

# Multi-Context-Aware Cache Accelerating Processing on Network Processors for Future Internet Traffic

Airi Akimura\*, Hiroaki Nishi\*\*

\*Science and Technology, Keio University, Japan

\*\* Science and Technology, Keio University, Japan

akimura@west.sd.keio.ac.jp, west@west.sd.keio.ac.jp

**Abstract**— The next-generation backbone routers need large-bandwidth communication capability because of increased future Voice over Internet Protocol (VoIP) services. Providing these or other future ubiquitous services will require new Network Processor (NP) designs, since these protocols or services are expected to require enormous increases in processing power in order to address their fine-grain communication pattern. These improvements of processing power will not be enough for future NP designs due to the requirements of cost reduction and lower power consumption. To meet these requirements, the cache mechanism has received increasing attention in NP design. Cache can reduce the load and time needed for table lookup by taking advantage of temporal localities in network traffic. For these reasons P-Gear, an advanced cache-based network processor, has been proposed by our laboratory. P-Gear, which provides a sophisticated cache architecture, can be used not only for table lookup but also for determining processing results, and achieves high-speed packet processing. In this paper, the cache architecture of P-Gear is shown and an extended cache mechanism called Multi-Context-Aware cache is proposed. Multi-Context-Aware cache can improve the speed of packet processing because it can control cache entry by analyzing the packet header and payload. To evaluate the validity of this proposed method, simulators of SIP (Session Initiation Protocol)-Aware cache, IKE (Internet Key Exchange)-Aware cache and FTP (File Transfer Protocol)-Aware cache were designed as subsets of Multi-Context-Aware cache. According to the simulation results, SIP-Aware, IKE-Aware and FTP-Aware cache can all make contributions to improving the cache hit rate by managing cache entry. In addition, they all can reduce processing unit requirements.

**Keywords**— Network Processor, Multi-Context-Aware Cache, VoIP, IKE, FTP

## I. INTRODUCTION

As a result of the continuing spread of the Internet, the amount of data streaming traffic over the Internet is growing. The next-generation backbone routers should meet the demands of large-bandwidth communication services caused by the further increase of Voice over Internet Protocol (VoIP) or by other future ubiquitous services. Network processors (NPs) for high-end router need to be designed to reduce both power consumption and the loads on packet processing

elements (PPEs) for processing this fine-grain network traffic effectively.

Network processors (NPs) are special devices optimized for packet processing. Table 1 lists the performance of high-end NPs that have been developed since 2005 [1], [2]. The internal architecture of these conventional NPs is different, but all of them contain many processing units (PUs) in order to achieve parallel processing. Most of contemporary high-end NPs are manufactured by the 0.13 $\mu$ m CMOS process. They run at about 2-350 MHz and consume about 15W. Their packet-processing throughput is about 20 Gbps. (These are mode values).

Today, the network infrastructure is mainly based on the 100-Mbps Ethernet. If the current trend toward an increase in Ethernet capacity were to continue, 100-Gbps Ethernet is expected to appear in near future. To attain a 100-Gbps packet-processing throughput by using traditional NPs, the number of processing units or operational frequency must be increased by about five times. However, these measures lead to certain issues in that they increase die size and power consumption, proportionally.

A new architectural approach to reduce power and increase throughput will therefore be required in this situation. Such an alternative architecture, which can realize high packet-

Table 1. LIST OF HIGH-END NETWORK PROCESSORS

| Product, (Vendor)         | Throughput, Frequency | Power, Process      | Number of PUs, (threads) |
|---------------------------|-----------------------|---------------------|--------------------------|
| SPP (Cisco)               | 40Gbps, 250MHz        | 35W, 0.13 $\mu$ m   | 188PUs                   |
| Toaster3 (Cisco)          | 10Gbps, 160MHz        | 14W, 0.18 $\mu$ m   | 16PUs                    |
| NPE10 (Internet Machines) | 20Gbps, 333MHz        | 18W, 0.13 $\mu$ m   | 64PUs                    |
| nP7510 (AMCC)             | 10Gbps, 350MHz        | 13W, 0.18 $\mu$ m   | 6PUs (144threads)        |
| nP37x0 (AMCC)             | 5Gbps, 700MHz         | 14w, 0.13 $\mu$ m   | 3PUs (72threads)         |
| X10q (Xelerated)          | 40Gbps, 200MHz        | 11W, 0.13 $\mu$ m   | 200PUs + 1110 PUs        |
| NP-1c (EZchip)            | 20Gbps, 250MHz        | 15W, 0.13 $\mu$ m   | 64PUs                    |
| iPP (Sili-con Acc.)       | 20Gbps, 333MHz        | 14.5W, 0.13 $\mu$ m | 32PUs (256 threads)      |
| IXP2800 (Intel)           | 10Gbps, 1400MHz       | 25W, 0.13 $\mu$ m   | 16PUs (128 threads)      |

