UDC 621.395.31:681.3

High-Performance IP Service Node with Layer 4 to 7 Packet Processing Features

●Tsuneo Katsuyama ●Akira Hakata ●Masafumi Katoh ●Akira Takeyama

(Manuscript received February 27, 2001)

In this paper, we propose an IP service node that provides sophisticated, high-performance IP packet control features. The proposed node processes not only layer 3 IP packets but also those of layers 4 to 7 in the OSI (Open Systems Interconnection) reference model. It can control application-oriented network services such as URL (Uniform Resource Locaters) based server load balancing, contents routing, and end-to-end QoS (Quality of Service) management, and it will play an important role in creating a new infrastructure in the mature IP age. An IP service node architecture closely connected to a high-speed packet processing engine and software-based server functions within the IP service node itself are discussed. An experimental system has been developed, and the proposed architecture was evaluated with a server load balancing application. We confirmed that the IP packet control functions work well and that the transfer speed is constant at about 800 k packets/s, which is about 10 times that of a conventional server based on a general processor.

1. Introduction

Recently, the Internet traffic has been growing at a tremendous pace and various application services provided on the Internet such as WWW (World Wide Web) databases, mobile computing, and electronic commerce (EC) are becoming indispensable in our everyday life. However, to keep up with the growing demand, the future Internet must have a higher performance and must have a sophisticated IP packet control functionality.

In this paper, we propose a new IP service node that processes not only layer 3 packet data but also higher layer information such as the packet data of layers 4 to 7 in the OSI reference model.

Various studies indicate that optical transmission technology has been growing successfully and that transmission capacity is increasing according to Guilder's law,[1)] with a doubling in performance every six months. Semiconductor technology, on the other hand, continues to improve according to Moore's law, with a doubling in performance only every 18 months. Because the performance of an IP service node greatly depends on semiconductor technology, this 6-month/18-month difference in growth rates may cause the IP service nodes to become network bottlenecks. Furthermore, because IP service node functions will often be added, changed, and integrated according to the customer's needs, a new IP service node architecture to realize high-speed packet data processing and flexible implementation is necessary. We believe that the network processor, which processes IP packet data efficiently by using micro-code, will be a key component of future IP service nodes. The network processor is a programmable semiconductor device designed for network applications such as IP packet forwarding, routing, and higher layer processing, including security features.

In this paper, we summarize the requirements of the IP service node and discuss the IP service node development policy. Then, we pro-

pose a new system architecture and high-speed packet processing mechanism. Finally, we describe an experimental system that uses network processors and evaluate its performance in a server load-balancing application.

2. Network services and the IP service node

2.1 Direction of network service enhancement

Current router networks only provide an IP packet transfer capability and convey the user packets by using IP address information. However, to execute application services more efficiently, we think that the future network will need to be controlled using a higher-level, for example, by using application-oriented information. **Figure 1** shows the example network control task of balancing multiple server loads. The same contents are stored in multiple servers, and the loads of the servers specified in the URL information are managed so they are equal. The session information between individual users and servers is also managed, and the packets from specific users are transmitted to specific servers during the session. The above-mentioned network functions can control the network servers efficiently and can smoothly execute application service procedures such as electronic commerce (EC). For the load balancing service, higher-level information such as XML (eXtensible Markup Language) is expected to be useful.

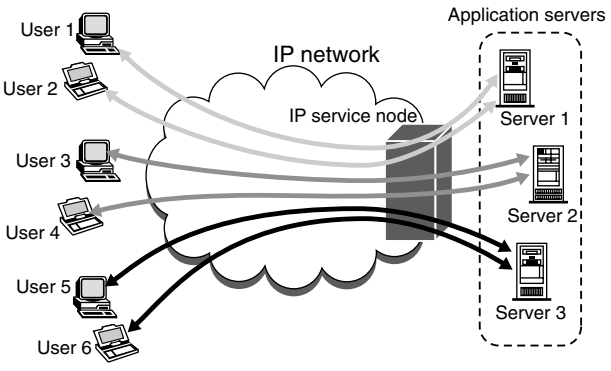

Figure 1
Network control example: balancing multiple server loads.

2.2 Requirements of the IP service node

Before discussing the requirements of the IP service node, we will give some background information about packet processing functions and the transfer flow of network devices. Generally speaking, there are three types of IP services:

1) Packet transit services (type-1)

All of the incoming packets are transferred to the outgoing ports without IP termination. A typical example is a load balancing service.

2) Packet relay services (type-2)

As with a packet transit service, all of the packets are transferred to the outgoing ports, but the connections are terminated in the IP layer. Typical examples of this type of service are a proxy server and firewall server.

