P2PNIC: High-Speed Packet Forwarding by Direct Communication between NICs

Yukito Ueno
The University of Tokyo / NTT Communications
Tokyo, Japan
eden@g.ecc.u-tokyo.ac.jp

Ryo Nakamura
The University of Tokyo
Tokyo, Japan
upa@nc.u-tokyo.ac.jp

Yohei Kuga
The University of Tokyo
Tokyo, Japan
sora@nc.u-tokyo.ac.jp

Hiroshi Esaki
The University of Tokyo
Tokyo, Japan
hiroshi@wide.ad.jp

***Abstract*—Against the background of the contiguous growth of the Internet and data center traffic, the performance requirement for software middleboxes is increasing rapidly. Although their performance has been improved by continuous research and development, their packet forwarding architecture depends on the CPU and the main memory. Thus, their throughput cannot exceed the limit of the memory bandwidth, for instance. To address these limits, we propose a novel packet forwarding architecture called P2PNIC. In P2PNIC, a NIC directly communicates with other NICs through the PCIe interconnect without CPU and main memory involvement, like the inter-linecard communication in a hardware router. To show the feasibility and the performance advantages of P2PNIC architecture, we implemented P2PNIC on a programmable 40 GbE NIC and compared the throughput and latency with TestPMD, which is an application of DPDK. The evaluation shows that P2PNIC achieves 40.37 Mpps for 64-byte packets, which is 1.45 times higher than TestPMD. In addition, P2PNIC shows 36% lower latency than TestPMD for 64-byte packets with 1 Gbps background traffic. The P2PNIC architecture accelerates packet forwarding on a general-purpose server and advances software-based network technologies.**

***Index Terms*—Ethernet NIC, packet forwarding, PCIe, Peer-to-Peer DMA**

I. Introduction

The rapid and contiguous growth of the traffic volume in career backbone and data center demand high-speed routers in both hardware and software. In commercial networks, hardware routers have been used as an inevitable choice that can achieve the required throughput. Hardware routers are composed of multiple boards with integrated Ethernet ports called linecards, connected by a switching fabric. In general, they can accommodate 32–48 100 Gbps ports in the wire-rate. On the other hand, nowadays, software middleboxes have increasingly been used [1], [2] for new applications such as network virtualization and cloud services. The throughput of software middleboxes can accommodate several 100 Gbps ports in the wire-rate [3].

Although the performance of software middleboxes has been improved by continuous research and development [4]–[6], their throughput and latency have not reached the grade of hardware routers yet. Software middleboxes run on a general-purpose server; the host CPU processes all packets, and thus the packets must be placed on the main memory at least once,

This work was supported by JSPS KAKENHI Grant Number JP20K19781.

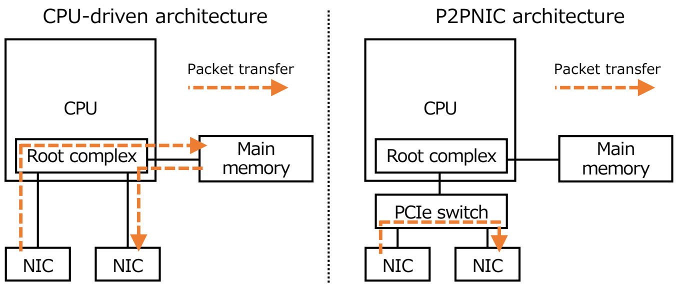

Fig. 1. Comparison of current packet forwarding architecture and P2PNIC.

as shown on the left-hand side of Figure 1. We refer to this way as CPU-driven architecture. As an example of potential limits in this architecture, the bandwidth of the main memory could limit the throughput of packet forwarding. The current DRAM (DDR4 SDRAM PC4-34100) for servers can provide a bandwidth of 34.1 GB/s, which is insufficient for accommodating more than two 100 Gbps ports with two memory channels and more than five 100 Gbps ports with four memory channels. Therefore, this approach cannot accommodate a higher-speed or a greater number of ports.