processing throughput with low-power consumption, is presented in the following section.

To solve these problems, cache-based PPE architecture has received increasing attention as an approach to reduce power consumption and to increase throughput by taking advantage of temporal localities in network traffic. Here, “temporal locality” means the well-known tendency in network traffic whereby packets that have the same source and destination addresses appear repeatedly over a short period of time. Therefore the result(s) of processes such as table lookups and/or analyzing the packet header and payload can also be used for the follow-on packets that will need to be processed in the same way.

Several works have proposed improved caching algorithms and architectures, all of which use cache for data memory access, instruction memory access and/or for the result of routing-table lookups [3], [4], [5], [6], [7], [8]. Meanwhile, we proposed P-Gear as a cache-based PPE architecture that caches not only the result of routing-table lookups but also all the processing information necessary for packet processing. [9], [10]

In this paper, we propose a Process Learning Cache (PLC) as well as Multi-Context-Aware cache as an extension to the PLC. Multi-Context-Aware cache will require much less NP processing by controlling cache entries by analyzing both the packet header and the payload. Multi-Context-Aware cache prepares entries for the follow-on packets before they arrive or reduces unnecessary entries by predicting traffic behavior, depending on the first packet’s type. It can be easily predicted that Real-time Transport Protocol (RTP) packets come after Session Initiation Protocol (SIP) packets. Therefore Multi-Context-Aware cache prepares entries for RTP packets while processing the SIP packet and enables the follow-on RTP packets to partially skip processing in the NP. This paper discusses SIP-Aware, IKE (Internet Key Exchange)-Aware and FTP (File Transfer Protocol)-Aware cache as subsets of Multi-Context-Aware cache.

## II. ARCHITECTURE OF P-GEAR

This section focuses on the architecture of the cache-based PPE called P-Gear that can achieve a 100-Gbps throughput at 333 MHz Fig.1 shows the architecture of P-Gear. P-Gear consists of a bit-stream manipulation path called a burst-stream path (BSP) and a programmable processing-unit (PPU) array. A detailed feature description of engines other than the cache engine (C-Engine), which is the most related part for Multi-Context-Aware cache, have been omitted due to paper limitations. The major functions of BSP and the P-Engine follow:

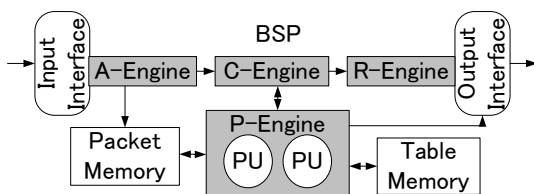


Figure 1. Architecture of P-Gear

**1) A-Engine:** The entrance part of the BSP is called the analysis engine (A-Engine). The main role of the A-Engine is the generation of a 432-bit-long information data token, created by analyzing the packet header. The token consists of three blocks of information defined as U-Info, A-Info, and E-Info. Each definition follows:

- The U-Info is the unique information for every packet. For example, its packet memory address. The length of the U-Info is 28 bits.
- The A-Info is information made by analyzing the packet header. A-Info includes the encoded major protocol type for each layer.
- The E-Info is information extracted from the packet header. For example, the source or destinations IP address.