3) Connection termination services (type-3)

Packet flows are terminated in the TCP layer, and not all of the incoming packets go out. A typical example is a Web server.

IP service nodes provide type-1 and type-2 services. These nodes play a key role in the IP network because many packets pass through them. Therefore, the loads of the IP service nodes become very high. Considering that these services require complex packet processing, rather than layer 3 packet processing, there is a strong possibility that the IP service nodes will become a serious bottleneck in the network. Therefore, it is very important to improve their performance, and this requires the use of high-speed transit/relay packet processing.

Another important requirement is to maintain the current OS interface so that servers can be used in an open-standard environment; therefore, when we designed the new IP service node, we decided to make a new functional configuration without changing the OS kernel.

2.3 Development policy

Figure 2 shows the general packet processing functions. The IP additional function unit

between the device driver and TCP/IP unit performs IP header conversion and TCP connection status management without IP protocol termination. The hardware-based packet engine below the driver layer provides the network interface. This functional block, which plays an essential role in realizing high performance, is described below.

The IP header address translation of each packet is a basic function of the load balancing service. This function is performed by the IP additional function unit. A comparatively simple packet header translation and connection status control for all packets is executed almost in the same way as for other services in the IP addition module. Therefore, a basic requirement is to ensure that the IP additional unit has a high performance. To achieve this, we concentrate on improving the performance of the IP additional unit through hardware assistance; that is, we decided to off-load the function. If the unit can be constructed with a hardware packet engine, a big increase in performance can be expected. When some of the packets cannot be processed by the packet engine, we expect to obtain a high-performance by processing some of the packets passing through a TCP connection by hardware assistance. This method is called the "short-cut method" and is described in Section 3.2.

There are three ways to improve performance.[2)-4)] The first one, the most common method, uses a co-processor for exclusive packet level processing. This method enables easy and flexible implementation of complex functions, but because of the software processing it involves and the relatively slow growth in computing power that can be expected according Moore's law, it is too slow. The second method uses ASICs that are specially designed for use in an IP service node. This method can provide a high performance, but it is too inflexible. The last method uses a network processor. We believe that the network processor approach can provide the performance and flexibility needed for new service functions, because its packet-processing functions are written in software and can be implemented using hardware assistance.

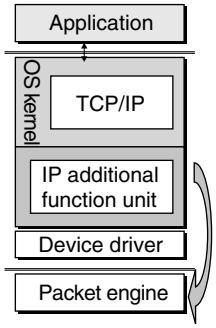

Figure 2
Packet processing functions in proposed IP service node.

3. System architecture

3.1 Function configuration

Figure 3 shows the proposed functional configuration. The packet engine performs packet header translation and management of the TCP status and provides the increase in processing speed. This translation and management is achieved by the network processor. The additional function unit in Figure 3 performs the remaining complex functions such as the analysis of XML tag information and URL addresses and also provides some of the functions needed to

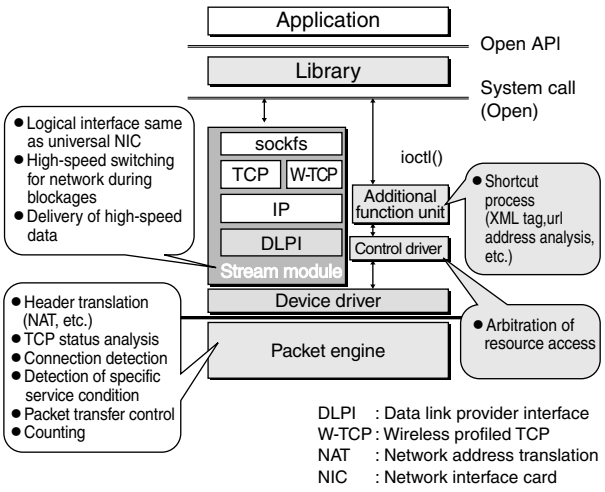

Figure 3
Functional configuration.

control a large storage.

In the case of the type-1 services described in Section 2.2, all packets can be processed by the packet engine block. On the other hand, in a type-2 service, some of the packets are transferred to the additional function unit from the packet engine and conventionally processed by software-based processing. Once the additional unit analyzes the data in the packets, the succeeding packets are processed solely by the packet engine, because they only contain user data.

3.2 Packet processing mechanism

Figure 4 shows the functional architecture and data flow of the proposed IP service node packet engine. The main task of the packet engine is fast packet transfer with packet classification, address translations, and buffer scheduling. To enhance the performance of the IP service node, the packet engine performs a relatively simple protocol, for example, layer 3 or layer 4, without using forwarding server processors. The server processors above the packet engine in Figure 3 perform relatively complex tasks that cannot be performed by the packet engine alone.