To address these potential limits, we propose a novel architecture called P2PNIC, where Ethernet NIC itself drives packet forwarding without the involvement of CPU and main memory on a general-purpose server. In this architecture, each NIC behaves like a linecard in hardware routers; each NIC transfers packets to the opposite NIC by issuing Direct Memory Access (DMA) for the packet data, as shown on the right-hand side of Figure 1. The P2PNIC architecture can address the potential limits of CPU-driven architecture because its throughput is not affected by the main memory and CPU capacity. Instead of the main memory and CPU, the PCIe bandwidth between NICs is dominant for the throughput in P2PNIC. With a single PCIe switch, P2PNIC can provide an aggregated bandwidth that is sufficient for 12 100 Gbps ports, while the throughput can be scaled easily by optimizing the PCIe composition. In this work, we have implemented and evaluated the P2PNIC architecture with some variants on SmartNICs [7]. The evaluation shows 1.45 times higher throughput and 36% lower latency compared with a DPDK application, which represents the performance of CPU-driven architecture. The contributions of this paper are as follows:

- We propose a novel architecture called P2PNIC, where a NIC directly communicates with other NICs for fast packet forwarding inside a general-purpose server.
- We demonstrate the feasibility of the P2PNIC architecture by implementing prototypes on SmartNICs, and the evaluation shows that the implementation has 1.45 times higher throughput and 36% lower latency compared with a DPDK application.

II. Related Work and Problem Description

Despite the continuous improvements to date, the current packet forwarding architecture on general-purpose servers cannot exceed the performance of the hardware routers. In career backbone networks, the conventional approach to achieve high-performance packet forwarding is to use dedicated ASIC chips and fabrics in hardware equipment [8]. In ASIC-based hardware routers, each linecard communicates with other linecards independently. The distributed communication form contributes to the scalability of the total throughput in the equipment. As a result, the throughput capacity of hardware routers is sufficient for 32–64 of 100G ports, which is adequate for the demand in today's career backbone and data centers.

On the other hand, software middleboxes require high-performance packet forwarding on a general-purpose server. The software middleboxes process all packets by the host CPU, as shown on the left side of Figure 1, and thus we refer to this as CPU-driven architecture. Here, the CPU sends/receives packets to/from NICs through the PCIe interconnect. In this architecture, a variety of methods have been proposed [4], [9], [10] to improve the performance of packet forwarding. In particular, with the recently proposed high-speed packet I/O technologies [5], [6], the packet forwarding performance of the CPU-driven architecture has been dramatically improved.

However, the evolution of network link speed is revealing potential limits in the CPU-driven architecture, which is caused by the unavoidable involvement of the CPU and main memory in packet forwarding. For example, the current memory (DDR4 SDRAM PC4-34100) cannot provide sufficient bandwidth for more than two 100 Gbps ports with two memory channels and five 100 Gbps ports with four memory channels.

Moreover, the processing capacity of the CPU would be the next bottleneck for packet forwarding in this architecture. By using the current top-grade CPU with 28 cores, accommodating a single 100 Gbps port requires more than two CPU cores [3]. In this case, the CPU can accommodate 14 100 Gbps ports at most; thus, accommodating 48 100 Gbps ports, which is common in hardware routers, requires more than 96 CPU cores. However, nowadays, it is not easy to increase the processing capacity of a single CPU core and the number of CPU cores [11]. As another consideration, the number of PCIe lanes a root complex in a CPU has could be a bottleneck in this architecture. For example, Intel Xeon CPU has 64 PCIe lanes at most, which is equivalent to only four 100 Gbps ports. In summary, the bandwidth of the main memory and the processing capacity of the CPU would limit the potential performance of the CPU-driven architecture.

III. Proposed Method

To address the potential limits of the CPU-driven packet forwarding architecture, we propose a novel packet forwarding architecture called P2PNIC. The P2PNIC architecture is NIC-driven; a NIC directly transfers packets to other NICs, like the inter-linecard communication in a hardware router, as illustrated on the right-hand side of Figure 1. In general, the NIC transfers packets to main memory over PCIe interconnect. The communication between the NIC and the main memory is called Direct Memory Access (DMA). On the other hand, for the specification, the target memory region of DMA is not limited to the main memory, as long as it is mapped into the physical address space. By exploiting this specification, in P2PNIC architecture, the NIC directly communicates with other NICs by issuing DMA destined for the memory region on other NICs. The form of DMA is called Peer-to-Peer DMA (P2P DMA) [12], [13], and the P2PNIC architecture is the first scheme that adopts the P2P DMA between Ethernet NICs to accelerate packet forwarding.