**2) C-Engine:** The core of the BSP is called the C-Engine. The main roles of the C-Engine are adding a P-Info generated by P-Engine to the token, replacing the E-Info of the incoming token with the R-Info and sending the token to the R-Engine. The contents of P-Info and R-Info are as follows:

- The P-Info includes various processing results for a packet such as the next hop address and the filtering policy. P-Info also contains instructions about how to forward, modify and/or drop a packet at the R-Engine.
- The R-Info is rebuilding information for the packet header or additional information for encapsulation. Encapsulation is the mechanism that enables packets of a given protocol to be processed on network that use different communication protocols.

The C-Engine contains a cache memory called the process learning cache (PLC), which reduces the number of packets processed on the PUs in the following way: the packets that are not yet registered in the PLC (PLC-missed) are transferred to P-Engine for processing. If the packets are registered in the PLC (PLC-hit), they get the “processed token” from the PLC entry immediately and are transferred to the R-Engine. The PLC-hit or PLC-miss is judged by tags created by an A-Info/E-Info pair. In addition, a Cache-miss handler (CMH) in the C-Engine prevents exhaustion of PU resources. This means there can be duplicated PU assignments when succeeding PLC-missed tokens, which have the same A-Info/E-Info that are currently being processed at the PU, and have not yet been registered in the PLC make a new entry for the token. To avoid this duplication CMH has FIFO queues called cache miss queues (CMQs). Packets that have the same A-Info/E-Info are held in the same CMQ making the number of CMQs equivalent to the number of entries.

**3) R-Engine:** The back-end part of the BSP is called rebuilding engine (R-Engine). The main roles of the R-Engine are reassembling and modifying the original packet according to the token thereby making the outgoing packet.

**4) P-Engine:** P-Gear can adopt various kinds of architecture for its P-Engine. The main role of the processing engine (P-Engine) is generating P-Info and R-Info as processed tokens.

An easy way to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it.

### III. MULTI-CONTEXT-AWARE CACHE

We propose a Multi-Context-Aware cache as an extension to P-Gear, which can achieve high throughput with a lower number of PU cores and lower power consumption. This section discusses the concept of Multi-Context-Aware Cache.

#### A. Traffic Behavior Prediction

Normally P-Gear analyzes packets only by their header information, but Multi-Context-Aware cache can predict traffic behavior. As a result of the traffic behavior prediction, a reduction of unnecessary entries as well as preparation for follow-on packets is attained. This leads to an improved cache hit rate. The method for predicting the traffic behavior is as follows:

**1) Check for Layer 4:** Data in Layer 4 such as TCP or UDP offers valuable information for predicting traffic behavior. For instance, if the TCP header shows that the FIN bit is 1 and the ACK bit is 0, it means that this is the final packet of a packet stream from a certain source. In addition, the packet type can be found by checking the port number which is included in the TCP/UDP header.

**2) Check for Layer 5 through Layer 7:** Information in Layer 5 through Layer 7 is analyzed as a block. The payload in these layers indicates what kind of packet follows after the packet currently being read. This information can be used in two ways. First, the PLC entry for a packet that has no temporal locality is ineffectual and should not be created. For example, an IKE packet can be thought of as having no temporal locality because it is sent only once at the start of a secure connection. The IKE packet can be detected by checking its port number included in the TCP header. The port number of the IKE is well-known, 500, and is easy to detect. There are additional routing protocols that are regarded as having similar types of temporal access in which the interval of the query packet is comparatively longer. Secondly, the header information of an outgoing packet includes the header information of the follow-on/return packet which will form a pair. For example, in FTP, control packets accessing port 21 are transmitted and the data transfer port is assigned dynamically by these control packets. It means that the first and follow-on packets are paired. Therefore high-speed packet processing can be achieved by registering the processing information for the data packet in advance, during the processing of the control packets. The header information of the data packet can be predicted by analyzing the payload of the control packets.

#### B. SIP-Aware Cache

We propose SIP-Aware cache as a subset of Multi-Context-Aware cache. With the SIP-Aware cache engine, the SIP traffic behavior can be identified to forward packets more efficiently. VoIP is one of the Internet based telephone services that uses SIP for managing a session. As the number of users of IP phone services grow, the amount of VoIP packets will increase and SIP-Aware cache will become more efficient in this situation.

