Simulation Analysis for Verifying an Implementation Method of Higher-performed Packet Routing

Jaewoo Park, Seong-Yong Lim and Kyou Ho Lee

Router Technology Dept., ETRI, 161 Gajong-dong, Yusong-Gu,Deajon ,305-350, Korea
Tel: 042-860-5723, Fax: 042-860-5213
E-mail: {parkjw,seylim and kyou}@etri.re.kr

As inter-network traffics grows rapidly, the router systems as a network component becomes to be capable of not only wire-speed packet processing but also plentiful programmability for quality services. A network processor technology is widely used to achieve such capabilities in the high-end router.

Although providing two such capabilities, the network processor can't support a deep packet processing at nominal wire-speed. Considering QoS may result in performance degradation of processing packet. In order to achieve faster processing, one chipset of network processor is occasionally not enough. Using more than one urges to consider a problem that is, for instance, an out-of-order delivery of packets. This problem can be serious in some applications such as voice over IP and video services, which assume that packets arrive in order. It is required to develop an effective packet processing mechanism for using more than one network processors in parallel in one linecard unit of the router system. Simulation analysis is also needed for verifying the mechanism.

We propose the packet processing mechanism consisting of more than two NPs in parallel. In this mechanism, we use a load-balancing algorithm that distributes the packet traffic load evenly and keeps the sequence, and then verify the algorithm with simulation analysis. As a simulation tool, we use DEVSim++, which is a DEVS formalism-based hierarchical discrete-event simulation environment developed by KAIST. In this paper, we are going to show not only applicability of the DEVS formalism to hardware modeling and simulation but also predictability of performance of the load balancer when implemented with FPGA.

1. Introduction.

When designing a IP router, we implement the forwarding engine of linecard with Network Processor (NP) instead of using ASIC. NP provides two characteristic: Programmability and High Performance somewhat. ASIC doesn't provide programmability. But in performance, unlike ASIC, NP doesn't provide deep packet processing at nominal wire speed that vendors argue.

In designing a fully distributed router, each line card has distributed forwarding engine and line interface. The line card supports simple packet forwarding by looking up forwarding table and QoS. In IntServ or DiffServ model, almost packet processing function is performed in line card. Packet filtering using Access Control List, Flow identification and state maintenance, and fine-grain traffic control function is performed in line card also. The more functions are added, the more performance degrades.

When considering QoS mechanism, we need more processing power to maintain the nominal packet processing speed. So, we adapt parallel processing mechanism using two NPs and load sharing between them. Two NPs are connected in parallel between the line interface and switch interface. We double up the processing power and process the packet without the packet loss. But, when the NP shares the input packet by simple distribution mechanism such as time division multiplexing-like method, there will be packet sequence error. Some applications, for example VoIP traffic and video stream traffic, assume that packet arrives in order and regards the out-of-order packet as packet loss and drop it. This problem can be serious in some applications such as voice over IP and video services, which assume that packets arrive in order.

It is required to develop an effective packet processing mechanism for using more than one network processors in parallel in one linecard unit of the router system. Simulation analysis is also needed for verifying the mechanism.

The line card containing two network processors needs a load balancer. The load balancer has more than two destinations and distributes packet traffic loads evenly. Accordingly, it would be desirable to share packet traffic loads among more than one such path, while maintaining the order in which the packets were sent in all cases where order matters.

To resolve such a problem, we propose a

load-balancing algorithm performing on a per-flow basis.

2. System Architecture

The target system shown in Fig. 1 is a switched router system under development. The system consists of a switch fabric with 16 x 1Gpbs input/output ports, multiple linecards providing connectivity to networks, and a control unit. Each of linecards has a 1 Gbps packet-forwarding engine. One of them is in charge with a uplink path requiring a 2~2.5 Gbps processing rate. It should require using two or more chipset of the network processor which is capable of 1Gbps packet-forwarding.

In distributed architecture, each line card has forwarding table and forwarding engine. A distributed forwarding table is fully synchronized with the table in the routing protocol processing module in system CPU periodically.