By adopting P2PNIC architecture, we can overcome the potential limits of the CPU-driven architecture because, in P2PNIC architecture, the throughput is not affected by the bandwidth of main memory and the processing capacity of the CPU. In addition, P2PNIC reduces latency by halving the number of DMAs. The CPU-driven architecture requires DMA to be issued twice to forward a packet; a NIC that received a packet issues DMA Write to transfer the packet to the main memory, and another NIC to send the packet issues DMA Read to retrieve the packet from the main memory. In contrast to the CPU-driven architecture, the P2PNIC architecture requires a single DMA to forward a packet; in one of the variants of P2PNIC, the NIC that received a packet issues DMA Write to transfer the packet to the opposite NIC.

In P2PNIC, the throughput capacity of packet forwarding can be scaled by optimizing the composition of PCIe with PCIe switches and the root complex of the CPU. For example, by connecting two PCIe switches through the root complex, the achievable throughput between the NICs in the same PCIe switches can be doubled, although the root complex constrains the inter-PCIe-switch traffic. As an example of components, the current highest grade PCIe switch has 96 lanes [14], which can provide an aggregated bandwidth that can accommodate 12 100 Gbps ports. The root complex in a CPU can also be a component; the current highest grade has 128 lanes [15] and can provide an aggregated bandwidth equivalent to 16 100 Gbps ports.

A. Difference between CPU-driven and P2PNIC architecture

A common NIC manages packet I/O through the queues allocated on the main memory, as shown on the left-hand side of Figure 2. The common NIC has Ethernet ports, PCIe endpoint, and small internal memory for buffering packets as its components. The NIC receives packets from its Ethernet

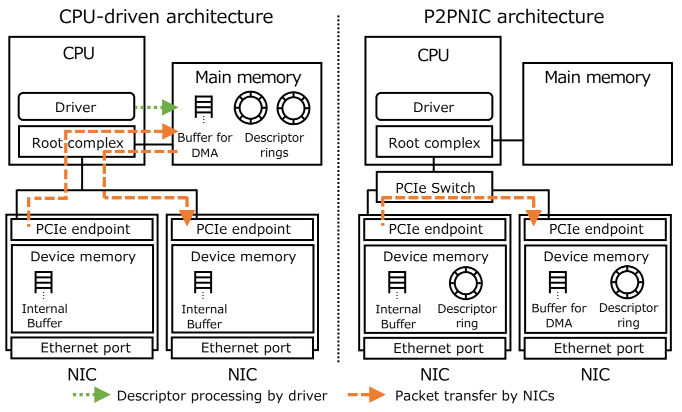

Fig. 2. Components and behavior of CPU-driven and P2PNIC architecture.

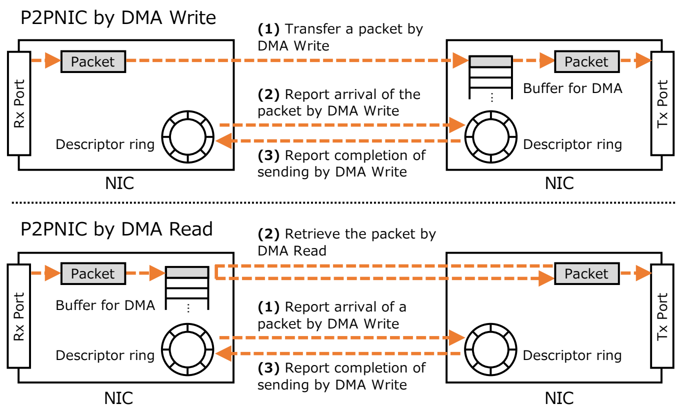
P2PNIC by DMA Write
(1) Transfer a packet by DMA Write
(2) Report arrival of the packet by DMA Write
(3) Report completion of sending by DMA Write
Rx Port
Packet
Descriptor ring
Buffer for DMA
Tx Port
NIC
P2PNIC by DMA Read
(2) Retrieve the packet by DMA Read
(1) Report arrival of a packet by DMA Write
(3) Report completion of sending by DMA Write

Fig. 3. NIC-to-NIC packet I/O in P2PNIC with DMA Read and Write.

port and transfers the packets to the main memory through the PCIe interconnect. Conversely, the NIC sends packets to its Ethernet port after retrieving the packets from the main memory through the PCIe interconnect. For the packet transfer between CPU and NICs, a ring buffer is used [16]. Each element of the ring buffer contains the metadata that describes a single packet; the element is called a descriptor, and the entire ring buffer is called a descriptor ring. The descriptor ring for CPU-NIC communication is placed on the main memory and updated by both host CPU and NIC.