As shown in Figure 4, the IP service node has two tables: a policy table and a session table. The policy table contains information used to determine the packet handling methods such as the filtering rule and traffic control rule for load balancing. If the packet engine was able to input rules in the policy table, there could be an inconsistency in the total system. Therefore, the packet engine can reference the policy table but not write to it and the rules in the policy table are written by the server processors. The session table contains the session identifier, session status, and processes required to manage sessions, for example, address translation.

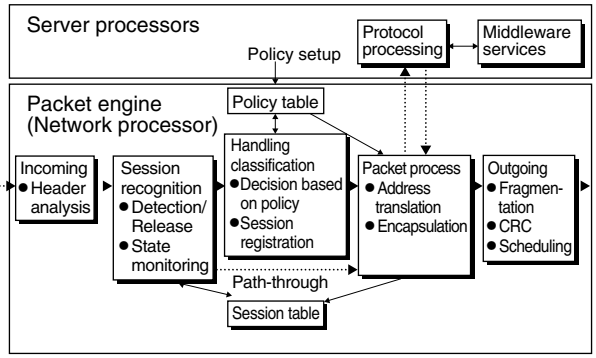

Figure 4
Functional blocks and data flow of packet engine.

We will now describe the functional blocks and packet flow in the packet engine. After checking its check sum value for validation, the session of an incoming packet is identified. That is, layer 3 and layer 4 header information such as the source/destination IP addresses and source/destination port numbers is used as a key to identify the session. Then, the session table is checked to see if the session has been registered or is a new one. If the session is new one, the policy table is accessed to determine how to handle the new session. Then, the selected process identifier is registered in the session table, unless the session has already been registered in the session table. If the process identifier for the session has already been registered, it is not necessary to pass the packet to the handling classification block.

After it has been determined how to handle the packet, it is forwarded to the packet processing block, where practical processes such as address translation and encapsulation are performed. If it is necessary to redirect the packet to the server processor, for example, at the beginning of an HTTP session, the packet is forwarded to the protocol processing block in the server processor, for example, to analyze the URL address. If the packet can be handled in the packet engine, for example, when the packet is for address translation, encapsulation, filtering, or the determination of the destination for load balancing, the packet is processed in this functional block without accessing the server processors. The final functional block is the outgoing block, where fragmentation, CRC, and scheduling are done.

The load of the server processors can be reduced by introducing the packet engine, which,

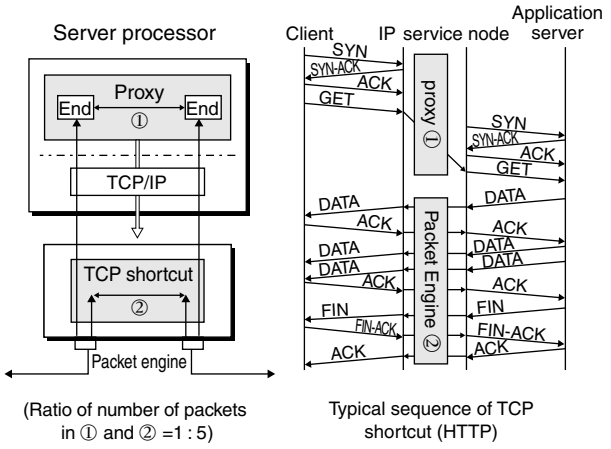

Figure 5
Shortcut method (for TCP).

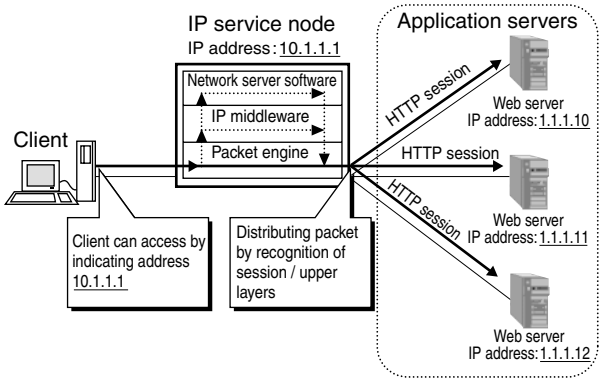

Figure 6
Performance evaluation system.

as described above, can handle all lower layer protocol procedures. **Figure 5** shows a connection level judgment processing called "shortcut processing." In URL load balancing, for example, after the network server receives a required URL address and analyzes it, the Web server address to which the bunch of packets should be sent is determined. The analysis is done in the IP additional function unit of the server processor. As a result, the software on the server processor does not participate any more. Usually, in Web services, about 20% of the packets are processed by the software unit, so a performance improvement of several times can be expected from load balancing. In general, the same method can be applied in proxy services, where the IP service node terminates the session instead of the IP host or application server. That is, after the server processor determines how to process the session, the packets, which consist of TCP connections, are handled in the packet engine.