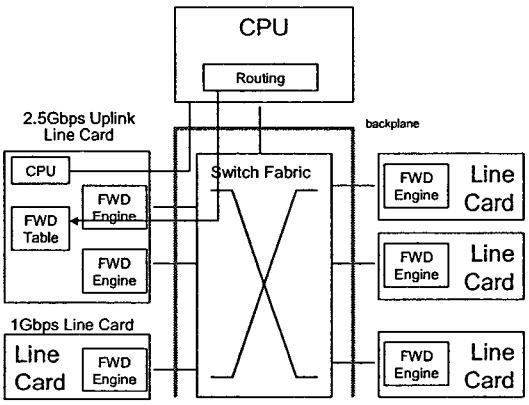

Fig. 1. Fully Distributed Forwarding-based router

In general, a router has an uplink port that has higher data rate that other interface port. The uplink line card is shown in Fig 1. It consists of the block of initialization and status check, PHY interface, Forwarding Engine, Address Lookup unit. Switch fabric supports a link aggregation (Port Trunking) of 2 or more switch link port and regards the aggregated port as one logically. In the uplink linecard, two chipset of network processor is used as shown in Fig. 2. Each network processor is connected to the each switch port of the logically aggregated link.

In fig. 2, Search Machine executes an IPv4 Longest-Prefix-Match search in CIDR for the next-hop lookup. CPU Module communicates with the main routing processor through IPC, initializes and controls the linecard. We are implementing a 2.5Gbps POS (Packet Over SONET) as an uplink interface. Packet Convert Module has the POS PHY Level 3 interface. It converts the packet from the PHY module into the form required by the Network It adds the control words of 32-bit width representing the packet information. In the egress, MUX module receives the packet from the two network processors, removes the header and tail word and sends the packet to the convert module whose header arrives earlier. If packets arrive at the same time, packets from No. 1 network processor are first served.

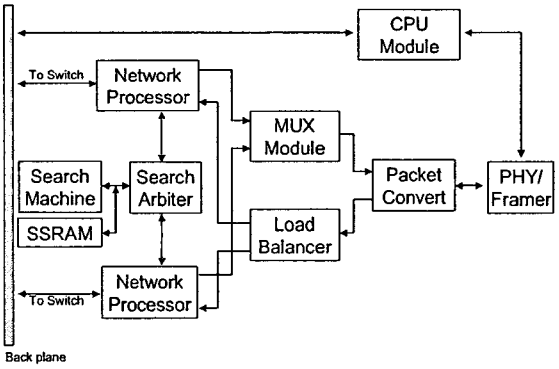

Fig. 2. Structure of 2.5Gbps uplink linecard

In the ingress, there is the Load Balancer as shown in Fig. 3. Load Balancer distributes the packet from the Packet Convert Module by the load-sharing algorithm. The algorithm should satisfy the characteristic of even distribution and packet sequence preservation.

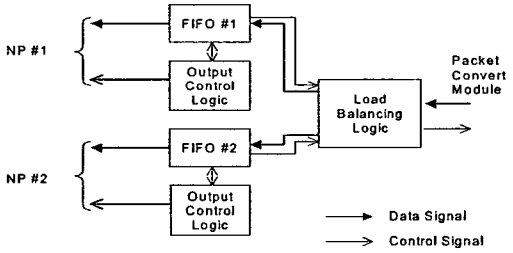

Fig. 3. Load Balancer

There are several simple algorithms that can be used such as giving a priority to a specified FIFO or sending a packet to one FIFO. If the overflow happens, then send packet the other FIFO. One of the algorithms can distribute packets to each network processor on by one alternatively. If a packet from PHY module is small and the next packet is large and the input pattern repeats every two packets, an overflow happens in one FIFO and packet sequence error will happen. To solve the overflow, Time Division-like Packet distribution algorithm can be used. This algorithm allocates the time slot to each network processor by packet boundary. But it can't also solve the problem of packet sequence error.

Simple algorithms like above can distribute packets evenly in long term, but ignore the packet

flow and cannot maintain the packet sequence in router.

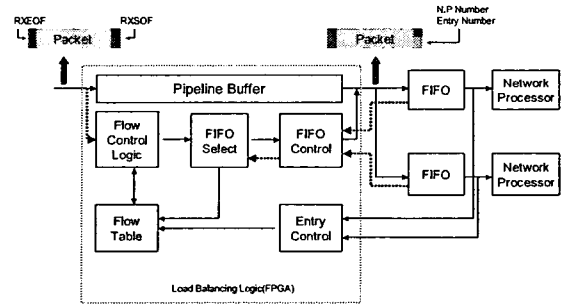

Fig. 4. Structure of Load Balancing Logic

To resolve such a problem, we propose a load-balancing algorithm performing on a per-flow basis. Fig. 4 shows the internal structure of Load Balancing Logic of the algorithm that satisfies the two characteristics. A flow is a sequence of packets transmitted between a specified source and a destination, generally representing a signal session using a known protocol. Each packet in one flow is expected to have identical routing and access control characteristics. We define a flow as a packet sequence that has the same source and/or destination IP addresses. If the packets of same flow are sent to the same network processor, packet sequence will be preserved.