A SIP message has information about the follow-on packets, such as RTP. SIP-Aware cache not only achieves

high-speed packet processing of the media stream but also brings the following benefits:

a) Quality of service (QoS) in an emergency call (911 in the US, 119/110 in Japan) is improved: QoS in an emergency call has not yet been considered in SIP-based VoIP, but SIP-Aware cache enables improvements in the QoS by rewriting the RTP header, which is sent with the emergency call.

b) Entries are utilized more efficiently: The start and the end of a media stream are identified by analyzing the SIP message. For example, if the SIP method is INVITE, then a RTP packet follows after the SIP packet. A new PLC entry can be made according to the SIP message. If the SIP method is BYE, then the SIP packet is the end of the media stream. Therefore, invalidation of the PLC entry can be done according to the SIP message.

### IV. EVALUATION

We simulated SIP-Aware cache, FTP-Aware cache, and IKE-Aware cache to assess the validity of the Multi-Context-Aware cache. We used 100-Gbps network traffic created from a real network trace. In this section, we present evaluation methods and results.

#### A. Packet Traces

We utilized packet traces collected by WIDE (Widely Integrated Distributed Environment) where the IP address is not anonymized and additional VoIP packet traces, which we create virtually. We created additional virtual VoIP packets because VoIP packets collected by WIDE have no payload in order to preserve privacy and to reduce capture cost. We generated VoIP traffic according to a Poisson distribution for their number and their arrival timing. Their communication durations varied according to an Erlang distribution. In the future, we will call virtually-created VoIP packets as virtual VoIP traces.

In this experiment, the size of a SIP packet was assumed to be 100 bytes while RTP packets were assumed to be 214 bytes, G.711 codec telephony. We assumed 30 seconds and 15 seconds as the average duration of a session. Statistical data published in 2009 by the Japanese Ministry of Internal Affairs and Communications says that the average duration of telephone communication is 150-300 seconds.[11] These statistics indicate that using 15 or 30 seconds as the average duration of a communication is too short compared to the actual average duration of communications, as noted above. However, considering that the number of RTP packets will increase when the communication duration is longer, and that this will increase the PLC hit rate, our experiment should be a more severe assessment. In addition, we had to limit the duration time in order to limit the total run time of the Multi-Context-Aware cache simulator. The number of lines used in our experiment was 100, 1000 and 10000. Here, a "line" means a VoIP session. We utilized the following information from the packet traces obtained from WIDE: packet size, time stamp, source IP address, destination IP address, source port number, and destination port number.

## B. Simulator

This subsection focuses on the simulator of SIP-Aware cache. We designed an event-driven, time-based simulator. One PU was assigned to one packet for processing when the packet arrives. Then the PU was released at a termination time scheduled as 30  $\mu$ s after arrival. In case of a SIP packet, this termination time was scheduled as 130 $\mu$ s after arrival. Here, 30  $\mu$ s and 130  $\mu$ s were estimated processing times. Compared to 10.8  $\mu$ s, the average latency of processing packets in an Intel® IXP 2800, 30  $\mu$ s is just long enough or slightly more than needed to carry on all the complicated processing required for the SIP-Aware cache, even considering the required computation complexity. Processing the SIP packets takes more time because its payload is text-based, requiring an estimated 130  $\mu$ s.

Timing issues must be addressed in the simulation. In the simulated PU array, a PU is assigned to process each new packet. The algorithm to create a Cache Miss Queue (CMQ) for RTP starts with checking the packet arrival time.

There are two conditions: 1) a next/new packet has arrived before the last PU's scheduled termination time, and 2) the last PU's scheduled termination time has occurred before the arrival of the next/new packet.

In condition 1, the packet is processed in the following way:

- 1) Adjust the simulator time to the packet arrival time.
- 2) If the packet has been processed before and there's cache for this packet in the PLC, the state goes back to the start of the main loop, but if not, it checks whether packets with the same header are currently being processed or not.
- 3) If a packet with the same header is currently being processed, the state goes back to the start of the main loop, but if not, one PU is allocated to process the packet, and a CMQ is created for any follow-on packets with the same header. When the packet is a SIP, the termination time of the PU is postponed, a CMQ for RTP is created and the state goes back to the start.

