagram protocol (UDP), and an Internet control message protocol.

In the fourth layer of the open systems interconnection model, there are typically two types of protocol flows: a TCP and a UDP [3], [4]. A TCP is an end-to-end protocol used to compensate for packet loss and disordering caused over networks. A TCP establishes a connection of flows using a three-way handshake. It is used for connection-oriented services like the World-Wide Web (WWW), e-mail, and the file transfer protocol (FTP). For a better throughput of TCP flows at a router, studies have been conducted on the admission control of a TCP flow that discards an initiated packet of the TCP flow in a network node [2], [5]-[7]. These studies show that the TCP admission control improves the performance of TCP flows in terms of the flow throughput and transfer completion time.

On the other hand, a UDP uses a simple transmission model without connection setup procedures to provide reliability, ordering, or data integrity. Therefore, UDP applications must generally accept some loss, errors, or duplication. Common network applications that use a UDP include the domain name system, such streaming media applications as IPTV, voice over IP, trivial FTP, IP tunneling protocols, and many online games. It is expected that the rise of new streaming applications [8] and new P2P protocols [9] will rapidly increase the use of a

UDP as a transport protocol. Congestion-insensitive UDP applications that consume a large fraction of available bandwidth could endanger the stability of the Internet. Network-based mechanisms have been proposed to minimize potential congestion collapse effects of uncontrolled heavy UDP traffic loads. The datagram congestion control protocol (DCCP) was designed as a partial solution to this potential problem by adding end host TCP-friendly congestion control behavior to high-rate UDP streams, such as streaming media [10].

As an alternative solution of DCCP, flow admission control (FAC) is proposed in this paper, in which a connection is set up for a UDP flow. FAC can provide a reliable transmission of UDP applications against congestion and does not inflict other flows in service simply by rejecting a new flow in a congested node. FAC can also be applied to TCP applications, replacing the TCP admission control algorithms used at routers [2], [5]-[7]. Thus, it can improve the performance of flow throughput without looking up synchronize packets to identify the starting packet of a TCP flow in an intermediate node.

OpenFlow was designed as a new network paradigm, which enables researchers to test new ideas on an existing network infrastructure [11], [12]. An enormous amount of effort has recently been focused on the commercial success of OpenFlow through the Open Networking Foundation (ONF) [13], [14]. For large-scale commercial OpenFlow networks, performance of the controller is one of the issues to be resolved [15]. This paper notes that today's controller implementations are unable to handle a huge number of new flows in high-speed networks with 10-Gbps links. Therefore, FAC is believed to be one of the key elements in both the forwarding plane and control plane for OpenFlow, which is trying to innovate current network technologies.

We have developed a network processor (NP) called a flow management (FM) NP (FMNP), also known as an OmniFlow processor, which implements the FAC algorithm based on hardware. The FMNP does not have a pipelined architecture with multiple stages but has rather a single stage with 32 core processors. Thus, it can conduct incorporate packet processing, from packet lookups to scheduling. It also supports a wire-speed connection setup for flows that are not registered in the flow information table and a connection termination of flows in which there is no packet transmitted within a period determined by the users. Wire-speed FAC is expected to be used not only for OpenFlow to develop new network services but also for network QoS to provide reliable transmissions of UDP/TCP applications against congestion. It has been shown that fast FM can contribute to the implementation of various applications [16]. In particular, upon analyzing the routing paths of active flows, it was found that wire-speed FAC is effectively used for the detection and prevention of distributed denial of service (DDoS) attacks [17]. We believe that the FMNP will be used to explore new network services by manipulating and monitoring active flows.

The remainder of this paper is organized as follows. In section II, the structure of the NP and the flow-based mechanism are reviewed. In addition, we propose a scalable architecture of the FMNP. In section III, we analyze the hardware performance of a DDR2 SDRAM controller, which is one of the most critical issues for implementing the FMNP. Through simulations, the performance of FAC is evaluated for a comparison with legacy models in terms of bandwidth utilization and packet loss. Next, section IV shows the results of implementation and verification of the FMNP chip. As an application of the FMNP, the OmniFlow system is proposed in section V. Finally, some concluding remarks are given in section VI.

II. Architecture

An NP is a software programmable device for packet processing, such as pattern matching, address lookups, queue management, scheduling, and so on. It has two architectural features for high performance: multicore processors and a pipeline [18], [19]. A multicore processor executes multiple instructions with two or more independent actual processors (called "cores"), which increases the performance of program execution. According to Amdahl's law, the speed of the program execution in multiple processors is dominated by the sequential fraction of the program rather than by the number of processors [20]. This means that how the workload (program) is assigned to each processor is important for efficient parallel processing. The FMNP, which has 32 multicore processors, usually distributes the workload to each core on a packet-by-packet basis and sometimes assigns the workload on a flow basis to maintain the sequence of incoming packets.

The FMNP does not have a pipelined architecture, and, thus, it can do all packet processing within a single stage, from a packet lookup to scheduling. Cisco's NP, QuantumFlow, is a typical case and enables more flexibility for cooperative packet processing [21].

1. Parallelism

A sequence of packets in the same flow should be maintained over networks to meet the requirements of higher layers, such as TCP [22]. However, there are several factors in changing the sequence in multicore processors. Therefore, multicore NPs require specific schemes to keep the packet sequence of each flow [23]. Figure 1 illustrates how the

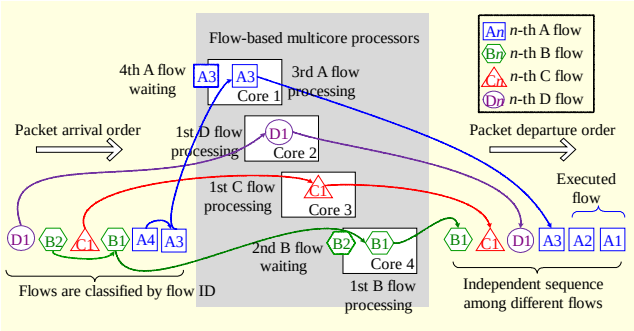

Fig. 1. Flow-based parallel packet process.