The driver, the software to manage NICs, is required for packet transfer between CPU and NIC. The driver initializes the NIC by informing it of various parameters through the configuration register. The driver also manages the descriptor ring. The basic procedure of the communication between CPU and NIC is as follows. Prior to receiving the packet, the driver informs the NIC of the memory address for the packet buffer on the main memory through the descriptor ring. Then when the NIC receives a packet, the NIC issues DMA Write to transfer the packet to the memory region. On the other hand, for sending, the driver informs the NIC of the memory address for the packet buffer on the main memory. Then the NIC issues DMA Read to retrieve the packets and sends them.

In P2PNIC, the NIC processes packet forwarding with the descriptor ring and the DMA buffer, which are not on the main memory as with the CPU-driven architecture but on the device memory inside the NIC, as shown on the right-hand side of Figure 2. The NIC transfers packets to the device memory of the opposite NIC through the PCIe interconnect. P2PNIC adopts the descriptor ring for the inter-NIC communication as with common NICs. The descriptor ring has to be placed on the device memory of each NIC to achieve direct packet transfer between NICs without main memory involvement. In addition to the descriptor ring, the corresponding packet buffer has also to be placed on the device memory of each NIC. Therefore, the NIC must have sufficient memory for the descriptor ring and packet buffer.

The driver of P2PNIC initializes each NIC as with common NICs but is not involved in the packet I/O. To communicate with other NICs, the NIC has to know the physical addresses of the descriptor ring and packet buffer of the opposite NIC. For the requirement, the driver of P2PNIC informs each NIC of the physical addresses in its initialization. After the initialization, the driver of P2PNIC does not process packet forwarding because each NIC drives it.

B. *Methods for NIC-to-NIC packet I/O*

In P2PNIC, the packet transfer between NICs can be DMA Read or Write, as shown in Figure 3. In the P2PNIC with DMA Write (P2PNIC-Wr), the NIC that received a packet issues DMA Write to transfer the packet to the device memory on the opposite NIC. After the transfer, the NIC updates the descriptor ring on the opposite NIC by DMA Write to report the transferred packet. The opposite NIC sends the packet and then reports the completion of sending through the descriptor ring by DMA Write. The procedure is similar to the CPU-driven architecture except that its direction is from NIC to NIC. Besides, P2PNIC can also be achieved based on DMA Read. In P2PNIC with DMA Read (P2PNIC-Rd), the NIC that received a packet reports the packet's arrival to the opposite NIC at first. Upon receiving the report, the opposite NIC issues DMA Read to retrieve the packet from the NIC. After the completion of DMA, the opposite NIC sends the packet and reports the completion of sending through the descriptor ring.

Because there is a possibility of packet corruption due to the DMA failure in P2PNIC, the NICs compute a checksum for each packet before and after the DMA. PCIe adopts packet-based communication, and its packets are called Transaction Layer Packets (TLPs). In PCIe packet transfer between CPU and PCIe devices, TLP drop does not occur because the required bandwidth is ensured between them. However, in DMA between NICs, the required bandwidth is not always ensured across the entire path between the devices. If data congestion occurs at a component (e.g., PCIe switch) on the communication path, the component may drop TLPs, even though the retransmission mechanism is provided on the links between the components. By using the checksum mechanism, the NIC can detect if the packet is corrupted due to DMA failure and drop it.

IV. Implementation

We implemented P2PNIC with Netronome Agilio LX 2x40GbE NICs [7] to show the feasibility and the performance compared with the applications of CPU-driven architecture in the 40 Gbps range. This NIC provides software programmability as its firmware and has a network processing unit (NPU) called NFP-6000 and two 40 Gbps Ethernet ports. We used this NIC because it has the capacity to process high-volume traffic inline and provides fine-grained programmability on DMA over PCIe; both are required to prototype P2PNIC.

There are two possible ways to achieve P2PNIC, as mentioned in Section III-B: DMA Write and Read. We implemented both of the methods, named P2PNIC-Wr and P2PNIC-Rd, as separated firmware to demonstrate their feasibility and performance. In addition, we prepared a modified version firmware of P2PNIC named P2PNIC-Bn that uses CPU to process the descriptor ring and main memory to transfer packets. The purpose of this firmware is to observe the impact on the performance by involving CPU and main memory compared with P2PNIC-Wr and P2PNIC-Rd, without changing the implementation as far as possible. For every P2PNIC-based method, we implemented the IP routing feature, including the longest prefix matching with the state-jump table [17], while it can be removed by a build option depending on the measurement condition.