In condition 2, the PU and CMQ are released in the following sequence:

- 1) Adjust the simulator time to the PU termination time.
- 2) Cache the information necessary for processing the follow-on packet.
- 3) Packets stacked on the CMQ are released.
- 4) If SIP packets are being processed by the PU, the information for processing the RTP is cached, and the CMQ created for the RTP is released. Then, the state goes back to the start.

## C. Experiment and Result

### 1) Experiment with virtual VoIP trace

First, we conducted experiments using only a VoIP trace. Tables 2 and 3 show the number of PUs required to process virtual VoIP traces when there is no packet types other than VoIP. The average number of lines represents the average number of VoIP sessions. Simulation of a traditional NP was

conducted by processing all packets on the PUs without using PLC. Therefore, we can compare the efficiency of a SIP-Aware cache based NP with a traditional NP. The SIP-Aware cache structure of a PLC is 4-way set associative and the number of entries per way in PLC is 2K, 4K and 8K.

By comparing data between Tables 2 and 3, it can be seen that there is no significant differences in the number of required PUs at different average communication durations. If packets are processed in the traditional way, 74 to 81 PUs are required for 1000 lines. This indicates that traditional NPs need double the number of PUs to process 1000 lines of VoIP packets. Traditional NPs which can support 40-Gbps half-duplex usually consist of 32 PUs. On the other hand, the NP with SIP-Aware cache is so effective in processing VoIP packets that it requires only 3-4 PUs. Tables 4 and 5 show that cache hit rates were high in this experiment.

However, when it comes to 10000 lines, more than 100 PUs are needed in NPs with 2K and 4K entry cache and the cache hit rate decreased markedly when using the 2K entry cache. This is because 8,192 entries, which is the maximum that the 2K entry cache on 4-way set associative can store, is too small to store the 20,000 entries (including the SIP) over the course of processing 10,000 lines and frequent rewriting of the entries decreased the cache hit rate. Meanwhile, the 8K entry cache can handle 32,768 entries which is sufficient to store cache from the 10,000 lines.

Tables 6 and 7 show the number of CMQ stacks. It was found that a large CMQ was not needed for processing virtual VoIP traces. This is because the intervals at which RTP packets arrive are long enough for the PU to finish its processing before

**Table 2.** THE NUMBER OF PUS WHEN THE AVERAGE COMMUNICATION DURATION WAS SET TO 15 SECONDS

| Average number of lines | Traditional NP | 2K  | 4K  | 8K |
|-------------------------|----------------|-----|-----|----|
| 100                     | 14             | 2   | 2   | 2  |
| 1000                    | 74             | 4   | 3   | 3  |
| 10000                   | 592            | 388 | 145 | 33 |

**Table 3.** THE NUMBER OF PUS WHEN THE AVERAGE COMMUNICATION DURATION WAS SET TO 30 SECONDS

| Average number of lines | Traditional NP | 2K  | 4K  | 8K |
|-------------------------|----------------|-----|-----|----|
| 100                     | 14             | 2   | 2   | 2  |
| 1000                    | 81             | 3   | 3   | 3  |
| 10000                   | 575            | 389 | 143 | 33 |

**Table 4.** CACHE HIT RATE WHEN THE AVERAGE COMMUNICATION DURATION WAS SET TO 15 SECONDS

| Average number of lines | 2K    | 4K    | 8K    |
|-------------------------|-------|-------|-------|
| 100                     | 99.7% | 99.8% | 99.8% |
| 1000                    | 99.7% | 99.8% | 99.8% |
| 10000                   | 37.5% | 80.8% | 97.0% |

**Table 5.** CACHE HIT RATE WHEN THE AVERAGE COMMUNICATION DURATION WAS SET TO 30 SECONDS

| Average number of lines | 2K    | 4K    | 8K    |
|-------------------------|-------|-------|-------|
| 100                     | 99.8% | 99.9% | 99.9% |
| 1000                    | 99.7% | 99.9% | 99.9% |
| 10000                   | 39.5% | 82.2% | 97.5% |

the next RTP packet arrives.