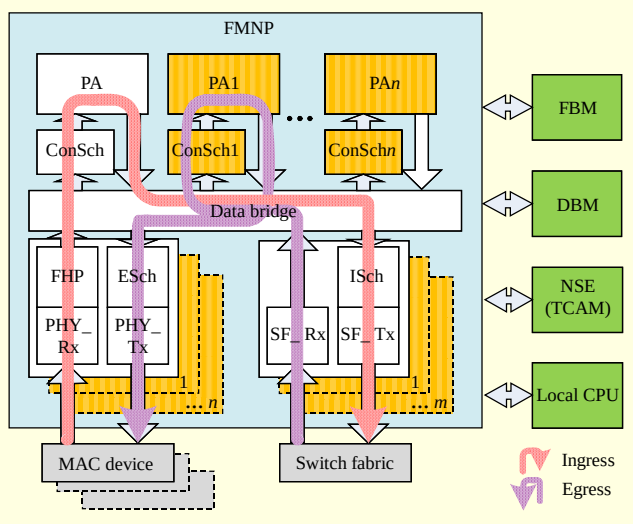

Fig. 2. Scalable architecture of FMNP.

sequence for the same flow is kept in the FMNP. There are eight packets belonging to one of four flows (A*n*, B*n*, C*n*, and D*n*) to be processed in the cores, where *n* represents the order of arrival. When the FMNP receives a packet, it should send the packet to one of the cores for packet processing. The FMNP uses the flow numbers to decide which core to use to process the packet. If there is any core processing a packet with the same flow number as the received packet, the received packet is sent to that core. If there is no core processing a packet with the same flow number as the received packet, the FMNP sends the received packet to one of the idle cores. Based on the rule of packet distribution, packet A3 is sent to Core 1, followed by packet A4, since packets A3 and A4 have the same flow number, A. This makes packet A4 wait in the queue of Core 1 even if all the remaining cores are in an idle state. Packet B1 is assigned to Core 4, followed by packet B2.

2. FMNP Architecture

The FMNP has a scalable architecture, as shown in Fig. 2, which consists of five major blocks: a physical interface block, that is, a flexible header parser (FHP), a physical layer receiver (PHY_Rx), an egress scheduler (ESch), and a PHY transmitter (PHY_Tx); a switch interface block, that is, an ingress Sch (ISch), a switch fabric Rx (SF_Rx), and an SF_Tx; a data bridge; a context Sch (ConSch); and a processor array (PA).

A data bridge is a bridge that connects one physical interface block to one of the PAs or one switch interface block to one of the PAs. Each PA contains 16 multicore processors. The packet processing power is easily increased simply by duplicating the PA block and then pasting it to the data bridge. External data interfaces are also easily extended by duplicating the physical interface block and the switch interface block and then pasting them to the data bridge. The white-colored blocks represent a unit of the functional block used for scalability. The FMNP is the second generation of the device that supports FAC. The first-generation device consists of a unit functional block. The FMNP has a unit functional block and an additional PA and physical interface block.

Incoming packets are processed along two data paths, as shown in Fig. 2: ingress and egress. Ingress packets arriving at the framer interface of the PHY_Rx are sent to one of the PAs through the data bridge. The packets leave the switch fabric interface of the SF_Tx through the data bridge after being processed in the PA. Egress packets coming from the switch fabric interface of the SF_Rx arrive at the PA through the data bridge and then return to the data bridge after packet processing by the PA. The packets finally leave the framer interface of the PHY_Tx.

For functional blocks along the ingress path, the PHY_Rx that receives packets through SPI4.2 from the external MAC device segments the incoming packets into 48-byte chunks and stores them in the data block memory (DBM), except for the first chunk. An FHP, which consists of four reduced instruction set computer (RISC) core processors, parses the first chunk using a microcode and generates a hash key and value to identify the flow for a packet. The data bridge has a mux and demux architecture that connects one of the framer interfaces to one of the PAs. The ConSch determines one of 32 contexts in the PA to process a chunk, as described in section II.1, and then forwards the chunk to the PA. The PA includes 16 RISC core processors. Each processor has two contexts (threads) and an internal memory that can store 2,048 instructions. One of the multicore processors continuously monitors all flows and terminates any registered flow that has an absence of packets for a period of time. The period of each flow, which we define as the "allowed sleeping timeout" (as-timeout), can be set by the users to be short. All of the multicore processors can access the flow block memory (FBM) to store the flow-state information, the DBM to store data packets, and the network

search engine (NSE) to store a lookup table. One of the main jobs of the PA is to schedule packets to meet the QoS requirements. It determines the scheduling information: time to send (TTS), queue number, and output port number. The TTS represents the output time of the packet. The PA sends a pointer to the ISch to locate a packet in the DBM along with the scheduling information above. The scheduler saves the pointer in a calendar queue of the schedule table of the DBM and eventually retrieves it after sending out a complete packet in due time. The SF_Tx converts the packet into internal switch frames to transmit to the switch fabric.

For functional blocks along the egress path, the SF_Rx, which receives egress packets from the switch fabric, stores all packets except for the first chunk in the DBM, after converting the internal switch frames of a packet into chunks. The PA sends the packets to the ESch along with scheduling information after processing them. Like the ISch does in the ingress path, the ESch schedules packets using the scheduling information, such as TTS and queue number. The PHY_Tx sends the packet to the MAC device after assembling the chunks to make a complete packet.

