IEICE TRANS. INF. & SYST., VOL.E102–D, NO.11 NOVEMBER 2019

PAPER

Fast Datapath Processing Based on Hop-by-Hop Packet Aggregation for Service Function Chaining

Yuki TAGUCHI[†*], **Ryota KAWASHIMA**[†a)], **Hiroki NAKAYAMA**[††b)], **Tsunemasa HAYASHI**[††c)], ***and*** **Hiroshi MATSUO**[†d)], ***Members***

SUMMARY Many studies have revealed that the performance of software-based Virtual Network Functions (VNFs) is insufficient for mission-critical networks. Scaling-out approaches, such as auto-scaling of VNFs, could handle a huge amount of traffic; however, the exponential traffic growth confronts us the limitations of both expandability of physical resources and complexity of their management. In this paper, we propose a fast datapath processing method called Packet Aggregation Flow (PA-Flow) that is based on hop-by-hop packet aggregation for more efficient Service Function Chaining (SFC). PA-Flow extends a notion of existing intra-node packet aggregation toward network-wide packet aggregation, and we introduce following three novel features. First, packet I/O overheads at intermediate network devices including NFV-nodes are mitigated by reduction of packet amount. Second, aggregated packets are further aggregated as going through the service chain in a hop-by-hop manner. Finally, next-hop aware packet aggregation is realized using OpenFlow-based flow tables. PA-Flow is designed to be available with various VNF forms (e.g. VM/container/baremetal-based) and virtual I/O technologies (e.g. vhost-user/SR-IOV), and its implementation does not bring noticeable delay for aggregation. We conducted two evaluations: (i) a baseline evaluation for understanding fundamental performance characteristics of PA-Flow (ii) a simulation-based SFC evaluation for proving PA-Flow's effect in a realistic environment. The results showed that throughput of short packet forwarding was improved by 4 times. Moreover, the total number of packets was reduced by 93% in a large-scale SFC.

key words: *Packet Aggregation, Network Functions Virtualization, Service Function Chaining, vhost-user, DPDK*

1. Introduction

Network Functions Virtualization (NFV) [1] has been introduced into commercial networks for flexible operation. In NFV, Service Function Chaining (SFC) [2] realizes a toy-blocking style composition of software-based Virtual Network Functions (VNFs). This is an essential feature for next-generation networks including 5G [3] to realize a highly-flexible network slicing feature, and the performance of SFCs is requested to be comparable to that of existing hardware-based networks.

However, many existing studies have revealed that the performance of current VNFs is insufficient for mission-critical networks because of various packet processing overheads [4]–[8]. Even though fast data-plane technologies, such as DPDK [9] are effective in NFV-nodes, per-CPU core performance is an unignorable performance factor [4] to achieve 10+ GbE throughput for single datapath.

Many approaches have been proposed to efficiently handle a huge amount of traffic in NFV-enabled networks. Scaling-out is the major one that increases the computational resources, and various techniques have been proposed for both data/control planes. For instance, cutting-edge data-plane frameworks adopt a Lock-Free Multi-Threaded (LFMT) [10] design to effectively parallelize datapath processing, and recent VNF management systems like E2 [11] support auto-scaling of VNF instances such that the number of them are adaptively adjusted depending on the system load. However, scaling-out approaches are not silver-bullet solutions for performance problems because there are scalability limitations of the physical resources themselves and management complexity of them. To address these problems, yet another scaling-up approach is required.

In this paper, we propose a fast datapath processing method, Packet Aggregation Flow (PA-Flow), based on hop-by-hop packet aggregation to enhance the performance of SFCs. PA-Flow is a network-wide packet aggregation mechanism specialized to modern NFV-node environment accelerated by DPDK, and has following three novel features, unlike existing aggregation-based methods. First, packet I/O overheads of intermediate network devices including NFV-nodes are mitigated by reduction of the number of short packets. Second, aggregated packets can be further aggregated as going through the service chain in a hop-by-hop manner, which results in drastic performance improvement of upstream VNFs. Finally, nexthop-aware packet aggregation is realized using OpenFlow-like flow tables. This feature enables PA-Flow to be used in flexible multipath-oriented NFV environment. In addition, PA-Flow is designed to be available with various VNF forms (e.g. VM/container/baremetal-based) as well as virtual network I/O technologies (e.g. vhost-user/SR-IOV [12]), and its implementation does not bring noticeable delay for aggregation or even improves forwarding latency as shown in Sect. 6.

PA-Flow has been first proposed in our previous paper [13]. In the current paper, we extend implementation of PA-Flow and performance evaluation. For new implemen-

Manuscript received December 28, 2018.
Manuscript revised July 27, 2019.
Manuscript publicized August 22, 2019.

[†]The authors are with Nagoya Institute of Technology, Graduate School of Engineering, Nagoya-shi, 466–8555 Japan.
[††]The authors are with BOSCO Technologies Inc., Tokyo, 105–0003 Japan.
[*]Presently, with Line Corporation.
a) E-mail: kawa1983@ieee.org (Corresponding author)
b) E-mail: nakayama@bosco-tech.com
c) E-mail: hayashi@bosco-tech.com
d) E-mail: matsuo@nitech.ac.jp

DOI: 10.1587/transinf.2018EDP7444

tation, we append a hypervisor-based PA-Flow mechanism (PA-Flow-ethdev) to support baremetal-formed VNFs and SR-IOV. In addition, we conducted a more thorough evaluation of PA-Flow as follows:

- Network speed is upgraded to 40 GbE.
- Container/Baremetal VNFs are added.
- Non-CBR traffic patterns are also evaluated.
- Realistic SFC topologies are simulated based on the actual performance characteristics of PA-Flow.

The remainder of this paper is organized as follows. Related work is introduced in Sect. 2, and the effect of packet aggregation for SFC is discussed in Sect. 3. The design of PA-Flow is described in Sect. 4, and implementation details are provided in Sect. 5. The evaluation results are described in Sect. 6, and finally, the conclusion of this study and future work are given in Sect. 7.

2. Related Work

The problem of low performance NFV-nodes has been tackled by two types of approaches: (i) scaling-out that adaptively increases VNF instances and (ii) scaling-up that directly enhances datapath efficiency in NFV-nodes. A brief introduction of these methods is presented below.

2.1 Scaling-Out Approaches

Currently, major VNF applications adopt data-plane frameworks based on Lock-Free Multi-Threaded (LFMT) [10]. LFMT maximizes performance gain of multi-threading by minimizing synchronization overheads. Hence, the aggregate throughput (sum of each datapath's throughput) can be linearly raised by increasing the number of threads and corresponding I/O queues with the help of Receive Side Scaling (RSS) [14]. However, in terms of per-datapath performance, DPDK and similar technologies cannot achieve the wire-rate of 10+ GbE for consecutive short packets.