To manage the NICs and inform them of the necessary parameters at initialization, we also implemented a dedicated driver in the user space of the Linux OS. The driver performs the initialization for each NIC, which includes enabling the PCIe bus mastering, gathering physical addresses of the descriptor rings and DMA buffers of installed NICs, and informing each NIC of the physical addresses. After the initialization, the driver does not process packet forwarding because each NIC drives that. Although this driver is implemented in the user space of the Linux OS, it can perform the same function as a kernel-space driver by using the VFIO [18] mechanism.

The checksum calculation against the packet data to detect DMA failure is a mandatory feature for P2PNIC implementation. However, implementing the checksum mechanism as a part of the firmware increases the processing load in proportion to the packet size. To minimize the impact of the checksum calculation on the performance, we calculate the checksum using only a representative four bytes for every configurable number of bytes, which has to be smaller than the size into which TLPs can be divided and dropped (64 bytes for DMA Read and 128 bytes for Write in typical systems). For the measurement we conduct in Section V, we use 64 as the configurable number of bytes for both DMA Read and Write to align the parameter with the worst case.

Moreover, to optimize the throughput of the implementations, we adopt the batching method to update descriptors. On the other hand, the total latency to forward a packet increases along with the batch size. To achieve a reasonable throughput–latency balance, we adopt eight as the smallest batch size that can achieve the 40 Gbps wire-rate by 128-byte packets.

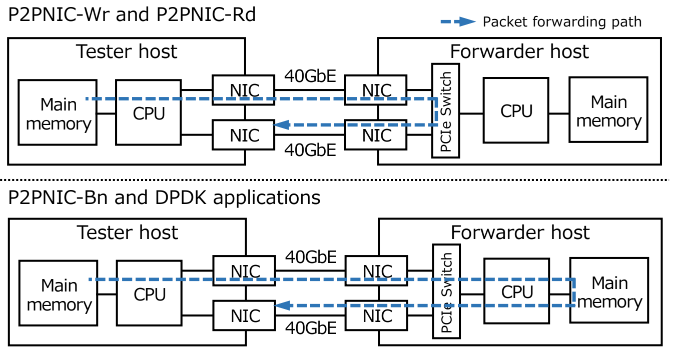

Fig. 4. Measurement setup for P2PNIC-Wr, P2PNIC-Rd, P2PNIC-Bn, and DPDK applications.

V. Evaluation

We evaluate the implementations of P2PNIC by measuring its performance and comparing it with the DPDK applications as examples of the CPU-driven packet forwarding architecture. The DPDK applications we used for the evaluation are TestPMD and L3FWD. TestPMD is for the performance test of drivers in DPDK, and it forwards packets without any processing on the packet data. On the other hand, L3FWD forwards packets based on IP routing. As the key metrics for the packet forwarding mechanism, we measured throughput (§ V-A) and latency (§ V-B). In addition, we confirmed that P2PNIC does not require any CPU processing and PCIe communication between CPU and NIC during packet forwarding by measuring their usage (§ V-C).

For the measurement setup, we used two hosts connected with two 40 GbE links with direct attach cables, as shown in Figure 4. We refer to a host to generate traffic as a tester host and another host to forward packets by the P2PNIC implementations or the DPDK applications as a forwarder host. For P2PNIC-Wr and P2PNIC-Rd, the CPU and main memory are not involved in the packet forwarding flow. The NIC for receiving packets transfers packets directly to the NIC for sending packets in the forwarder host by P2P DMA through the PCIe switch. In contrast, for P2PNIC-Bn and the DPDK applications, the CPU and main memory are involved in every packet forwarding.

The forwarder host has an Intel Core i9-9820X 10 core CPU and 32 GB DDR4 memory with an ASUS WS X299 SAGE motherboard. This motherboard has PLX8747 PCIe switches, and two Netronome Agilio LX 2x40GbE NICs were installed on PCIe slots under the same PCIe switch. The tester host has an Intel Core i7-9700K 8 core CPU, 32 GB DDR4 memory, with an ASRock Z390 motherboard. For NICs of the tester host, we used both Intel X710 and Mellanox ConnectX-4 NICs, depending on the measurement requirements.

For the configuration basis in this evaluation, all methods were configured to maximize their throughput. The parameters set in this basis are the batch size of NIC's descriptor updating (8) in P2PNIC-Wr, P2PNIC-Rd, and P2PNIC-Bn, and the number of queues (4), the corresponding CPU cores (4),

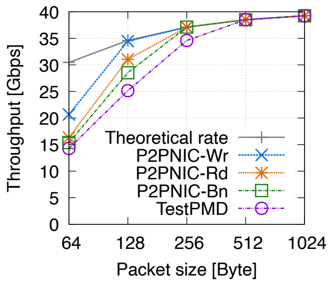

Fig. 5. Throughput without IP routing. All methods achieve the wire-rate above 1024-byte packet sizes.