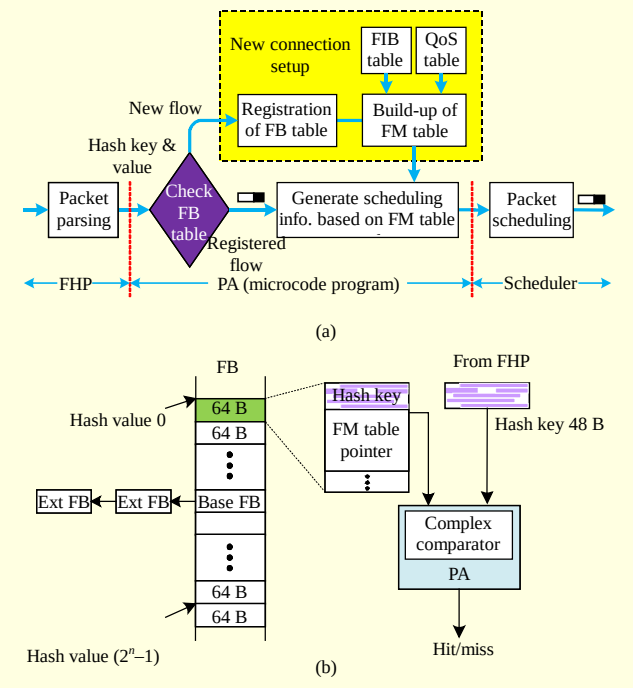

Fig. 3. FM: (a) flow setup and (b) flow matching using hash value.

3. Flow Management

The FMNP has a wire-speed connection setup of flows to support the wire-speed FAC. Figure 3(a) shows the connection setup and packet forwarding procedures.

When a packet arrives at the FMNP, a flow ID is determined by the FHP block, which generates a 48-byte hash key and 22-bit hash value. The hash key is extracted from the header of the packets or a certain part of the packets programmed by the users. Receiving the hash information and the packet, the PA accesses the flow base (FB) table by taking a 22-bit hash value as a table address to search for the flow information of the packet. If no flow information exists at the address, the flow connection is not set up. Thus, the PA starts a new connection setup of the flow. First, the PA registers the new flow into the FB table that stores the hash information, flow statistics, and so on. Second, the PA builds up and stores the flow information of the FM table by referring to the forwarding information base (FIB) and classification tables. The FM table is used to forward the packet to a destination port of the flow and to generate the scheduling information of the flow. The entries of the FIB and FM tables are filled with predefined parameters that are mapped to the flows. The parameters of the FIB and FM tables are determined by the routing protocols and management systems. After creating the FM table, the PA transfers the packet to the scheduler along with the forwarding and scheduling information. If the flow of a packet has already been registered in the FB table when the packet arrives at the FMNP, the PA refers to the FM table to create the scheduling and forwarding information. It then transfers the packet to the scheduler along with this information.

The FIB, classification, and FM tables are all included in the FBM, 1G DDR2 SDRAM, which can be accessed by the 32 multicore processors. The FB table has a maximum of 6M flow entries. Each entry has 64-byte flow data that includes the hash information, as-timeout, statistics, and the pointer to an entry of the FM table for the forwarding and classification information. We use a technique to register the entries, which is shown in Fig. 3(b).

Receiving a hash key and value from the FHP, the PA accesses the FB table by taking a 22-bit hash value as an address of the table. If there is no flow information at the address, the connection of the new flow is set up for the received packet. For the registration, the entry of the FB table is stored with the statistics, as-timeout, hash key, and so on. If any flow information exists in the address, the PA compares the hash key of the received packet with the one stored in the FB. If they match, the stored flow information is used to process the packet. If there is no match, the PA jumps to the extended FB to search for the flow information. It then repeats the same steps as above. After setting the as-timeout of the flow, the PA continuously checks the flow status. The as-timeout of each flow can be set differently and is continuously checked to update the flows. The flows are terminated if there is no packet during the as-timeout. If the period is set to be short, we can

monitor the active flows that have packets during service. This feature can be applied to security systems to prevent DDoS attacks and to a fast FAC to guarantee the QoS of the flows.

III. Performance Analysis

In this section, we analyze the performance from two different perspectives: the bandwidth of a memory controller and an FAC. The hardware processing power of the FMNP can be estimated by the former, and the differentiated FM can be evaluated by the latter.

1. Data Block Memory Interface

One of the most critical issues when implementing a high-speed NP is a lack of bandwidth of the external data memory interfaces used to store and forward incoming packets. We must develop effective arbitration logics to maximize the bandwidth utilization of the memory interfaces. Therefore, analyzing the performance of the total memory bandwidth is important to estimate the processing power of NPs.

A. Structure

The FMNP has a DBM controller (DBMC) with an efficient arbitration algorithm to control many agents that compete to access the DBM, as shown in Fig. 4. The DBMC has four memory interfaces of DDR2 SDRAM DIMM. The DBM externally connected to the DBMC is allocated for the queue buffer to store packets and the queue table to contain the scheduling information. The following agents have access to the DBM for packet processing.

- PHY0_Rx and PHY1_Rx write the queue buffer packets received from the physical links.
- ESch0_A and ESch1_A write a pointer in a calendar queue of the queue table to schedule packets transmitted to the physical links.
- ESch0_B and ESch1_B obtain a pointer in a calendar queue of the queue buffer by a virtual clock and then read a complete packet from the queue buffer to transmit packets to the physical links.
- SF_Rx writes the queue buffer packets received from the switch fabric.
- ISch_A writes a pointer in a calendar queue of the queue table to schedule packets transmitted to the switch fabric.
- ISch_B obtains a pointer in a calendar queue of the queue buffer by a virtual clock and then reads a complete packet from the queue buffer to transmit packets to the switch fabric.
- PA0 and PA1 write the first chunk of packets to the queue buffer after processing.

Table 1 shows the access types and ratios of the queue buffer and queue table in the DBM obtained using a simulator made with Verilog code and Perl script. According to three types of flows, that is, max, typical, and min, we analyze the ratio performance based on the types of DBM access. A "min" flow means that all of the packets are 44 bytes in size. A "max" flow means that all of the packets are 4,096 bytes. Finally, a "typical" flow has two types of packets: one that is 4,096 bytes (75%) and one that is 44 bytes (25%). The DBM has a queue buffer and queue table for ingress and egress packet flows. Many internal blocks of the FMNP have access to the DBM information. Thus, we can classify the access to the DBM into six types of operations, as shown in Table 1. The WR/RD buffer is used to store a packet and retrieve it from the queue buffer of the DBM. The RD table is used to read information from the queue table.