E2 [11] is a VNF management framework that adaptively deploys VNF instances based on the processing load. However, the load-based auto-scaling has pragmatic management problems. Auto-scaling requires both real-time load monitoring of each instance and on-demand path calculation/reflection mechanisms. However, the processing load of DPDK-powered VNFs cannot be measured by commonly used metrics, such as CPU/memory usage. In addition, increase in physical/virtual resources complicates their management and is restricted by physical capacity. Therefore, enhancement of *single* datapath performance in NFV-nodes is necessary to cope with exponentially growing-up traffic amount.

2.2 Scaling-Up Approaches

M. Bourguiba et al. [15] have revealed that packet copying at virtual network I/O is the performance bottleneck for VM-based VNFs. They proposed a packet aggregation method that reduces the amount of packets at the virtual network I/O, a datapath between the virtual NIC driver within the VM and the underlying hypervisor. However, processing costs of the underlying virtual switch and physical NIC are not mitigated, although improving the processing efficiency of these components is important for 10+ GbE networks. The work requires a queuing-before-aggregation-style implementation due to the non-burst-oriented packet processing framework in Xen/Linux, which results in obvious latency increases. Although the core idea of packet aggregation is similar to our proposal, our PA-Flow extends the datapath of aggregated packets, meaning that aggregated packets can be directly transmitted to the substrate network. We have carefully designed PA-Flow to realize this feature by introducing aggregation tables, identifiers, and the L2-in-L3 tunnel encapsulation technique as explained in Sect. 4. PA-Flow is dedicated to the burst-oriented framework based on DPDK and vhost-user, which brings negligible latency overhead at the aggregation process (see Sect. 5). Furthermore, our implementation of PA-Flow provides container-formed VNFs thanks to portability of the vhost-user protocol.

Zero-copy is a well-known technique to directly improve virtual network I/O performance. Various implementations of zero-copy have been proposed, such as memory page swapping [16] and packet pool sharing [17], [18]. Zero-copy actually improves VM-to-hypervisor data I/O, but its performance gain is small for short packets. We expect that zero-copy can be combined with our approach that increases average packet size, which maximizes the effect of zero-copy.

Cutting-edge network vendors have released SmartNICs [19], [20] having a programmable hardware chip. SmartNICs allow packet processing of virtual switches or VNFs to be offloaded, and they outperform fully-software-based processing. However, such a hardware offloading requires virtual switches or VNFs to be compatible with vendor-specific features, which worsens interoperability and maintainability of them. Besides, the offloading does not push up the throughput to the wire-rate speed of 40 GbE because of the limited computational power of the NICs.

T. Lan et al. [21] have proposed a SmartNIC-like FPGA platform for offloading stateless packet processing of VNFs. They demonstrated a high-performance virtual router using this platform. However, stateful processing required by firewalls cannot be offloaded due to the complicated multiplexing of VNFs. On the other hand, our approach does not depend on VNF processing types and physical devices, which consolidates its interoperability in practical network environment.

3. Problem Statement

3.1 Softwarization of Core Network

Softwarization of network functions is now a down-to-earth idea in large-scale backbone networks [22]–[24].

Generally, packets are prone to be gathered to the up-

stream of the network, and therefore, upstream VNFs have to handle a huge amount of short packets. According to several benchmark results [4], [6], per-datapath† performance of current NFV-nodes is far from the wire-rate of 10+ GbE. Hence, further improvement of per-datapath efficiency is necessary.

3.2 Practical Solutions

Scale-out approaches are effective to handle exponentially growing traffic amount, but continual enlargement of various resources is not viable. On the other hand, scale-up approaches directly enhance per-datapath performance by improving packet processing efficiency, and various types of techniques have been proposed including hardware-offloading and zero-copy. However, existing approaches have pragmatic issues in NFV environment as discussed in Sect. 2. We design our proposed PA-Flow as not only high-performance but also practical by satisfying the following conditions.

- The de-facto technologies including DPDK, vhost-user, and SR-IOV should be used with.
- Existing platforms (OS, hypervisor, virtual switch) should be supported.
- Stateless/Stateful VNFs should be supported.
- Various forms of VNFs (VM/Container/Baremetal) should be supported.
- A control plane should be provided using a standard communication protocol.
- Special hardware features should not be required.

3.3 Packet Aggregation

Packet aggregation is a promising approach to balance performance and pragmatism. First, packet aggregation can be provided as a software-implemented middle-box, and its functionality is transparent to VNFs and the underlying network if an appropriate protocol is used. Second, packet aggregation enlarges the average size of outgoing packets in nature, which mitigates the load of both NFV-nodes and network links. Finally, packet aggregation is an iterative process. That is, aggregated packets can be further aggregated as going through the network. In the following two sections, we explain the detailed architectural design and implementation of PA-Flow.

4. Proposed Method (PA-Flow)

In this section, we describe the design and policy of our method, PA-Flow, that is a yet another fast datapath processing mechanism based on hop-by-hop packet aggregation.

†A datapath is a set of functions (Rx, Process, and Tx) performed on a minimal set of CPU cores (typically, single core)

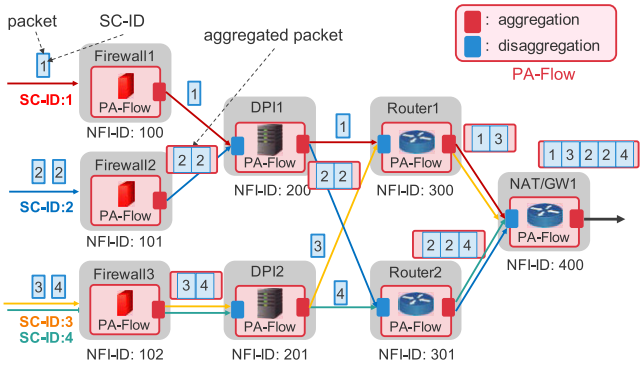


Fig. 1 An overview system of PA-Flow-enabled service chains

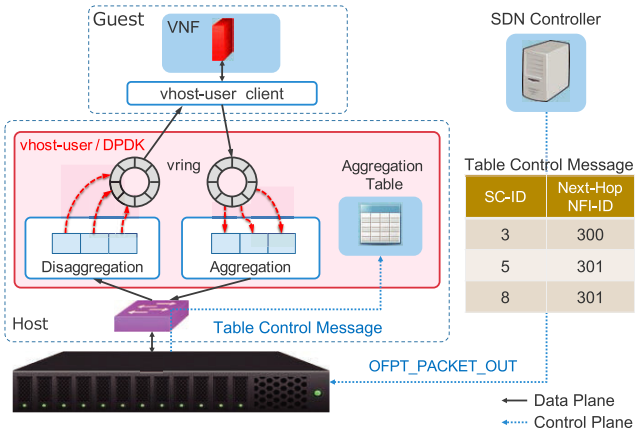


Fig. 2 An overview architecture of PA-Flow-enabled NFV-node

4.1 Architectural Overview

Before describing PA-Flow internals, we show a full picture of *PA-Flow-enabled* NFV network in Fig. 1. PA-Flow is just a software module and installed in each NFV-node that hosts VNF applications. Large-scale NFV networks provide multiple Service Functions Chaining (SFC) paths of the same function set for load balancing. Client nodes will transmit packets to their destinations, and the packets contain an identification of SFCs (Service Chain ID, SC-ID). The edge VNFs (a head of an SFC) handle the packets, and then, the PA-Flow module aggregates the outgoing packets with taking account of the next-hop VNFs that can be identified by Network Function Instance ID (NFI-ID). In the NFV-node of the next-hop VNF, the received (aggregated) packets are disaggregated by the PA-Flow module before the VNF's processing. After the processing, the module aggregates the transmitting packets again. This PA-Flow usage and nature bring considerable performance enhancement without losing compatibility with the existing NFV framework.