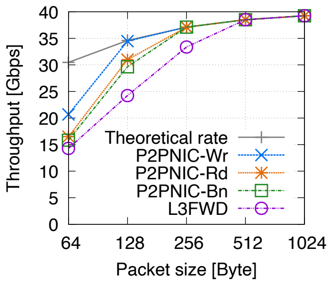

Fig. 6. Throughput with IP routing. All methods achieve the wire-rate above 1024-byte packet sizes.

and the batch size of CPU's processing (8) in P2PNIC-Bn and DPDK applications. To observe the bare impact of going through memory by comparing P2PNIC-Bn with other P2PNIC methods, we made their implementation and parameters (e.g., the batch size) as common as possible.

A. Throughput

Because throughput is one of the key metrics for the packet processing mechanism, we evaluated the throughput of the P2PNIC implementations and the DPDK applications with and without IP routing. For the packet generation, we used MoonGen [19], a DPDK-based software packet generator, and a Mellanox ConnectX-4 NIC to send packets because of its achievable rate, while we used an Intel X710 NIC to receive packets. We observed the throughput for packets of 64, 128, 256, 512, 1024, 1280, and 1518 bytes according to the hardware counter of the NIC for an average of 10 seconds, excluding the 10 seconds before and after.

According to the results for the measurements shown in Figure 5 and Figure 6, the throughput of P2PNIC-Wr and P2PNIC-Rd exceeded TestPMD and P2PNIC-Bn. When the packet size was 64 bytes, and IP routing was disabled, the P2PNIC-Wr achieved 40.37 Mpps, which is 1.45 times higher than that of TestPMD. For packet sizes above 512 bytes, all methods achieved the theoretical wire-rate, and this trend did not change with IP routing. The reason for the higher throughput of P2PNIC methods is that P2PNIC requires either DMA Read or Write to forward a packet, while the applications of the CPU-driven architecture require both. In this case, the P2PNIC-Wr's once issuing DMA Write showed higher throughput than the TestPMD's twice issuing of DMA Read and Write. Whether DMA write or read is faster may be different on other NIC because its performance depends on the DMA implementation in NIC. In addition, although the throughput gap between P2PNIC-Rd and P2PNIC-Bn is slight, it indicates that transferring packets through main memory has performance overhead compared with P2P DMA. Moreover, the gap would increase when the throughput exceed the bandwidth of the main memory.

B. Latency

To show that P2PNIC also has the advantage in latency over the CPU-driven architecture, we evaluated unidirectional latency for the P2PNIC implementations and the DPDK applications. We used Intel X710 NICs for both sending and receiving packets and MoonGen to measure the latency with the hardware timestamp feature. To observe the characteristics of latency with various conditions, we changed the traffic pattern for the packet size and traffic volume. However, because the XL710 NIC does not support the timestamp mechanism with high traffic volume, only 1 Kpps traffic was counted by the timestamp mechanism, and other packets were forwarded but not counted. We observed the latency in the aggregation of 30 seconds; thus, around 30,000 samples were counted for each measurement.