The WR table is used to write scheduling information to the queue table. Table 1 also shows the ratio of the DBM access bandwidth based on the types of DBM access. For a max flow, the ratio of the packet WR/RD buffer is dominant, whereas the

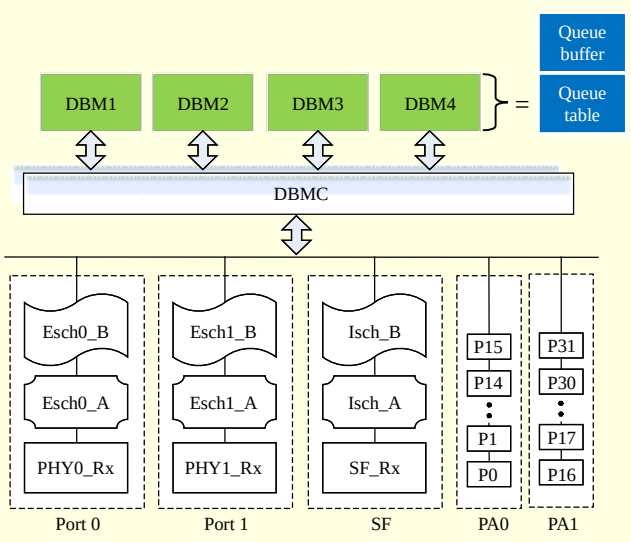

Fig. 4. DBMC interface.

Table 1. DBM access ratio (%).

	Type of DBM access	Max	Typical	Min
Ingress	Buffer WR/RD	49.60	45.00	36.69
	Table RD	0.19	2.38	6.34
	Table WR	0.13	1.33	3.52
Egress	Buffer WR/RD	49.60	45.00	36.69
	Table RD	0.30	4.07	10.89
	Table WR	0.18	2.22	5.87

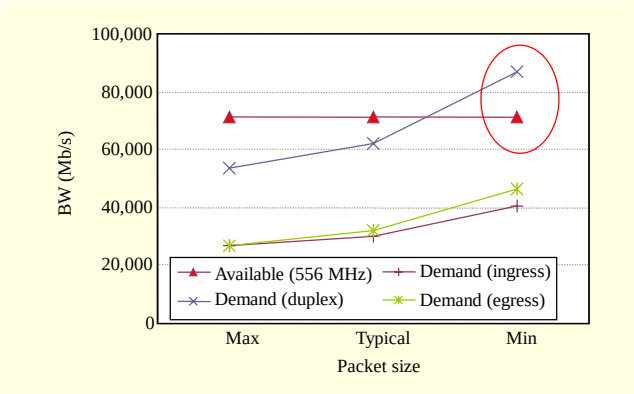

Fig. 5. Available vs. demanded bandwidth of DBM.

ratio of the RD/WR table is small. As the packet size of the flow increases, the ratio of the RD/WR table increases. Table 1 also shows that the ratio of the RD/WR table is higher in the egress path than in the ingress path. The reason for this is that the egress path requires more scheduling procedures than the ingress path requires, in architectural terms.

B. Performance Analysis

We calculated the available bandwidth of the DBMC using (1). Since the FMNP has four external memory buses 64 bits wide and operates using a 556-MHz clock, the total available bandwidth is about 72 Gbps. We assume that the bus utilization is about 50%.

$$BW_{\text{avail}}\,(\text{Mb/s}) = B_{\text{num}} \times W_{\text{b}} \times Intf_{\text{clk}} \times U. \quad (1)$$

B_{num} : the number of banks
W_{b} : bus width per bank (bits)
$Intf_{\text{clk}}$: interface clock rate (MHz)
U : bus utilization

We simulated the demanded bandwidth of the DBM interface using the three types of flows and compared it to the available bandwidth in Fig. 5.

The figure indicates that the FMNP has sufficient power to process typically sized 40-Gbps packets, though it does not support the flow of the minimum packet size (min). However, we think that the actual bus utilization is higher than the estimation, as the bus utilization is higher than the 50% conservatively used in (1). Thus, we believe that the FMNP can almost support wire-speed packet processing.

2. Evaluation of FAC

Generally, the legacy router supporting connection admission control (CAC) determines the acceptance or rejection of a new flow depending on the service level agreement, which includes the flow information, such as the service class, negotiated flow bandwidth, and so on. If the total bandwidth passes over the threshold of the allowed bandwidth by a new flow, the flow is rejected. Contrary to the legacy router, the flow router implemented with the FMNP supports the CAC (as with the FAC) when considering the actual traffic. The flow router does not admit a new flow if the total sum of the average input rates of the admitted flows exceeds the capacity of the flow router. In addition, the flow router can discard the packets of a specific flow under a congestion situation to control the throughput of the flow router.

We simulated three types of routers to estimate the performance of their bandwidth utilization in terms of the CAC: the peak cell rate (PCR)-based CAC (Juniper model), the measurement-based CAC (measurement-based admission control [MBAC] model), and the proposed flow router (FAC model).

In the Juniper router [24], the CAC is performed based on the PCR for a label switched path (LSP) to support DiffServ traffic engineering. In [24], to enhance the link utilization, LSP size oversubscription and link size oversubscription are recommended, especially for best effort traffic. According to [24], "LSP size oversubscription is to configure simply less bandwidth than the peak rate expected for the LSP, and the link size oversubscription is to increase the maximum reservable bandwidth on the link." The oversubscriptions may increase packet delay and cause packet loss [24]. We assume that the LSP size oversubscription is 0.8 × PCR and that the link size oversubscription is 1.2 × output link rate.

In the MBAC model [25], the input traffic is measured over period (P), which is split into units of sampling period (S). The largest value among the input traffic values measured over each unit of the sampling period is considered the current load (CL). A new flow is permitted if the sum of the PCR of the new flow and the CL is less than or equal to the output link rate. It is simulated under the condition of P/S=10 and S=1,000 unit times (2 μs), where the unit time is the transmission time of a 256-byte packet at the output link rate.