Figure 2 shows the internal of an NFV-node with PA-Flow. The PA-Flow module consists of three parts, Aggregation, Disaggregation, and the Aggregation Table. Aggregation/Disaggregation are performed transparently to both the VNFs and the virtual switch. The table, referred to identify the next-hop VNF, is indirectly connected to an SDN

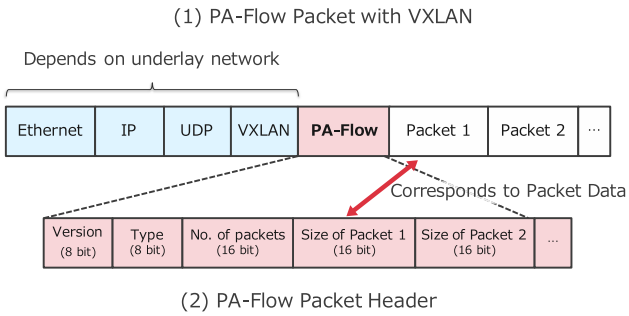


Fig. 3 A structure of an aggregated packet

controller using a common OpenFlow channel. The details of this controlling are explained in Sect. 5.3.1.

4.2 Remarkable PA-Flow Features

PA-Flow has three novel features derived from our design principles described in the previous section.

4.2.1 Network-Aware Aggregation

First, PA-Flow aggregates multiple packets into a single large packet that can be sent to the substrate network, unlike the similar method [15]. This feature enhances packet I/O performance of the hypervisor, and a dedicated encapsulation protocol is also designed to the aggregated packets can go through the network. If aggregated packets are directly transmitted to the network, not only physical network equipment but also a virtual switch, one of the major bottlenecks within the NFV-node, handles dramatically fewer amount of packets. As a result, this feature drastically improves the performance of NFV-nodes.

Figure 3 shows the format of our protocol. The PA-Flow header is a heart of packet aggregation/disaggregation, and indicates the number of packets (to be aggregated) and their locations within the aggregated packet using Type-Length-Value (TLV) format. The *version* field stores the protocol version, and the *type* field stores the type of embedded packets. The *number* field stores the number of embedded packets. Pairs of each embedding packet and its length follow the *number* field. The PA-Flow header is located on an existing protocol (e.g. UDP or VXLAN) that can specify the next protocol, which enables PA-Flow to co-work with various existing networks with little help of network administrators.

4.2.2 Hop-by-Hop Aggregation

Service Function Chaining is an essential building block of NFV, and the chains are extended by appending VNFs. In PA-Flow enabled SFCs, the aggregated packets can be further aggregated as going through the SFC, thanks to network-aware aggregation of PA-Flow. As a result, average packet sizes increases (= the number of packets decreases), and NFV-nodes on upper-stream of the SFC receive drastically fewer packets from the chain. For example, as shown in Fig. 1, the NFV-node for the uppermost VNF (NAT/GW) receives only two packets, while the NFV-nodes for edge firewall VNFs receive five packets in total.

4.2.3 Service Chain Oriented Aggregation

In real NFV environment, there are multiple next-hop VNF instances depending on load of NFV-nodes and network links. Hence, PA-Flow should aggregate packets depending on the next-hop VNF instances. We realized such an aggregation by using an OpenFlow-like flow table that containing *Key-Value* type entries. Service chain ID (SC-ID) that denotes a chain of involving VNF instances is used as a key in the table, and VLAN or VXLAN ID will be used in actual networks. Likewise, a value field contains a Network Function Instance ID (NFI-ID) to distinguish the next-hop. As in the case of Fig. 1, DPI1 and DPI2 have two next-hops, and therefore, the PA-Flow module of each node creates different addressed aggregated packets for Router1 and Router2 by consulting its table.

4.3 Concerns of Packet Drop

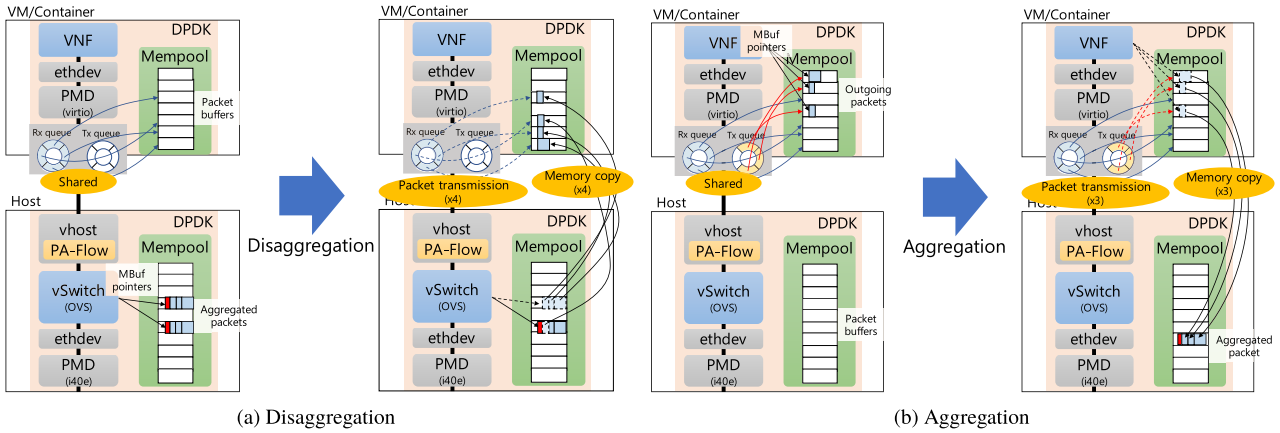
One of the major concerns of packet aggregation is the effect of packet losses. Once an aggregated packet is dropped, multiple packets will be re-transmitted from the source nodes due to the re-transmission oriented protocols (e.g. TCP, SCTP, and QUIC), and this results in deterioration of the communication quality. Certainly, the loss of aggregated packets is problematic; however, as shown in Sect. 6, PA-Flow decreases the packet loss probability because the number of packets on the network is drastically reduced (*burstiness* of the traffic is alleviated).

5. Implementation

We implemented the PA-Flow module as extensions of packet I/O APIs of DPDK ver. 17.11. The PA-Flow module has two implementations to fit in with various use cases.