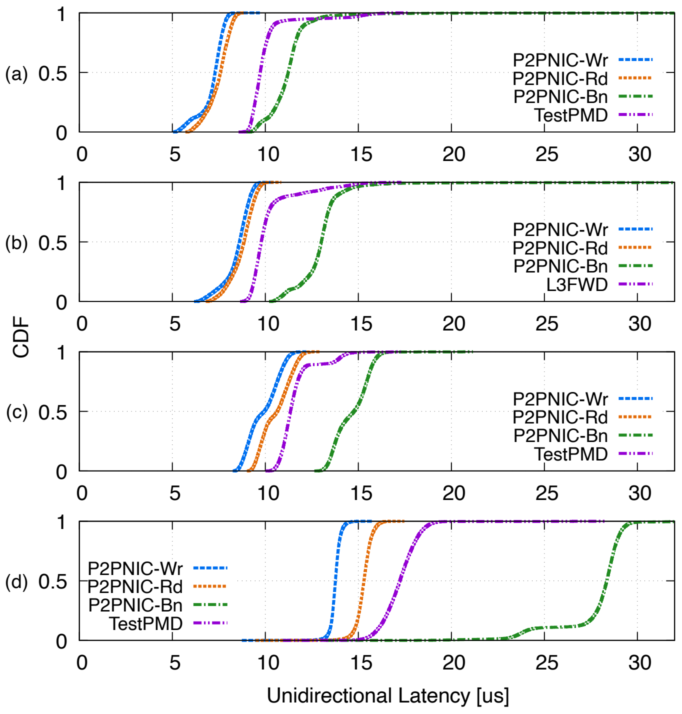

Fig. 7. CDF of the unidirectional latency in following conditions: (a) 64 bytes packets with 1 Gbps background traffic without IP routing, (b) 64 bytes packets with 1 Gbps background traffic with IP routing, (c) 1518 bytes packets with 1 Gbps background traffic without IP routing, and (d) 1518 bytes packets with 40 Gbps background traffic without IP routing.

In overall measurements, P2PNIC-Wr and P2PNIC-Rd had lower latency than the methods of CPU-driven architecture, including the DPDK applications. For the latency for 64-byte packets without IP routing shown in part (a) of Figure 7, P2PNIC-Wr and P2PNIC-Rd had 36% and 32% lower latency than that of TestPMD, respectively.

We see the three major factors for the increase of latency through the measurements: (1) processing of the protocol stack (i.e., IP routing), (2) packet size, and (3) traffic load. Part (b) of Figure 7 shows the latency for 64-byte packets when IP routing is enabled. There is an increase in latency of 18%, 16%, and 15% compared with the result without IP routing shown in part (a) of Figure 7 at the 90th percentile for P2PNIC-Wr, P2PNIC-Rd, and P2PNIC-Bn, respectively. For the DPDK applications, although a simple comparison between TestPMD and L3FWD

TABLE I
RESOURCE USAGE OF EACH METHOD
FOR FORWARDING 1518-BYTE PACKETS IN 40 GBPS.

	CPU usage	PCIe throughput
P2PNIC-Wr	0% per CPU core	0 MB/s
P2PNIC-Rd	0% per CPU core	0 MB/s
P2PNIC-Bn	100% per CPU core	9,785 MB/s
TestPMD	100% per CPU core	9,885 MB/s

is not accurate because their implementations totally differ, the results of L3FWD showed a 12% increase in latency over TestPMD at the 90th percentile.

In addition, the latency increases with the packet size for all methods. Part (c) of Figure 7 shows the latency for forwarding 1518-byte packets with 1 Gbps background traffic, and we can observe the latency increase with the packet size by comparing it with part (a) of Figure 7: the latency increases were 41%, 42%, 30%, and 27% at the 90th percentile for P2PNIC-Wr, P2PNIC-Rd, P2PNIC-Bn, and TestPMD, respectively.

Moreover, the latency also increases with background traffic. By comparing (d), which shows the latency for 1518-byte packets with 40 Gbps background traffic, with (c) of Figure 7, the increase rate of each method were 27%, 35%, 84%, and 38% at the 90th percentile for P2PNIC-Wr, P2PNIC-Rd, P2PNIC-Bn, and TestPMD, respectively. The higher increase rate of P2PNIC-Bn is likely because of its larger batch size than TestPMD for updating the descriptor rings. A larger batch size increases latency because of waiting for packets up to the size. P2PNIC-Wr and P2PNIC-Rd also used the same batch size, but they process batched packets in parallel by the NPU while P2PNIC-Bn processes them sequentially by the CPU. Thus, P2PNIC-Wr and P2PNIC-Rd are more tolerant to background traffic from the viewpoint of latency.

C. CPU usage and PCIe throughput

P2PNIC achieves fast packet forwarding on a general-purpose server without any processing on CPU for packet forwarding. We confirm the characteristic by measuring the actual usage of CPU and PCIe throughput in 40 Gbps traffic load by 1518-byte packets. The PCIe throughput means the amount of data that passed through the CPU for the data transfer between main memory and NICs. Therefore, the value would be near zero for P2PNIC-Wr and P2PNIC-Rd because their communication is completed under the PCIe switch.