We assume that all of the three systems have the same simulation parameters shown in Fig. 6: 32 flows, 110k × 256 byte queue buffer per flow, 1-Gbps maximum output rate, and 256-byte input packets. We use the self-similar traffic model that each connection is fed by a Pareto distributed on-off process. The shape parameters for the on and off intervals are set to 1.4 and 1.2, respectively. Hence, the Hurst parameter is 0.8. It is simulated until the number of admitted packets reaches at least 500M.

The output bandwidth is 1 Gbps, and the flow bandwidth is variable at up to 100 Mbps. We measured the following parameters, increasing the input rate of each flow from 10% of the maximum rate (100 Mbps) to 98%: throughput (utilization),

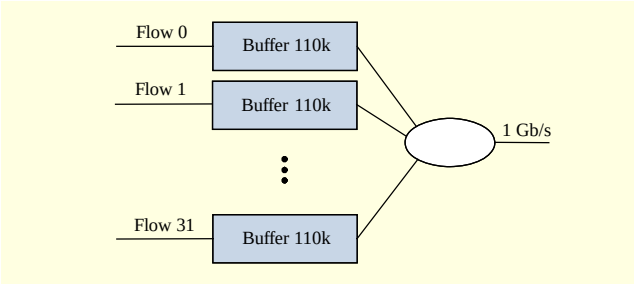

Fig. 6. System configuration for simulation.

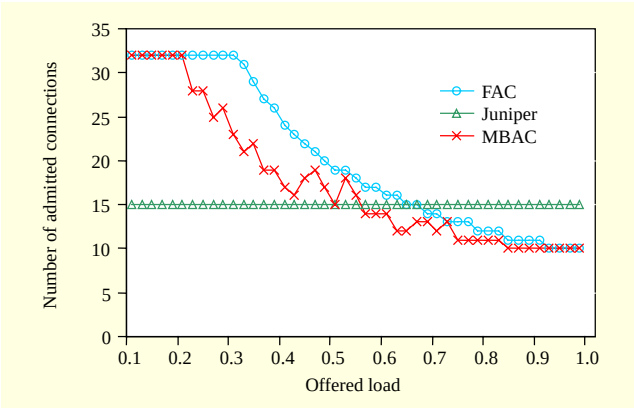

Fig. 7. Number of admitted flows.

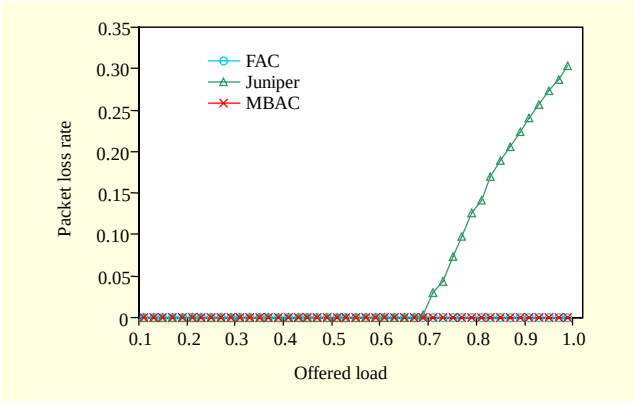

Fig. 8. Packet loss rate of admitted connections.

mean delay, number of admitted flows, and packet loss rate.

Figures 7 and 8 show the number of admitted flows and their packet loss ratio, respectively, as the offered load increases. The offered load means the input rate of a flow. The packet loss in the rejected connections is not considered.

In the case of the Juniper model in Fig. 8, the packet loss starts, owing to traffic congestion, when the offered load reaches 0.7, and the packet loss ratio increases as the load increases because of the LSP and link size oversubscription. The number of admitted flows of the FAC and MBAC models decreases as the load increases. Thus, we can protect a loss of

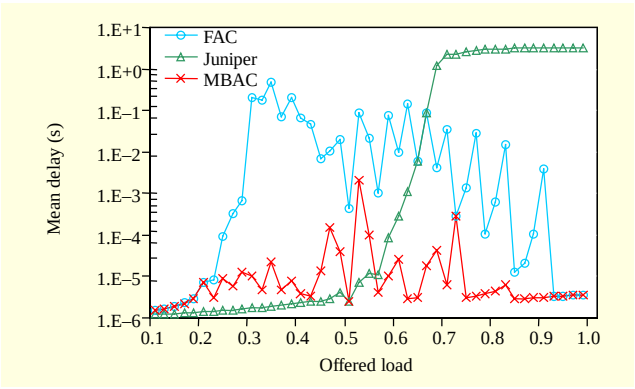

Fig. 9. Mean packet delay.

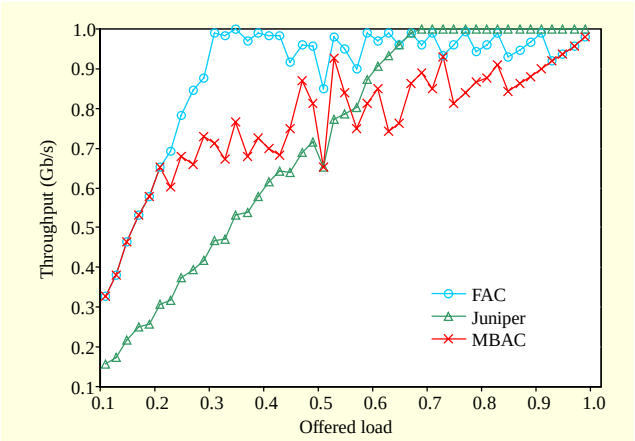

Fig. 10. Packet throughput.

packets in the flows serviced by limiting the number of accepted flows. Even if a packet loss occurs from an unexpected surge of traffic flows, these models can minimize the loss. The number of admitted flows in the MBAC model is less and decreases less regularly than that in the FAC model, depending on the measurement error.