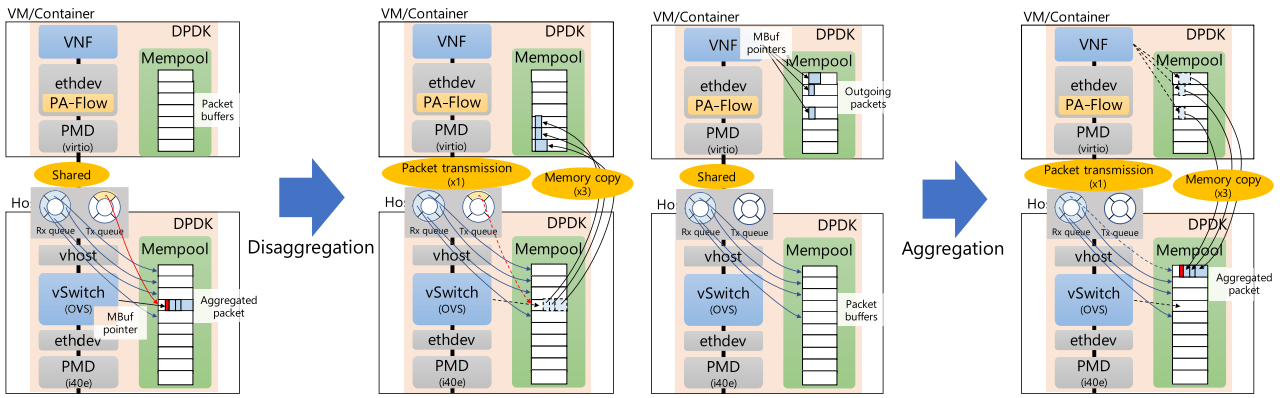
5.1 PA-Flow-Vhost

In this type, PA-Flow is deployed in the host, and works in the context of Host-to-VM/Container communications based on the vhost-user protocol. Specifically, the aggregation/disaggregation features are implemented in `rte_vhost_enqueue_burst/rte_vhost_dequeue_burst` APIs provided by a `librte_vhost` library of DPDK. Figure 4 shows implementations of the features for PA-Flow-vhost. A VM (or container) is running on the host, and there is a pair of shared Rx/Tx queues (vrings) between them. The queues themselves do not contain packet data but hold pointers to packet buffers provided by DPDK. Disaggregation is performed when a received (aggregated) packet is enqueued to the Rx queue in the context of the vhost-user protocol, and the PA-Flow module copies each embedded packet data within the aggregated packet to different packet



(a) Disaggregation (b) Aggregation

Fig. 4 Aggregation/Disaggregation implementations of PA-Flow-vhost



(a) Disaggregation (b) Aggregation

Fig. 5 Aggregation/Disaggregation implementations of PA-Flow-ethdev

buffers in the VM/Container by adjusting source memory addresses. Aggregation is also executed in the context of the vhost-user protocol. The PA-Flow module copies multiple packets that are located in different packet buffers of the VM/Container to a same packet buffer in the host as forming a PA-Flow's aggregated packet by adjusting destination memory addresses. The number of aggregating packets depends on the number of queue entries in the vring. That is, PA-Flow does NOT introduce any additional period for waiting packets, and this implementation prevents its latency from being prolonged. One of the major advantages of this implementation is that no-modification/configuration of VMs or containers is necessary, because PA-Flow's work is hidden by the vhost-user interface.

5.2 PA-Flow-Ethdev

The other implementation is PA-Flow-ethdev depicted in Fig. 5. The PA-Flow module resides in the *librte_ether* library[†] of the VM/Container. Contrary to PA-Flow-vhost, the host needs to be a vhost-user client that provides vrings to the server, meaning that the VM/Container can access packet buffers in the host via the queues. The purpose of the ethdev-type deployment is to reduce the number of packet transmissions (not memory copy) over the vhost-user protocol for better performance. For the disaggregation process, packets are transferred to the VM/Container based on the Tx queue, and the number of consumed queue entries equals to that of aggregated packets, which implies that aggregated packets are directly transferred on the vhost-user protocol. The PA-Flow module internally copies multiple embedded packets to different packet buffers as the disaggregation process, and therefore, the VNF handles the original packets without care of packet aggregation. The aggregation process is performed in the opposite direction to disaggregation, and the Rx queue is used to packet transmission to the host. The PA-Flow module copies multiple outgoing packets to a same packet buffer of the host by adjusting the destination memory addresses as long as they can be integrated as a single aggregated packet. Therein only a single Rx queue entry is consumed, meaning that the aggregated packet is sent from the VM/Container to the host in the context of the vhost-user protocol. One concern of PA-Flow-ethdev is that VNF managers must use PA-Flow-enabled DPDK for their VNF applications, while PA-Flow-vhost does not require such op-

[†]*librte_ether* is an abstraction layer for the underlying network devices, and this abstraction enables various VNF forms such as baremetal and VM/Container (w/wo SR-IOV).

erations to the managers.

5.3 Aggregation Details

5.3.1 Aggregation Table

PA-Flow looks up the aggregation table to identify the next-hop VNF instance when aggregating packets. The table has two columns, SC-ID and NFI-ID, and the entries are dynamically set by the SDN controller. Specifically, SC-ID is an OpenFlow-supported protocol header field, such as VLAN ID (VID) or VXLAN Network ID (VNI), but NFI-ID is a custom value and can be represented by another (seldom used) protocol header field. If VIDs are used as keys, the memory space of the table is 8 KB (2 bytes × 4096 entries), and this size is small enough to be stored in L1/L2 caches, which results in efficient search cost (discussed in Sect. 6.3). Matching mechanism of the aggregation table is depicted in Fig. 6. In this example, the table is referred for each packet stored in the *vring*, and VID is used as a key. Then, two aggregated packets are created based on the destinations (NFI-IDs).

Note that the flow entries can be dynamically installed (or modified/removed) using an OpenFlow channel. The PA-Flow module can interpret OpenFlow's *OFPT_FLOW_MOD* messages that contain entry information, but the module does not manage the channel. Instead, an OpenFlow-enabled virtual switch like Open vSwitch manages the channel, and it redirects the messages that are originally encapsulated in *OFPT_PACKET_OUT* to the module[†]. Such a mechanism requires additional C-Plane messages, but which does not affect D-Plane performance.

5.3.2 Limitations

Increasing the number of embedded packets in an aggregated packet brings out effectiveness of PA-Flow; however, our implementation requires a limitation of aggregated packet size due to Maximum Transmission Unit (MTU) of the network (Generally, 1500 bytes in Ethernet). Therefore, PA-Flow internally has a maximum size parameter of aggregated packets, and must care about not only the next-hop NFI-ID, but also packet size at the aggregation process.

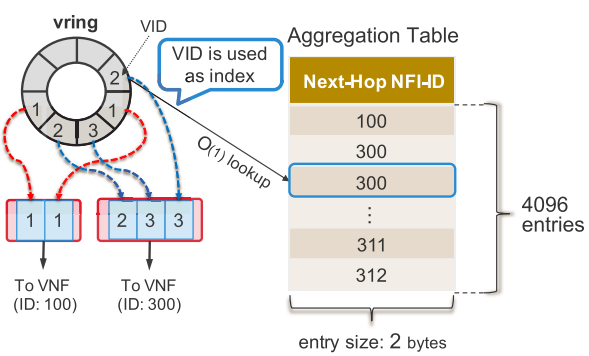


Fig. 6 Matching mechanism to get Next-Hop NF instance ID

[†]This technique is first presented in [25].