## 2) Experiment with a mixture of WIDE trace and virtual VoIP trace

Next, we conducted experiments using WIDE traces and virtual VoIP traces. Traces, which were collected from WIDE at two different times, are referred to as WIDE2005 and WIDE2007, respectively. Each of the two traces is an all-day trace collected as 96 assorted 15-minute trace segments on January 7, 2005 and January 10, 2007 (24 hours, each). WIDE2005 included 642,387,274 packets and WIDE2007 included 412,672,547 packets. VoIP packets for 10000 lines were created virtually with the average communication duration assumed to be 15 seconds, as previously discussed.

Although packet traces collected from WIDE were streaming at 100 Mbps, we simulated 96-Gbps traffic by streaming 96 15-minute traces simultaneously and by shrinking the time scale to one-tenth of their original trace in this experiment. This simulated congested traffic as one of the future Internet environments where traffic streaming over end-host gateway is ten times faster and the number of users is 96 times greater than now.

Tables 8 and 9 show the number of PUs needed to process the mixed traces of WIDE2005 and virtual VoIP traces as well as the mixed traces of WIDE2007 and virtual VoIP traces. There was a small decrease in the increment of the number of PUs needed to process the VoIP-containing packets as cache increased. Additionally, the reason why WIDE2007 needs more PUs to be processed than WIDE2005 is that the temporal locality in WIDE2007 is smaller than that in WIDE2005.

Tables 10 and 11 show the cache hit rates. With SIP-

Aware cache, VoIP-containing packets yield a 1-3% decrease in cache-hit rate because of the RTP packet interval previously discussed.

Tables 12 and 13 show the number of CMQ stacks. These results indicate that VoIP packets have no effect on the number of CMQ stacks because the intervals at which VoIP packets come are long enough not to consume the entire CMQ. Additionally, Table 14 shows that the number of PUs and cache hit rate when the entries prepared for RTP are set to be held in SIP-Aware cache as much as possible. Although the number of PUs normally decreases with an increase in cache hit rate, in this case, the number of PUs decreased even when the cache hit rate decreased. This is because the number of entries for other protocols decreases, since entries for RTP are preferentially held in cache, which reduces the number of PUs as well as the load on the PUs. Cache hit rate for virtual VoIP is low, 0.08% in P-Gear without SIP-Aware cache since entries for RTP are rewritten many times for reasons similar to those described above. This means that the caching mechanism of P-Gear without SIP-Aware cache would not work well to process RTP.

## 3) Experiment with the packets of other protocol

Finally, we carried out experiments using packets of other protocols, IKE and FTP. Packet traces for these experiments were also collected by WIDE. An IKE packet is sent only once at the start of a secure connection. This means that some of the unnecessary rewrites can be prevented by not creating entries for the IKE packet when it comes. Table 15 shows the hit rate of this IKE-Aware cache. Although the number of PUs and CMQ stacks remained the same, the cache hit rate increased slightly. Considering that, in the future, the security of communication will become more important and the amount of

**Table 6.** THE NUMBER OF CMQ STACKS WHEN THE AVERAGE COMMUNICATION DURATION WAS SET TO 15 SECONDS

| Average number of lines | 2K | 4K | 8K |
|-------------------------|----|----|----|
| 100                     | 1  | 1  | 1  |
| 1000                    | 1  | 1  | 1  |
| 10000                   | 2  | 2  | 2  |

**Table 7.** THE NUMBER OF CMQ STACKS WHEN THE AVERAGE COMMUNICATION DURATION WAS SET TO 30 SECONDS

| Average number of lines | 2K | 4K | 8K |
|-------------------------|----|----|----|
| 100                     | 1  | 1  | 1  |
| 1000                    | 1  | 1  | 1  |
| 10000                   | 2  | 2  | 2  |

**Table 8.** THE NUMBER OF PUS NEEDED FOR VOIP-CONTAINING AND NO-VOIP WIDE2005 TRACE

| Trace           | Traditional NP | 2K  | 4K  | 8K  |
|-----------------|----------------|-----|-----|-----|
| No-VoIP         | 364            | 129 | 116 | 103 |
| VoIP-containing | 373            | 166 | 152 | 125 |
| The increment   | 11             | 37  | 36  | 22  |