Figure 9 shows the mean delay of packets. The mean delay of the Juniper model increases sharply as the waiting time of packets in the queues increases. The delay time of the FAC and MBAC models repeats the ups and downs after the load reaches 0.2. The reason for this is that the delay time drops whenever the number of admitted flows decreases. Whenever the number of admitted flows decreases by one, a congested state is instantly released, such that the packet processing becomes faster. The mean delay of the FAC model is greater than that of the MBAC model due to the difference in the number of admitted connections.

Figure 10 shows that the throughput of the FAC model is best among the three types of systems, on average. Although the throughput of the Juniper model is best in a high load, the packet loss occurs due to congestion, as shown in Fig. 8. The

throughput of the FAC model linearly increases up to 1 Gbps, while the packet loss rate is almost zero. There are ups and downs in the throughput after the offered load reaches 0.3. The throughput instantly drops whenever the number of admitted flows decreases by one. As we can see from Figs. 8 through 10, it is clear that the FAC model has the best performance in terms of throughput and packet loss rate in this simulation scenario because it performs the CAC based on accurate traffic variables per flow.

IV. Verification and Implementation

The most difficult part of a chip design is the task of verifying whether the logic design conforms to the specifications. In this section, we present the test bench designed for functional verification of the FMNP and then show the implemented chip specification.

1. Design Verification

We developed a test bench architecture that is targeted to verify the FMNP design. The FMNP supports the layer-3 protocol of IPv4, IPv6, and multiprotocol label switching. It has programmable support for Ethernet, frame relay, point-to-point protocol, and fiber channel over Ethernet initialization protocol of the second layer. The complete test bench implemented with Verilog and Perl is designed in a layer-based architecture, as shown in Fig. 11.

The Perl module layer (SL1_1 and SL1_2) is on the top level. The second-level layer is the simulation executable layer, SL2, which interfaces with the Perl layer. The third layer, SL3, is the complete design and test bench component layer. The major Perl modules are a sender and a checker that create a connection to the Verilog test bench through the corresponding sockets and pipes. The sender generates the command and packet data based on the test case configuration. The Verilog test bench receives the command data, which it decodes, and then applies the packet data as a stimulus to the FMNP.

The checker receives the command and packet information from the sender, which it uses as a reference. The checker also receives the resulting data from the Verilog test bench using a receiver socket, which it compares with the reference data to obtain a pass/fail status. The post processor module counts the number of errors to generate the pass and fail statistics. On each interface bus, the Verilog test bench has interface monitors that capture the data on the interface and send it to the checker via Socket2 using Verilog programming language interface (PLI) calls. The functional coverage is captured at the SPI4.2 interface, which targets the various packet formats to be supported in the FMNP. We executed about 1,000 test cases to verify all corner cases of the FMNP design and achieved complete coverage (100%) for the functional verification.

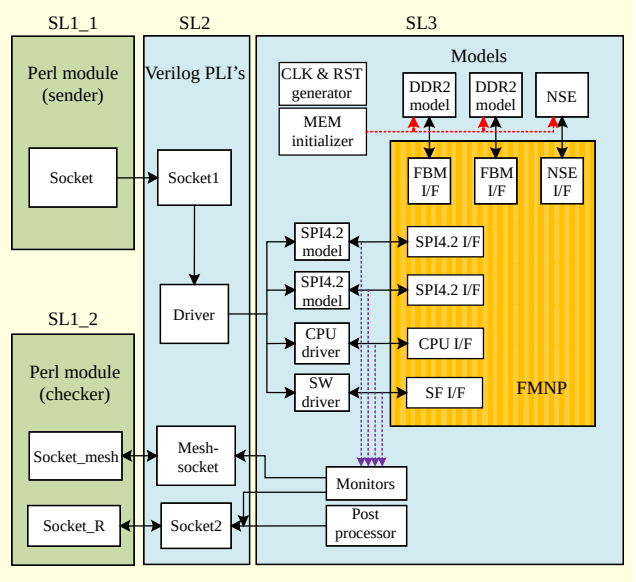

Fig. 11. Layer-based architecture of test bench.

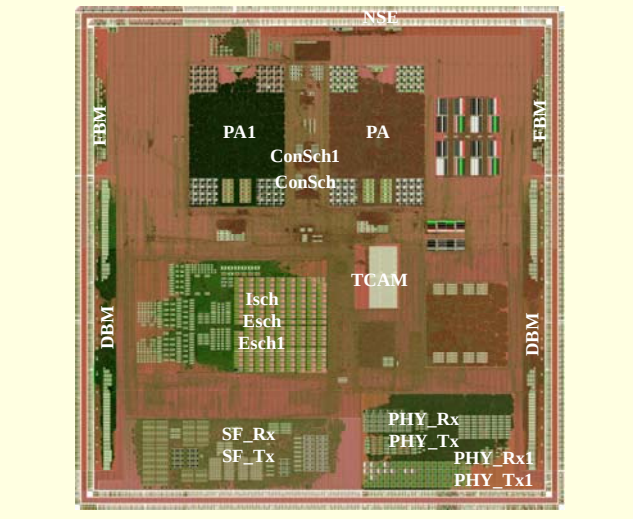

Fig. 12. Layout of FMNP.

2. Chip Implementation

We chose a TSMC foundry, which is suitable for the FMNP design in terms of cost and technology. The FMNP uses several third-party libraries including ternary content-addressable memory (TCAM), phase-locked loop, RAM, and I/O cells. Most of the FMNP design is synthesized using Verilog models of a standard cell library and third-party libraries in 65-nm CMOS technology. Figure 12 shows the layout of the chip.

The FMNP has 2,196 RAM instances, including 140

instances of repairable SRAM. The gate count of the synthesized chip is about 25M, and the total memory bits are 17.6M, except for the repairable memory at 9.25 Mbits. The die size is 17 mm × 17 mm. The power consumption is about 48 W. The core voltage is 1 V. The FMNP package is a 2,377-ball high-performance flip-chip ball grid array with a 12-layer substrate, and its size is 50 mm × 50 mm. A back-annotated timing analysis and simulation were performed under a worst-case condition, and the maximum operating clock rate was 556 MHz with a considered clock uncertainty of 30%.