6. Evaluation

In this section, we describe the results of performance evaluation of PA-Flow, compared with that of common forms of VNFs atop the DPDK/vhost-user. We conducted two types of evaluations. The first one clarifies the fundamental performance characteristics of an NFV-node powered by PA-Flow. This evaluation consists of five types of experiments focusing on throughput, aggregation ratio, latency, traffic patterns, and packet drop respectively. Then, we simulated the effects of PA-Flow in large-scale service function chains using NS3 [26], according to the baseline evaluation results. Finally, we discuss how the aggregation table affects packet aggregation in terms of performance. The specifications of the physical servers are presented in Table 1.

6.1 Baseline Evaluation

The forwarding performance of a PA-Flow-enabled NFV-node was evaluated on a testbed that consists of two physical servers: *Tester* and *Device under Test (DuT)*. MoonGen [27], running on *Tester*, generated continuous fixed-length UDP packets with a specified transmission (Tx) rate for 30 seconds, and received the returning packets from DuT. The internals of the *Tester* node is depicted in Fig. 7 as Tester (A) and (B). *DuT* forwarded inbound packets to *Tester* via the forwarding-only VNF (Testpmd [28]). Open vSwitch powered by DPDK played a role of bridging between the VNF and the hypervisor, and PA-Flow aggregated outgoing packets or disaggregated received (aggregated) packets without introducing extra delay as explained in the previous section. The detailed compositions of *DuT* are shown in Fig. 7 as DuT (1) to (8). We further examined the performance of container-formed VNFs supported by Docker [29]. Docker-based containers can use the DPDK/vhost-user for their datapaths, and therefore, PA-Flow can be used like VM-based VNFs. Note that the lookup of PA-Flow's aggregation table was disabled for the evaluations. The lookup overhead is analyzed in Sect. 6.3.

Table 1 Machine specifications

	Physical Host	Virtual Guest
OS	CentOS 7.4 (3.10.0-693-x86_64)	
CPU	Intel Core i7-6900K (3.2 GHz, 8 cores, HT: off)	vCPU (4 cores)
Memory	64 GB	16 GB
NIC	Intel XL710 (40 GbE, 2 ports)	virtio-net (with vhost-user)
VMM	QEMU/KVM 2.10.1	-
Container	Docker-ce 17.12.1	-
vSwitch	Open vSwitch 2.9.0	
D-Plane	DPDK 17.11	

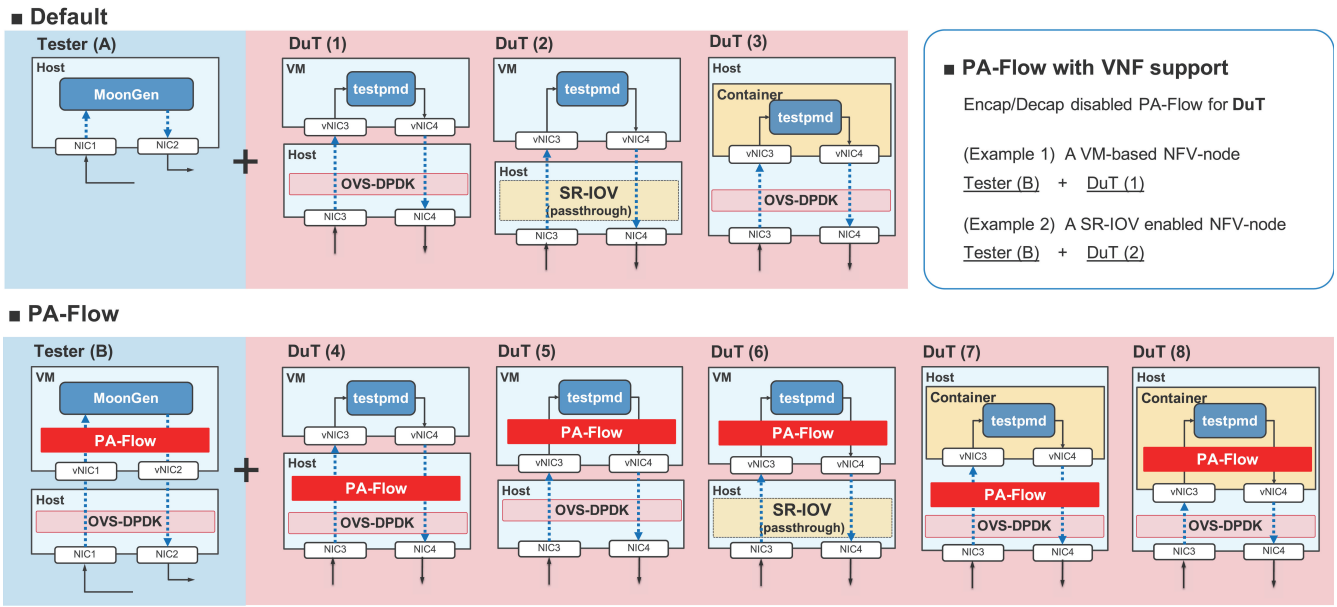


Fig. 7 Detailed configuration of Tester and DuT machines

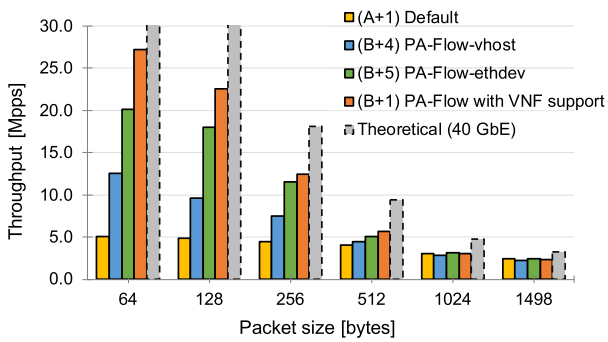


Fig. 8 Throughput of VM-based NFV-nodes

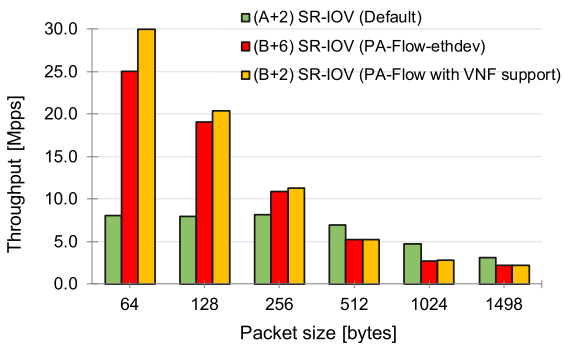


Fig. 9 Throughput of NFV-nodes with SR-IOV enabled

6.1.1 Throughput

Figure 8 shows the throughput of the NFV-node hosting a VM-formed VNF, with varying sizes of transmitting packets from MoonGen from 64 to 1498 bytes. We tested the following five patterns: ***(A+1) Default*** using DPDK/vhost-user for datapaths, ***(B+4) PA-Flow-vhost*** using vhost implementation of PA-Flow, ***(B+5) PA-Flow-ethdev*** using ethdev implementation of PA-Flow and ***(B+1) PA-Flow with VNF support*** using Encap/Decap disabled PA-Flow for DuT. As a result, when the packet size was 64 bytes, PA-Flow-vhost improved the throughput by 2.5 times, compared to Default. PA-Flow-ethdev improved the throughput by 4.0 times, and such a performance difference between two PA-Flow implementations were caused by efficiency of vring usage. In addition, in case VNFs can directly handle aggregated packets as they are (VNF support), maximum performance was seen for short packets, because overheads of packet disaggregations and re-aggregations on DuT were entirely skipped.