**Table 9.** THE NUMBER OF PUS NEEDED FOR VOIP-CONTAINING AND NO-VOIP WIDE2007 TRACE

| Trace           | Traditional NP | 2K  | 4K  | 8K  |
|-----------------|----------------|-----|-----|-----|
| No-VoIP         | 573            | 217 | 186 | 153 |
| VoIP-containing | 591            | 250 | 214 | 176 |
| The increment   | 18             | 33  | 28  | 23  |

**Table 10.** CACHE HIT RATE (WIDE2005)

| Trace           | 2K    | 4K    | 8K    |
|-----------------|-------|-------|-------|
| No-VoIP         | 70.7% | 74.6% | 79.7% |
| VoIP-containing | 67.5% | 71.9% | 77.2% |

**Table 11.** CACHE HIT RATE (WIDE2007)

| Trace           | 2K    | 4K    | 8K    |
|-----------------|-------|-------|-------|
| No-VoIP         | 67.3% | 73.7% | 80.0% |
| VoIP-containing | 65.8% | 72.0% | 78.3% |

**Table 12.** THE NUMBER OF CMQ STACKS (WIDE2005)

| Trace           | 2K  | 4K  | 8K  |
|-----------------|-----|-----|-----|
| No-VoIP         | 137 | 132 | 137 |
| VoIP-containing | 137 | 131 | 124 |

**Table 13.** THE NUMBER OF CMQ STACKS (WIDE2007)

| Trace           | 2K | 4K | 8K |
|-----------------|----|----|----|
| No-VoIP         | 44 | 44 | 44 |
| VoIP-containing | 44 | 44 | 44 |

**Table 14.** THE NUMBER OF PUS AND CACHE HIT RATE (RTP HOLD)

|                             | P-Gear | SIP-Aware |
|-----------------------------|--------|-----------|
| The number of Pus           | 193    | 174       |
| Cache hit rate              | 74.50% | 74.12%    |
| Cache hit rate of VoIP only | 0.08%  | 99.3%     |

key-exchange protocol packets will increase, the hit rate of the IKE-Aware cache will be improved.

We also propose two methods for creating entries in the FTP-Aware cache; a static FTP-Aware cache and a dynamic FTP-Aware cache. When using Static FTP-Aware cache the PLC creates a cache entry with its port number set to 20 when a control packet arrives. On the other hand, when using dynamic FTP-Aware cache, the PLC creates its cache entry for the data stream by analyzing the payload of a control packet from port 21. In the dynamic FTP-Aware cache, the port number is not fixed at 20, but is dynamically changed by the PLC based on information in the FTP control packet. Cache hit rates of the static and dynamic FTP-Aware caches are shown in Table 16. The hit rate of the Static FTP-Aware cache is lower than in the normal P-Gear. This is because cache entries created by using only port 20, the normal data transfer port, may or may not be used since the FTP data transfer port can be changed dynamically. Although the port number actually used can be detected by analyzing the payload of the FTP control packet, it is impossible to conduct this analysis since traces used for this simulation have no payload. Therefore, the static FTP-Aware cache, if it is used in this condition, shows the same performance as the dynamic FTP-Aware cache, to which PLC create/inject an entry by predicting the actual data port number when it is used with real network traces. From the results shown in Table 16, it can be seen that the cache hit rate is improved

**Table 15.** CACHE HIT RATE (IKE-AWARE)

| P-Gear | IKE-Aware |
|--------|-----------|
| 72.96% | 72.98%    |

**Table 16.** CACHE HIT RATE (FTP-AWARE)

| P-Gear | Static FTP-Aware | Dynamic FTP-Aware |
|--------|------------------|-------------------|
| 72.96% | 71.15%           | 79.42%            |

using dynamic FTP-Aware cache.

## V. CONCLUSION

Existing high-end NPs have achieved high throughput only by increasing the number of PUs in order to execute parallel processing. Typical NPs are manufactured by the 0.13 $\mu$ m CMOS process, run at about 2-350 MHz and consume about 15W. Since it is hard to reduce their power consumption any further by micro-fabrication techniques alone, a new architectural approach to reduce power and increase throughput will be required.