V. Application

Recent network services have become more complicated by the deep relationship between the L2 and L3 forwarding protocols and the L7 application protocols. This means that a result of L7 protocol processing affects L2 and L4 protocol processing. Some premium network services strongly require the incorporation of protocol processing through multilayers for packet forwarding. This is shown in security systems like intrusion detection and prevention systems, in which incoming packets are inspected by L7 protocols and then dropped or forwarded by L2 or L3 protocols reflecting the results of the L7 protocol processing [26]. Sufficing for the trend of network services and problems described above, the OmniFlow system was invented, which consists of two subsystems: a high-layer OmniFlow system (HLOS) and a low-layer OmniFlow system (LLOS) [27], as shown in Fig. 13.

LLOS, including multiple FMNPs, is well fitted for the protocol processing of lower layers from L2 through L4, generating a flow ID of incoming packets, managing the flow database (DB), and retrieving the flow ID when the flow is terminated. HLOS comprising multiple general purpose processors is used for protocol processing of higher layers like layer 7. It is a scalable architecture that can be easily extended simply by adding more processors if its traffic load increases. The key element of the OmniFlow system is the reference flow ID that provides a good communication channel between two subsystems. Each subsystem has a DB referenced to the flow ID to process incoming packets. The DB has two types of information. Shared information (S-IN) is the information commonly used and shared by two subsystems. Exclusive information (E-IN) is the information exclusively used for each subsystem. The information includes statistics, packet processing rules, and results. By exchanging the information of S-IN with the peer subsystem, each subsystem can maintain the complete amount of information necessary to seamlessly process protocols of incoming packets from a low layer to a high layer. This means that packets with the same flow ID assigned by LLOS are interactively processed by two separate subsystems. It is exemplified that HLOS can sometimes detect a critical virus and attack in incoming packets because LLOS transmitted them to HLOS with a flow ID. If this occurs, HLOS notifies the situation to LLOS along with the flow ID of the packets shared by LLOS. LLOS is then able to block the following incoming packets, so that HLOS is unloaded by getting rid of the job of processing the hazard packets.

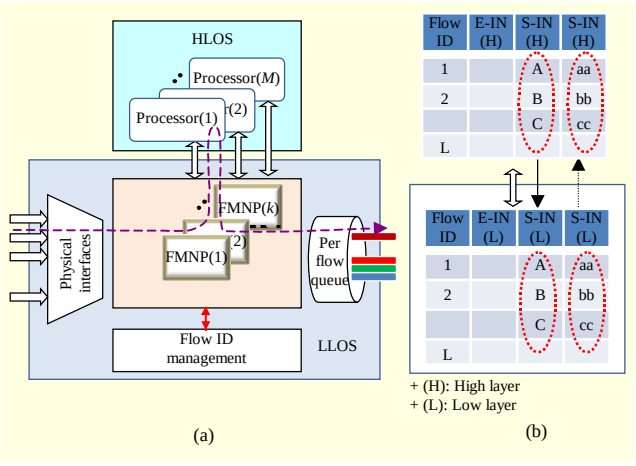

Fig. 13. Configurations of OmniFlow system, consisting of low- and high-layer subsystems: (a) packet flow and (b) control information flow.

The OmniFlow system, which is a flow-based connection system, offers more advantages than existing systems for the following reasons. The first reason is that two separate subsystems can be independently implemented and evolved, resulting in a shortening of the development time of the whole system. The second reason is that we can reuse existing systems or applications of the high layer running on the general purpose processor without modifying it too much. The third reason is that the scalability of the general purpose processor system can be easily expanded to the OmniFlow system through Ethernet interfaces if a user wants to increase the system performance.

A DDoS attack occurs when multiple systems flood the bandwidth or resources of the target systems, which are usually one or more Web servers. This usually creates many abnormal flows within a certain period of time. A DDoS attack can be detected using the relationship between the number of active flows and their characteristics [17]. For example, a DDoS attack can be declared when the total flow bandwidth is abnormally small compared to the number of flows.

As a strong candidate for future Internet use, OpenFlow separates the control plane from the data plane and connects them using an open interface called the OpenFlow protocol. An open interface allows network users to control packet flows in the data (forwarding) plane, which enables the implementation of virtual networks, user mobility, future Internet, and transport

layer protocols. Generally, when the OpenFlow system receives a packet, it first checks whether the packet flow is in service by checking the forwarding table. If the packet flow is not registered in the forwarding table, it is forwarded to the controller. The controller receiving the packet creates the flow information of a new flow, including the forwarding and classification information, and then sends it to the switch. Thus, we observed that there are some similarities between FAC and the packet processing of OpenFlow. Therefore, we believe that a function such as large FM or a wire-speed connection setup can be very useful for implementing OpenFlow systems for future network services.

VI. Conclusion

The proposed FAC supports a wire-speed connection setup. It also terminates a flow that does not have a packet transmitted within a certain period set by users. Thus, it provides the function to monitor active flows that carry a packet over networks during a short time period. We propose that this wire-speed FM will be very useful for the following applications: the reliable transmission of UDP/TCP applications against congestion, security (DDoS), and future network services.

To support a hardware-based FAC, we implemented the FMNP using 65-nm CMOS technology that has a scalable architecture and 40 core processors (32 packet processing cores and 8 packet hashing cores). The FMNP is the second generation of the device that supports FAC. The FMNP is a scalable architecture that easily increases the performance by adding each functional block to an internal bridge. We analyzed the external memory bandwidth, which is the most critical part in the design of a high-speed NP. It was proven that the FMNP has a sufficient performance to support 40 Gbps. We also showed that the flow router equipped with the FMNP was better than legacy systems in terms of throughput and packet loss. We believe that the FMNP will provide solutions for exploring future network services.

References

[1] N. Brownlee, C. Mills, and G. Ruth, "Traffic Flow Measurement: Architecture," IETF RFC 2722, Oct 1999.

[2] J. Rajahalme et al., "IPv6 Flow Label Specification," IETF RFC 3697, Mar. 2004.