The throughput of NFV-nodes with SR-IOV is depicted in Fig. 9, and it shows PA-Flow improved the performance of SR-IOV by 3.1 times with regard to 64 bytes short packets. This implies packet aggregation is effective for hardware-oriented datapaths. On the other hand, PA-Flow shows lower throughput for larger packet sizes (512-). When the packet size exceeds 746 bytes, PA-Flow did not aggregate the packets due to the MTU constraint. Therefore, the lower values of 1024 and 1498 packet sizes can be explained as pure processing overhead of PA-Flow (e.g. calculating aggregated packet size). In case of 512-byte packets, only two packets can be aggregated and the overhead was still larger than the reduction of I/O overhead.

In the container evaluation, we compared Default and two implementations of PA-Flow module. The throughput of the container is shown in Fig. 10. The short packet forwarding of the container was improved by our method, and the maximum throughput was 20 Mpps. The result indicates that the bottleneck of the container-based VNF is the virtual switch running on the host, which is the same as the VM-based VNF. The PA-Flow's performance gain of docker-formed VNFs is same as VM-based ones.

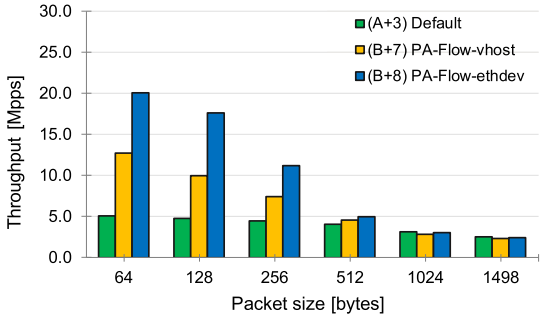


Fig. 10 Throughput of container-based NFV-nodes

Table 2 Relationship between Rx rate and packet Aggregation Ratio

Rx Rate [pps]	100 K	1 M	2M	4 M	6M	8 M
AR* (non-bursty) [%]	48.2	27.2	25.6	16.4	6.5	6.2
AR (bursty) [%]	12.4	12.0	10.5	9.6	8.3	6.2

*AR: Aggregation Ratio

6.1.2 Packet Amount in the Physical Network

We measured the relationship between the Rx rate of the NFV-node and the reduction of packet amount. We compared non-bursty traffic that has equal inter-packet gaps and bursty traffic that transmitted 32 packets consecutively (default burst value of DPDK). For clarity, "AR: Aggregation Ratio"[†] is defined as indicating the degree of reduction of packets before and after the aggregation. The actual measurement of aggregation ratio is shown in Table 2. In either traffic, as the Rx rate rose, the aggregation ratio also improved. Compared with non-bursty traffic, PA-Flow can aggregate more packets under bursty traffic. This was because PA-Flow-vhost aggregated multiple packets stored in *vring*, and more packets were stored at once under bursty traffic.

6.1.3 Latency

Next, we evaluate latency/jitter of PA-Flow[††]. Figure 11 shows the latency histogram with 99.9 percentile of ten million packets under 1 Mpps Rx rate of the DuT machine. From the result, our method does not deteriorate both latency and jitter. The reason is that reduction of the queuing delay was comparable or even larger to the processing overhead of PA-Flow, with considering the packets amount was reduced to about 27% in this experiment. Our method reduces end-to-end latency at high-rate communications because the aggregation rate increased as growing traffic amount. Besides, our implementation did not introduce additional waiting time as explained in the previous section.

[†]AR = (No. of aggregated packets) / (No. of packets before aggregated) * 100

[††]We used Intel X540 instead of Intel XL710 because both Default and PA-Flow showed unexplainable results on Intel XL710.

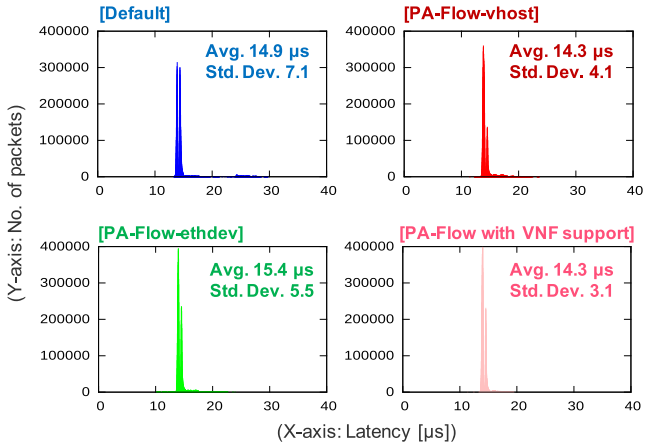


Fig. 11 Latency of NFV-nodes under 1 Mpps traffic

6.1.4 Traffic Patterns

As discussed in the above section, burstiness affects the aggregation ratio. In case of traffic with high burstiness and traffic with sufficiently high-rate, PA-Flow can aggregate the small packet into a larger packet. On the other hand, PA-Flow aggregates fewer packets under low-rate traffic, but in such a case, the performance of the NFV-node is sufficient to handle the traffic. Packet batching is now a crucial nature for high-performance software-based packet processing [9], [30], [31], and DPDK/vhost-user enable multiple packets to be handled in bursty manner throughout the Physical-Virtual-Physical (PVP) path. Therefore, traffic coming from such a node has bursty nature regardless of original traffic pattern. In terms of packet size, smaller packets (64-256 bytes) account for biggest percentage in real network environment (e.g. carrier networks) as growing the needs for video streaming and IoT communications. Therefore, PA-Flow can boost the performance of NFV-nodes in actual network environment regardless of traffic patterns.

6.1.5 Packet Drop

In this experiment, we evaluate fundamental characteristics of packet drops to understand how PA-Flow's packet aggregation affects the packet loss amount which is an important performance factor in TCP communications. We measured the total packet drop counts of PA-Flow comparing with the default method. The traffic was 64-byte UDP[†††] flows with varied bit rates controlled by MoonGen, from 1 Mpps to 5 Mpps. Table 3 shows the total number of dropped packets under various Rx rates of the DuT machine, and the result proves that PA-Flow can reduce the total amount of packet loss even through multiple packets are packed into a single large packet. In addition, we investigated the burstiness of packet loss (length of consecutive packet loss) at

[†††]We used UDP flows instead of TCP because of the following reasons; (i) TCP will avoid packet drops by reducing the bit rate (ii) MoonGen does not support a full-fledged TCP stack.