P-Gear is a cache-based PPE that can reduce power consumption and increase throughput by exploiting temporal localities in network traffic. P-Gear has a bit-stream manipulation path called a BSP and a programmable processing-unit array called P-Engine. The BSP consists of the A-Engine, which analyzes incoming packets and generates tokens, the C-Engine, which adds processing results and information about how to forward packets to the tokens referencing PLC, and the R-Engine, which reassembles and modifies outgoing packets with the original

packet information by referencing the token. The PLC caches the packet processing method, with all result, and enables most packets to skip processing at the PU array in the P-Engine.

In this paper, we proposed Multi-Context-Aware cache as an extension to P-Gear, which can lower the processing load on the PUs by reducing unnecessary cache entries and preparing entries for the follow-on packets before they arrive. This control is based on traffic behavior prediction, which can be attained by analyzing the packet header and payload. The cache hit rate and the QoS are improved by detecting this traffic behavior.

To assess the validity of Multi-ContextAware cache, we simulated SIP-Aware cache, IKE-Aware cache and FTP-Aware cache. The results show that SIP-Aware cache decreases the number of PUs needed for packet processing. In addition, IKE-Aware cache and FTP-Aware cache both improve cache hit rates. Considering that the current high-end NPs consist largely of PUs, SIP-Aware cache could be one of the solutions to reduce hardware cost.

## ACKNOWLEDGEMENT

The authors thank the National Institute of Information and Communication Technology and the Ministry of Internal Affairs and Communications, Japan for their support to this research. The authors also thank John Helwig for his editing suggestions; however, retain responsibility for any errors.

## REFERENCES

- [1] Linley Gwennap and Bob Wheeler. *A Guide to NETWORK PROCESSORS*. The Linley Group, Fifth edition, 2003.
- [2] Peter N. Glaskowsky. Toaster3 Pops Up at MPF. In *Microprocessor Report*, Oct 16 2002.
- [3] Bryan Talbot, Timothy Sherwood, and Bill Lin. IP Caching for Terabit Speed Routers. In *Global Communications Conference (GlobeCom'99)*, pages 1565–1569, 1999.
- [4] Tzi cker Chiueh and Prashant Pradhan. High-Performance IP Routing Table Lookup Using CPU Caching. In *INFOCOM'99*, pages 1421–1428, 1999.
- [5] Kari Seppanen. Novel IP Address Lookup Algorithm for Inexpensive Hardware Implementation. *WSEAS Transaction on Communications*, 1(1):76–84, 2002.
- [6] Pankaj Gupta, Steven Lin, and Nick McKeown. Routing Lookups in Hardware at Memory Access Speeds. In *INFOCOM'98*, pages 1240 – 1247, 1998.
- [7] I.L. Chvets and M.H. MacGregor. Multi-zone Caches for Accelerating IP Routing Table Lookups. In *High Performance Switching and Routing (HPSR) 2002*, 2002.
- [8] Zhen Liu, Jia Yu, Ziaojun Wang, Bin Liu, and Laxmi Bhuyan. Revisiting the Cache Effect on Multicore Multithreaded Network Processors. In *11<sup>th</sup> Euromicro Conference on Digital System Design Architecture, Methods and Tools*, pages 317–324 2008.
- [9] Okuno, M. and Nishi, H.: Network Processor Accelerator Using Temporal Locality of Traffic (in Japanese), *IPSJ Trans. Advanced Computing System*, Vol.45, No.SIG 6, pp.45-53 (2004).
- [10] Okuno, M., Ishida, S. and Nishi, H.: Low-Power Network-Packet-Processing Architecture Using Process-Learning Cache for High-End Backbone Router, *IEICE Trans. ELECTRON.*, Vol.E88-C, pp.536-543 (2005)
- [11] Ministry of Internal Affairs and Communications, State of Japan's Communications in Terms of Traffic [http://www.soumu.go.jp/main\\_sosiki/joho\\_tsusin/eng/Statistics/pdf/090216\\_2.pdf](http://www.soumu.go.jp/main_sosiki/joho_tsusin/eng/Statistics/pdf/090216_2.pdf)
- [12] Peter N. Glaskowsky. Toaster3 Pops Up at MPF. *Microprocessor Report*, Oct. 16, 2002.