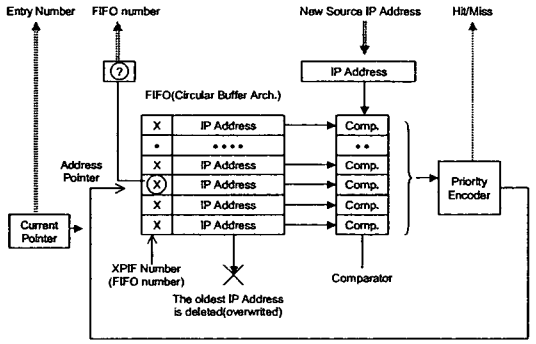

Fig. 5. Flow Table

The load balancing logic consists of such components as flow table, flow control logic, pipeline buffer, two FIFOs and its control logic, and FIFO selection compensator. Flow Control logic extracts the source IP address from newly injected packet and sends the address to the flow table. The flow table has the information of packet flows that pass the linecard for specified time. It has the table of FIFO-like structure. Each entry has a field of address, number of allocated network processor. It inputs the address, compares the address with the table entry and outputs hit/miss signal, entry number, and number of network processor. If the hit signal is active, Flow Control Logic modifies the packet header with the information, number of network processor, from the flow table. If inactive, FIFO select compensator allocates the packet to the FIFO, modifies the packet header and adds the new entry to the flow table. FIFO Control block lookups the packet header for the number of FIFO and sends the packet to specified FIFO. FIFO Control block relay the pack pressure signal to the Packet Convert Module

3. Function Modeling and Simulation of Load Balancing Logic

3.1 Models of Load Balancing Logic

A set-theoretic formalism, the DEVS formalism [3] specifies discrete event models in a hierarchical, modular form. Within the formalism, one must specify the basic models, from which larger ones are built, and how these models are connected together in hierarchical fashion. A basic model, called an atomic model, has specification for dynamics of the model. The second form of the model, called a coupled model, defines how to couple several component models together to form a new model. For effectiveness of load balancer system, suggested in this paper, that system (fig. 4) is modeled as load balancer model with packet generator and NPs, based on the DEVS formalism.

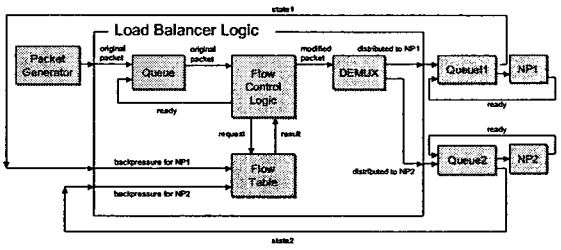

Fig. 6. Load Balancer Model

In fig. 6, there are considered as three parts of the experimental system, Packet Generator (PG), Load Balancer Logic, and Network Processors (NPs). The atomic model PG is used to model the packet traffic nodes. Such model repeatedly generates packets with exponential duration and sends them to Load Balancer Logic with some attributes; packet ID, source IP, input port, destination IP, output port, service NP, enter time, and service time. The coupled model Load Balancer Logic is developed from input queue, Flow Control Logic, Flow Table, and Demux for describing behavior of packet analyzing, classification, marking, and distribution.

In top of fig. 7, Flow Control Logic (FCL) receives input packets from input queue when FCL is ready to do. FCL has two phases behavior, and the one behavior is packet receiving and request to Flow Table (FT) for packet classification. The other behavior is receiving modified packet and transmitting to Demux.