Table 3 The total number of dropped packets at certain bit rates

Rx Rate [pps]	1 M	2M	3 M	4M	5 M
# of drops (Default)	200K	184K	187K	1722K	3450K
# of drops (PA-Flow)	67K	130K	222K	313K	1654K

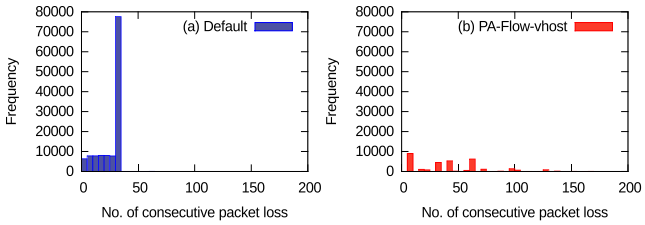


Fig. 12 Distribution of burstiness of drops under 5 Mpps Rx rate

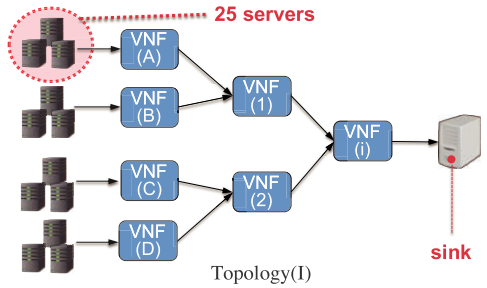


Fig. 13 Aggregated topology model for SFC evaluation

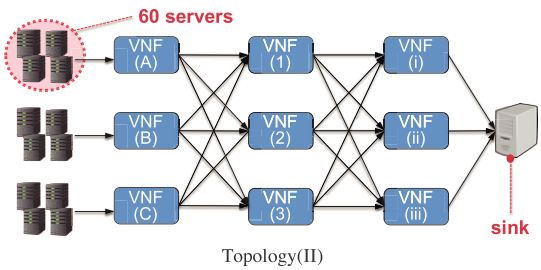


Fig. 14 Linear topology model for SFC evaluation

5 Mpps, and the result is shown in Fig. 12. From the figure, almost all the consecutive drop length was 32 (DPDK's maximum batch size) for the default method. On the other hand, the maximum burst length for PA-Flow was far larger, even though the total drop counts were lower. In summary, packet aggregation of PA-Flow did not increase the total packet drop amount, but in case aggregated packets consecutively dropped, the burstiness of (original) packet drops can be larger than that of the default method.

6.2 SFC Evaluation

We conducted a simulation-based evaluation to measure the performance impact of PA-Flow under large-scale SFCs, comparing two test network topologies as shown in Fig. 13 and Fig. 14. Topology(I) is a model where the number of VNF decreases towards the upstream of the network, which is favorable for packet aggregation. While Topology(II) is a model where the number of VNF does not decrease and there are some *branches*. Each end server generated fixed-length UDP packets in On-Off traffic that had 1 ms burst states and 0.1 - 1 ms random intervals.

We used NS3 version 3.26 for the simulation and created a model of NFV-node that can simulate the packet processing of PA-Flow. The model was designed so that the following features were equivalent to the real NFV-node with PA-Flow. First, Aggregation ratio was determined by the input rate. Second, the maximum output rate was calculated depending on whether the PA-Flow feature was enabled or not. Conversely, the following behavioral/characteristic changes were made to simplify the performance evaluation. First, next-hop VNFs were determined in round-robin fashion. Second, packet drops in NFV-nodes never occurred.

Table 4 Frequency distribution of packet size for Topology(I)

Default

Packet Size [bytes]	Link(A)-(1)	Link(1)-(i)	Link(i)-(sink)
64	1571032	3131644	4865252
65 - 1518	0	0	0
Avg. size [bytes]	64	64	64

PA-Flow

Packet Size [bytes]	Link(A)-(1)	Link(1)-(i)	Link(i)-(sink)
64 - 255	532496	116744	10240
256 - 511	0	699264	10848
512 - 767	0	0	21068
768 - 1023	0	0	395988
1024 - 1518	0	0	2452
Avg. size [bytes]	223.17	276.43	897.63

Table 5 Frequency distribution of packet size for Topology(II)

Default

Packet Size [bytes]	Link(A)-(1)	Link(1)-(i)	Link(i)-(sink)
64	1282443	1283147	3849441
65 - 1518	0	0	0
Avg. size [bytes]	64	64	64

PA-Flow

Packet Size [bytes]	Link(A)-(1)	Link(1)-(i)	Link(i)-(sink)
64 - 127	2761	3822	2366
128 - 255	426868	425992	2873
256 - 383	216	230	939935
384 - 511	0	0	503
512 - 639	0	0	31
Avg. size [bytes]	225.10	225.01	290.39

6.2.1 Performance Impact

The frequency distributions for each packet size on Link(A-1), Link(1-i), and Link(i-sink) in the Topology(I) and Topology(II) are shown in Table 4 and Table 5. In Topology(I), the average packet size was gradually increased as going through the service chain. The *sink* store received PA-Flow packets consisting of 15 short packets on average, and the

maximum length was 1426 bytes. In Topology(II), the average packet size was smaller than Topology(I) because packets were not gathered sufficiently due to the branch of the service chain, and the aggregation ratio was not improved. In terms of packet amount, the total number of packets at Link(i-sink) was reduced by 93% on Topology(I) and 75% on Topology(II). In real enterprise networks, usage of uplinks from edge devices (e.g. Top of Rack switches) will be high enough. Therefore, introducing the PA-Flow features into the bottleneck paths is a best way to handle a huge amount of traffic.

6.3 Discussion

In this part, we discuss the additional processing cost introduced by PA-Flow. We conducted two experiments to figure out aggregation/disaggregation and table lookup costs by comparing PA-Flow and the original implementations. First, we measured the CPU cycle consumption of two DPDK APIs, `rte_vhost_dequeue_burst` and `rte_vhost_enquque_burst`. The entire aggregation process is performed within the former API, and the disaggregation process is performed within the later one. Our results showed that the overhead for the former API was -29.0 cycles (performance was improved!), and the later one was +20.2 cycles on average. The main cause of the improvement is that the number of dynamic allocations/settings for rte_mbuf is reduced thanks to the packet aggregation.

Next, we investigated the table lookup overhead considering with the cache hit effect. From the result, average lookup cost was 25.4 cycles per packet when every packet had a same key (the cache is effective), while average lookup cost was 26.3 cycles in the worst case (each packet has a different key). These values are equivalent to about 8 ns overhead for each packet considering the clock rate (3.2 GHz), which means that the actual throughput (for 64-byte packets) can be reduced by 10% at maximum when the table lookup process is enabled.