4. Evaluation

4.1 Experimental system

To evaluate the system architecture we propose, we developed an experimental system based on a UNIX server. The PCI bus connects the server processors with the packet engine, which is realized by a network processor. The experimental IP service node system has two 1-Gigabit Ethernet interfaces.

We have built a network using the experimental IP service node system in order to evaluate how our architecture improves the performance. We evaluated the performance improvement in two types of services: the packet transit services and packet relay services described in Section 2.2. We assumed that the IP service node was being used to perform the typical task of balancing the load among multiple application servers (**Figure 6**). The client accesses the Web servers using a single IP address assigned to the IP service node. The IP service node recognizes the load of each application server to distribute traffic equally. The IP service node uses a WRR (Weighted Round Robin) algorithm for buffer scheduling. When a new session is established, the IP service node selects an application server and sets the address translation from the designated IP address and the IP address for the selected application server. Next, the packets of the session are forwarded based on the address translation by the packet engine without processing in the server processor of the IP service node. Therefore, the dominant factor in the packet forwarding performance is the performance of the network processor's packet engine and we can expect a significant improvement in this performance area.

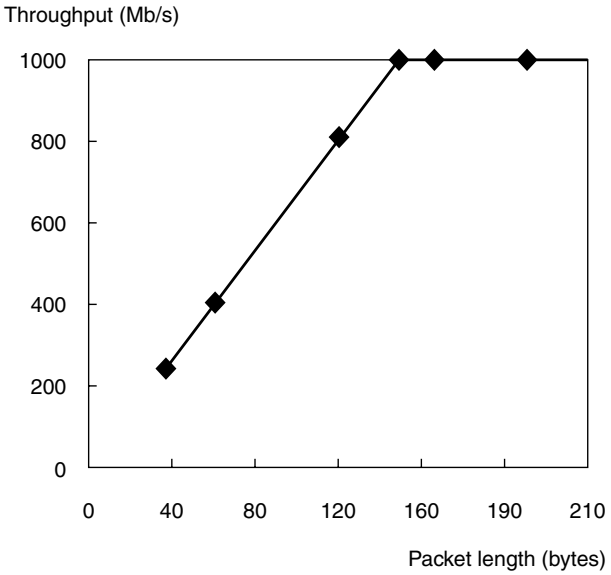

Figure 7
Performance evaluation result (1).

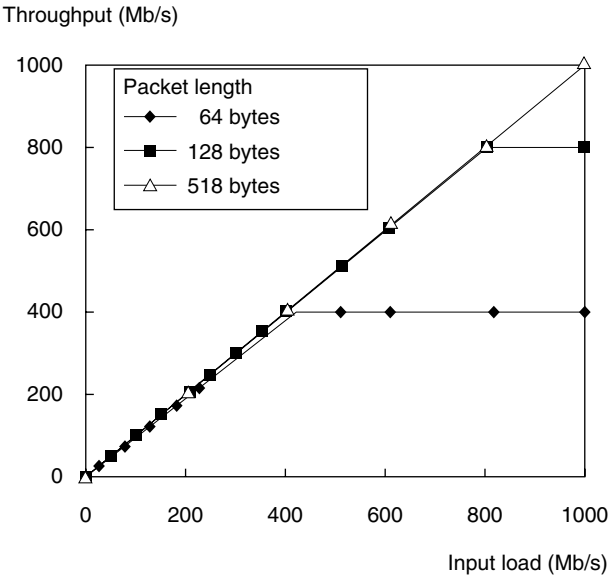

Figure 8
Performance evaluation result (2).

4.2 Performance evaluation

The packet forwarding evaluations for packet transit services are shown in **Figures 7** and **8**. These figures show the packet forwarding performance of a single interface whose throughput is 1 G bits/s during a session when packets are loaded by the packet generator. Note that these are not the performance figures during the session establishment phase, when other processes such as looking up the policy table are included in packet forwarding. In Figure 7, when the packet is long, the throughput is high. For example, when the packet length is 128 bytes (1024 bits), the throughput is 800 M bits/s, which means that the packet forwarding capability of the packet engine is constant at about 780 k packets/s. This performance is about 10 times better than the performance without a packet engine, when all packets are processed by software on the UNIX server's four 300 MHz SPARC64 CPUs. This result confirms that the proposed architecture improves the performance by reducing the load of the server processors. The upper bound of the throughput is 1 G bits/s due to the bit-rate limitation of the interface. In Figure 8, when the input load is high, the throughput is also high. However, if the packet is very short, the throughput is limited. For example, when the packet length is 128 bytes, the throughput is limited to 800 M bits/s. This is due to the performance limit (780 k packets/s) of the network processor packet engine.