[3] J. Postel, "Transmission Control Protocol," RFC 793, Sept. 1981.

[4] http://en.wikipedia.org/wiki/User_Datagram_Protocol

[5] A. Kumar et al., "Nonintrusive TCP Connection Admission Control for Bandwidth Management of an Internet Access Link," *IEEE Commun. Mag.*, May 2000.

[6] R. Mortier et al., "Implicit Admission Control," *IEEE J. Sel. Areas Commun.*, Dec. 2000.

[7] J.W. Roberts and S. Oueslati-Boulahia, "Quality of Service by Flow Aware Networking," *Phil. Trans. Royal Soc. London*, 2000.

[8] P. Pan, Y. Cui, and B. Liu, "A Measurement Study on Video Acceleration Service," *IEEE CCNC*, 2009.

[9] "Micro transport protocol," Wikipedia.org, Apr. 2009.

[10] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," IETF RFC 4340, Mar. 2006.

[11] N. McKeown et al., "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Commun. Rev.*, vol. 38, no. 2, Apr. 2008, pp. 69-74.

[12] "OpenFlow Switch Specification, V1.2," *Open Netw. Foundation*, Dec. 2011.

[13] http://opennetsummit.org/index.html

[14] https://www.opennetworking.org/

[15] M. Jarschel et al., "Modeling and Performance Evaluation of an OpenFlow Architecture," *ITC2011*, Sept. 2011.

[16] N.S. Ko et al., "Quality-of-Service Mechanisms for Flow-Based Routers," *ETRI J.*, vol. 30, no. 2, Apr. 2008, pp.183-193.

[17] H.K. Yi et al., "DDoS Detection Algorithm Using the Bidirectional Session," *CN*, vol. 160, June 2011, pp.191-203.

[18] http://www.ezchip.com/products.htm

[19] http://www.xelerated.com/en/hx/

[20] G.M. Amdahl, "Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities," *ACM Press*, vol. 30, 1967, pp. 483-485

[21] "The Cisco QuantumFlow Processor: Cisco's Next Generation Network Processor," Cisco Systems, 2008.

[22] V. Paxson, "End-to-End Internet Packet Dynamics," *IEEE/ACM Trans. Netw.*, vol. 7, no. 3, June 1999.

[23] S. Govind, R. Govindarajan, and J. Kuri, "Packet Reordering in Network Processors," *IEEE IPDPS*, Mar. 2007.

[24] "Junos OS MPLS Applications Configuration Guide R12.1," *Juniper Netw.*, Mar. 2012.

[25] J. Lakkakorpi, "Flexible Admission Control for DiffServ Access Networks," *Proc. SPIE*, vol. 5244, Aug. 2003.

[26] "Guide to Intrusion Detection and Prevention Systems (IDPS)," *NIST Special Publication 800-94*, 2007.

[27] B.Y. Yoon, B.C. Lee, and S.S. Lee, "Scalable Flow-Based Network Processor for Premium Network Services," *ICTC*, Nov. 2011, pp. 436-440.

Kyeong-Hwan Doo received his BS and MS in electronics engineering from Chonbuk National University, Jeonju, Rep. of Korea, in 1996 and 1998, respectively. In 2001, he joined ETRI, Daejeon, Rep. of Korea, and has participated in several projects, including work on the PON MAC device, flow processor design, and ATM switching systems. Currently, he is working toward his PhD at Chungnam National University, Daejeon, Rep. of Korea. His research interests are passive optical networks and network processor design.

Bin-Yeong Yoon received his BS and MS in electrical engineering from Chung-Ang University, Seoul, Rep. of Korea in 1986 and 1991, respectively. He received his PhD in electrical engineering from Chungnam National University, Daejeon, Rep. of Korea in 2004. He joined ETRI in 1991, where he has worked on several projects, including PON MAC device design and interface design of routers, MPLS, ATM, and TDX. Recently, his research interests are SDN (OpenFlow) and network processor design.

Bhum-Cheol Lee received his BS in electrical engineering from Kyunghee University, Seoul, Rep. of Korea, in 1981. He received his MS and PhD in electrical engineering from Yonsei University, Seoul, Rep. of Korea, in 1983 and 1997, respectively. In 1983, he joined ETRI, Daejeon, Rep. of Korea, where he worked on the development of the TDX digital switching system family, including TDX-1, TDX-10, and ATM switching systems. In 2009, he worked on the development of the OmniFlow processor for the QoS Router system. His research interests are flow processors, virtual networks, and synchronization.

Soon-Seok Lee received his BS, MS, and PhD in industrial engineering from Sungkyunkwan University, Seoul, Rep. of Korea, in 1988, 1990, and 1993, respectively. In 1993, he joined ETRI, Daejeon, Rep. of Korea, where he has worked on several projects related to high-level designing and planning of networks, such as ATM networks, mobile networks, and optical networks. He is currently working in the engineering area for future networks as the director of the Computing Network Research Department and a principal member of the engineering staff in ETRI. His research interests are network architecture and its business models, including content-centric networks, named data networks, and unified all-IP converged networks.

Man Soo Han received his BS, MS, and PhD in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Rep. of Korea, in 1992, 1994, and 1999, respectively. He was a senior researcher at ETRI, Daejeon, Rep. of Korea, from 1999 to 2003. He is an associate professor in the Department of Information and Communications Engineering at Mokpo National University, Mokpo, Rep. of Korea. His research interests include scheduling in high speed networks, passive optical networks, and wireless networks. He is a member of IEEE, OSA, IEICE, and KICS.

Whan-Woo Kim received his BS in electronics engineering from Seoul University, Seoul, Rep. of Korea, in 1977 and his MS and PhD from KAIST, Daejeon, Rep. of Korea, and the University of Salt Lake City, UT, USA, in 1979 and 1988, respectively. Since 1980, he has been with the Division of Electrical and Computer Engineering at Chungnam University, Daejeon, Rep. of Korea, where he is a professor. His current research interests include cable, digital broadcasting, and OFDM systems.