From the result shown in Table I, we confirmed that P2PNIC-Wr and P2PNIC-Rd do not require any processing on the CPU for packet forwarding. Besides, because the communication of these methods is completed under the PCIe switch, the PCIe throughput did not increase during the measurement. On the other hand, because P2PNIC-Bn and TestPMD adopt busy waiting to process the packet received from the NIC, both methods consume all the capacity of allocated CPUs. In addition, we observe reasonable PCIe throughput for P2PNIC-Bn and TestPMD, which is nearly equal to 80 Gbps, that is, the sum of receiving and sending at 40 Gbps.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a novel packet forwarding architecture called P2PNIC, in which NICs drive packet forwarding, to overcome the potential limits of the CPU-driven architecture. We have implemented P2PNIC, and its evaluation shows that P2PNIC-Wr, which is one of the variants of P2PNIC, achieves 1.45 times higher throughput and 36% lower latency compared with TestPMD, a reference application of DPDK. P2PNIC architecture contributes to the enhancement of software middleboxes, and it could be an effective technology to accelerate network virtualization and cloud services. For future work, we aim to improve the scalability of P2PNIC, which would include the throughput improvement by multiple NICs and corresponding evaluation.

REFERENCES

[1] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, 2015.

[2] Y. Ohara, Y. Yamagishi, S. Sakai, A. D. Banik, and S. Miyakawa, "Revealing the Necessary Conditions to Achieve 80Gbps High-Speed PC Router," in *Proc. Asia. Int. Eng. Conf. (AINTEC)*, 2015, pp. 25–31.

[3] Intel DPDK Validation team, "DPDK Intel NIC Performance Report Release 20.08," 2020. [Online]. Available: https://fast.dpdk.org/doc/perf/DPDK_20_08_Intel_NIC_performance_report.pdf

[4] T. Barbette, C. Soldani, and L. Mathy, "Fast userspace packet processing," in *Proc. ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, 2015, pp. 5–16.

[5] DPDK Project, "DPDK: Home," 2020. [Online]. Available: https://www.dpdk.org

[6] J. D. Brouer and T. Høiland-Jørgensen, "XDP: challenges and future work," in *Proc. Linux Plumbers Conf.*, 2018.

[7] Netronome, "Agilio LX SmartNICs," 2020. [Online]. Available: https://www.netronome.com/products/agilio-lx/

[8] J. Aweya, *Architectures With Bus-Based Switch Fabrics: Case Study—Cisco Catalyst 6000 Series Switches*. Wiley-IEEE Press, 2018.

[9] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "RouteBricks: exploiting parallelism to scale software routers," in *Proc. ACM Symp. Operating Systems Principles (SOSP)*, 2009, pp. 15–28.

[10] L. Rizzo, "Netmap: a novel framework for fast packet I/O," in *Proc. USENIX Annu. Tech. Conf. (ATC)*, 2012, pp. 101–112.

[11] J. L. Hennessy and D. A. Patterson, "A New Golden Age for Computer Architecture," Feb 2019. [Online]. Available: https://cacm.acm.org/magazines/2019/2/234352-a-new-golden-age-for-computer-architecture/fulltext

[12] J. Zhang, D. Donofrio, J. Shalf, M. T. Kandemir, and M. Jung, "Nvmmu: A non-volatile memory management unit for heterogeneous gpu-ssd architectures," in *Proc. Int. Conf. Parallel Architecture Compilation Tech. (PACT)*. IEEE, 2015, pp. 13–24.

[13] S. Bergman, T. Brokhman, T. Cohen, and M. Silberstein, "SPIN: Seamless operating system integration of peer-to-peer DMA between SSDs and GPUs," *ACM Trans. Comput. Syst.*, vol. 36, no. 2, pp. 1–26, 2019.

[14] Broadcom, "PEX88000 Series Managed PCI Express 4.0 Switches," 2020. [Online]. Available: https://docs.broadcom.com/doc/BC-0484EN

[15] AMD, "AMD EPYC 7002 Series Processors," 2020. [Online]. Available: https://www.amd.com/en/processors/epyc-7002-series

[16] S. Pirelli and G. Candea, "A Simpler and Faster NIC Driver Model for Network Functions," in *Proc. USENIX Symp. Oper. Syst. Des. Implement. (OSDI)*, 2020, pp. 225–241.

[17] Y. Li, D. Zhang, A. X. Liu, and J. Zheng, "GAMT: a fast and scalable IP lookup engine for GPU-based software routers," in *Proc. ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, 2013, pp. 1–12.

[18] Linux Kernel, "VFIO - Virtual Function I/O," 2020. [Online]. Available: https://www.kernel.org/doc/Documentation/vfio.txt

[19] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "Moongen: A scriptable high-speed packet generator," in *Proc. Internet Measurement Conf. (IMC)*, 2015, pp. 275–287.