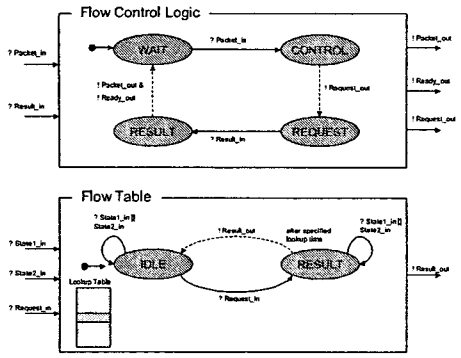

Fig. 7. Atomic Model of Flow Control Logic and Flow Table

In bottom of fig. 7, FT has a relation list with some columns; flow ID, Source IP, Destination IP and Service NP. FT that is requested with input packet searches the row of same source IP and destination IP. If such row is existed, the row is refreshed and the packet is marked service NP attribute with the item of the row. If not, new row, related with source IP and destination IP of input packet, is appended, and service NP is decided from the comparison of output queue states for load balancing.

Finally, distributed output queues and NPs receive all output packets and calculate packet delay and serialized status.

3.2 Simulation Result

Fig. 8 shows load-balancing result from summation of total packet numbers with different source IP. All packets with a same flow ID were serviced in a same NP, and all flows were distributed to two NPs almost balanced.

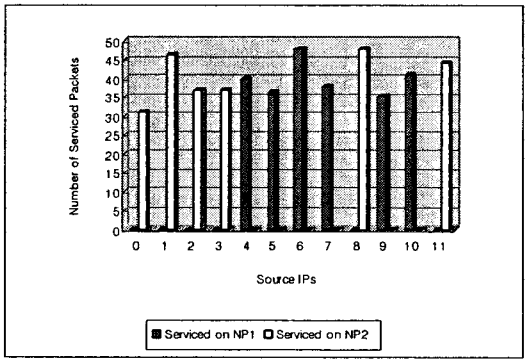

Fig. 8. Flow classification result with source IPs

As seen in fig. 9, all packets are serviced at almost the same time independent of the source IP and the NPs process the packets of the same flow with the same load balance. Fig. 10 illustrates the load of the service NPs. It is clearly seen that adequate load balancing in being achieved when comparing the processing time and packet processing capacity of the two NPs.

Through the three figures shown above, using the algorithm that this paper proposes, when input packet flows with different source IP enters the system, the flows are adequately distributed across the NPs and the packets are serviced at almost similar time and the same flow are processed at same NP.

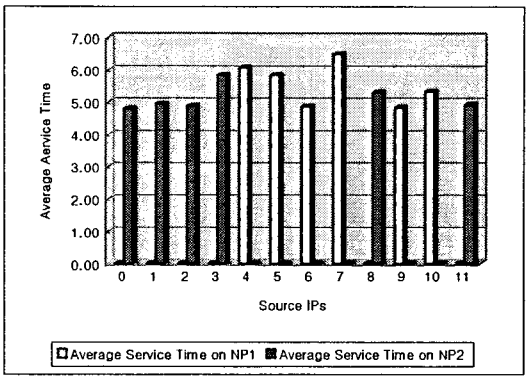

Fig. 9. Average Service Time Result of Serviced Packet with Source IPs

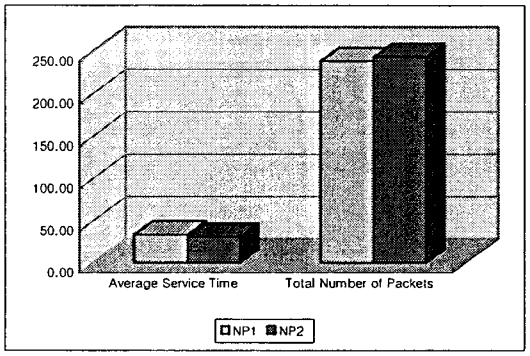

Fig. 10. Average Service Time and Number of Packets Comparison of NP1 and NP2

4. Conclusion

The algorithm can be used to implement the uplink linecard having the multiple speeds of other ports of switched router. When using more than two network processor in parallel, if one network processor fails, the performance of linecard degrade gracefully. Generally, the speed of transmission line advances the packet processing speed of linecard. Parallel architecture and the load-balancing algorithm are very useful for the router. This method is applicable to the linecard of OC-192 and firewall.

5. References

[1] Jaewoo Park, Kyouho Lee, Design and implementation of POS inteface linecard in high-end router, NCS2000, pp. 476-479, Dec. 2000

[2] W Bux W.E.Denzel and T. Engberson, Technologies and Building Blocks for Fast Packet Forwarding, Communications Magazine, pp. 70-79, Jan. 2001

[3] B.P. Zeigler, H. Praehofer, T.G. Kim, Theory of Modeling and Simulation 2nd, Academic Press, 2000.