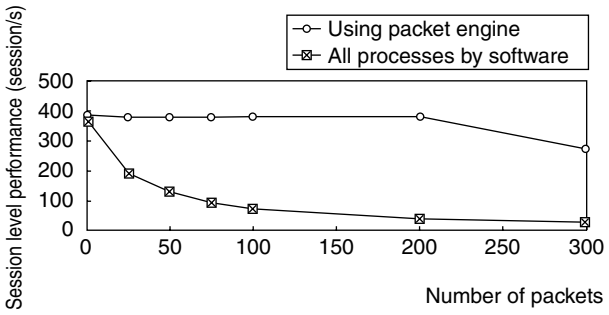

Figure 9
Evaluation of session-level performance in proxy services.

Figure 9 shows the evaluation results for session processing in a proxy type service. We compared the number of sessions that the IP service node can handle in two cases. In the first case, all processes were done by software on a server processor. In the second case, the packet engine was used after the server processor decided how to process the session. The evaluated application was a request and response by HTTP. The num-

ber of packets from the application server to the client varied from 1 to 300. When all processing was done by software, the number of sessions fell as the number of packets increased. For example, when the number of packets was 200, the number of sessions was less than 10% of the number when there was only one packet. However, when the packet engine was used, the number of sessions the IP service node could handle was roughly constant because the load on the server processor did not change. That is, when the number of packets was very small, introducing the packet engine made little difference to the session performance because the load on the server processor remained basically the same. However, when the number of packets was large, the packet engine greatly improved the session performance because it significantly reduced the load on the server processors.

5. Conclusion

An IP service node that processes not only layer 3 packet data but also the packet data of layers 4 to 7 in the OSI reference model will be a key component for controlling end-to-end user-requested services. The key technologies here are high-speed IP processing mechanisms and software-based service control functions. We proposed an IP service node architecture that closely connects a high-speed packet processing engine and software-based server functions within the IP service node to improve system performance. In the proposed architecture, the user packets are forwarded to a software-based server processor at session establishment to analyze the service requirements and the remaining packets of the session are handled in the same way as in high-speed IP processing.

To evaluate our IP service node architecture, we developed an experimental system based on UNIX servers connected to a network processor-based packet engine via the server's PCI busses. We confirmed that the packet level and the connection level functions performed well. We also confirmed that the transfer capability of the packet engine was constant at about 800 k packets/s, which is about 10 times the capability of a conventional server based on a general processor.

References

1) G. George: Telecosm? How Infinite Bandwidth will Revolutionize our World. 2000, Free Press.
2) P. Gupta, S. Lin, and N. McKeown: Routing lookups in hardware at memory access speeds. Proc. of IEEE INFOCOM'98, March 1998.
3) S. Keshav and R. Sharma: Issues and trends in router design. *IEEE Commun. Magazine*, pp.144-151 (May 1998).
4) A. K. Parekh and R. G. Gallager: A generalized processor sharing approach to flow control the single node case. Proc. of IEEE INFOCOM'92, pp.915-924, March 1992.

Tsuneo Katsuyama received the B.S. and M.S. degrees in Instrumentation Engineering from Keio University, Yokohama, Japan in 1974 and 1976, respectively. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1976 and has been engaged in research and development of communication systems/services and Internet middleware. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan and ACM.

Masafumi Katoh received the B.S. and M.S. degrees in Information Engineering from Yokohama National University, Yokohama, Japan in 1979 and 1981, respectively. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1981 and has been engaged in research and development of network architecture and traffic control of ISDN, ATM, and IP networks. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan.

Akira Hakata received the B.S. and M.S. degrees in Instrumentation Engineering from Keio University, Yokohama, Japan in 1977 and 1979, respectively.
He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1979 and has been engaged in research and development of broadband network systems.
Currently, he is a Vice General Manager of the Advanced Photonic Network Systems Development Div., Transport Systems Group. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan.

Akira Takeyama received the B.E. and M.E. degrees in Electrical Engineering from Keio University, Yokohama, Japan in 1975 and 1977, respectively.
He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1977 and has been engaged in research and development of network architecture and computer network systems.
He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan and the Information Processing Society of Japan. He received the OHM Technology Award in 1990.