7. Conclusion

Softwarizing various network functions and dynamically composing their chains make networks flexibly manageable. Scaling-out approaches, such as auto-scaling of VNFs, could handle a huge amount of traffic. However, scale-out will hit a plateau of physical resources without boosting up the fundamental datapath performance. Hardware-oriented approaches like SmartNICs can enhance packet processing efficiency, but they can pose flexibility and compatibility issues. In this paper, we proposed a fast datapath processing method, Packet Aggregation Flow (PA-Flow), that realizes fast Service Function Chains (SFCs). Thanks to the three NFV-dedicated features, throughput of SFCs is remarkably improved, and packet amount on the substrate network is drastically reduced as well. We have implemented the two types of PA-Flow modules, (*vhost* and *ethdev*), and this variation has PA-Flow used in myriad NFV environment. We evaluated performance characteristics of PA-Flow using 40 GbE environment (for baseline) and using NS3 simulator (for large-scale SFCs). From the results, PA-Flow achieved 20 Mpps throughput with 64-byte packets for a single datapath, and this is 4 times higher than that of common VNF environment using DPDK/vhost-user. Moreover, the results show that our approach gradually increases the average packet size (reduces the number of packets) as going through SFCs. Our evaluation results provide insight into effectiveness of PA-Flow in real network environment such that PA-Flow boosts up the performance limitation of DPDK/vhost-user as long as small packets (< 512 bytes) are predominant within the traffic. We are planning to extend the aggregation process of PA-Flow by handling multiple *vrings* together, which can improve aggregation ratio.

References

[1] "Network Functions Virtualisation [Online]," (2018, Aug. 10). Available: http://portal.etsi.org/nfv/nfv_white_paper.pdf
[2] "Service Function Chaining RFC 7665 [Online]," (2018, Aug. 10). Available: https://datatracker.ietf.org/doc/rfc7665/
[3] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From network sharing to multi-tenancy: The 5G network slice broker," IEEE Comm. Mag., vol.54, no.7, pp.32–39, July 2016.
[4] R. Kawashima, H. Nakayama, T. Hayashi, and H. Matsuo, "Evaluation of Forwarding Efficiency in NFV-nodes toward Predictable Service Chain Performance," IEEE Trans. on Network and Service Management, vol.14, no.4, pp.920–933, Dec. 2017.
[5] S. Lange, A. Nguyen-Ngoc, S. Gebert, T. Zinner, M. Jarschel, A. Köpsel, M. Sune, D. Raumer, S. Gallenmüller, G. Carle, and P. Tran-Gia, "Performance benchmarking of a software-based LTE SGW," Proc. 11th International Conference on Network and Service Management (CNSM 2015), pp.378–383, Nov. 2015.
[6] T. Barbette, C. Soldani, and L. Mathy, "Fast userspace packet processing," Proc. ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS 2015), CA, USA, pp.5–16, May 2015.
[7] R. Bonafiglia, I. Cerrato, F. Ciaccia, M. Nemirovsky, and F. Risso, "Assessing the performance of virtualization technologies for nfv: A preliminary benchmarking," Proc. 4th European Workshop on Software-Defined Networks (EWSDN 2015), Bilbao, Spain, pp.67–72, Sept. 2015.
[8] S. Gallenmüller, P. Emmerich, F. Wohlfart, D. Raumer, and G. Carle, "Comparison of frameworks for high-performance packet IO," Proc. ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS 2015), CA, USA, pp.29–38, May 2015.
[9] "DPDK," (2018, Aug. 10). Available: http://dpdk.org/
[10] "Vector Packet Processing (VPP) [Online]," (2018, Aug. 10). Available: https://fd.io/technology/
[11] S. Palkar, C. Lan, S. Han, K. Jang, A. Panda, S. Ratnasamy, L. Rizzo, and S. Shenker, "E2: a framework for NFV applications," Proc. 25th Symposium on Operating Systems Principles (SOSP 2015), New York, USA, pp.121–136, Dec. 2017.
[12] "SR-IOV [Online]," (2018, Aug. 10). Available: http://pcisig.com/specifications/iov/single_root
[13] Y. Taguchi, R. Kawashima, H. Nakayama, T. Hayashi, and H. Matsuo, "PA-Flow: Gradual Packet Aggregation at Virtual Network I/O for Efficient Service Chaining," Proc. 9th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2017), Hong Kong, pp.335–340, Dec. 2017.
[14] "Microsoft corp. receive side scaling [online]," (2018, Aug. 10). Available: http://msdn.microsoft.com/library/windows/hardware/

ff556942.aspx
[15] M. Bourguiba, K. Haddadou, I.E. Korbi, and G. Pujolle, "Improving Network I/O Virtualization for Cloud Computing," IEEE Trans. Parallel and Distributed Systems, vol.25, no.3, pp.673–681, 2014.
[16] D. Wang, B. Hua, L. Lu, H. Zhu, and C. Liang, "Zcopy-vhost: Eliminating Packet Copying in Virtual Network I/O," Porc. 42nd IEEE Conference on Local Computer Networks (LCN) 2017, Singapore, pp.632–639, Oct. 2017.
[17] "Soft patch panel - dpdk resources management framework [online]," (2018, Aug. 10). Available: http://git.dpdk.org/apps/spp/
[18] J. Hwang, K.K. Ramakrishnan, and T. Wood, "NetVM: High Performance and Flexible Networking Using Virtualization on Commodity Platforms," 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI'14), USA, pp.445–458, April 2014.
[19] "Open vSwitch Offload and Acceleration with Agilio®CX SmartNICs [Online]," (2018, Aug. 10). Available: https://www.netronome.com/media/documents/WP_OVS_Benchmarking.pdf
[20] "Mellanox BlueField SmartNIC [Online]," (2018, Aug. 10). Available: http://www.mellanox.com/related-docs/prod_adapter_cards/PB_BlueField_Smart_NIC.pdf
[21] T. Lan, Q. Han, H. Fan, and J. Lan, "Fpga-based packets processing acceleration platform for vnf," Proc. 8th IEEE International Conference on Software Engineering and Service Science (ICSESS) 2017, China, pp.314–317, Nov. 2014.
[22] "Lagopus Switch [Online]," (2018, Aug. 10). Available: http://lagopus.github.io/
[23] Y. Ohara and Y. Yamagishi, "Kamuee Zero: the Design and Implementation of Route Table for High-Performance Software Router [Online]," Internet Conference 2016, Tokyo, Japan, Oct. 2016.
[24] D.E. Eisenbud, C. Yi, C. Contavalli, C. Smith, R. Kononov, E. Mann-Hielscher, A. Cilingiroglu, B. Cheyney, W. Shang, and J.D. Hosein, "Maglev: a fast and reliable software network load balancer," Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation (NSDI'16), CA, USA, Oct. 2016.
[25] R. Kawashima and H. Matsuo, "Virtual NIC Offloading Approach for Improving Performance of Virtual Networks," IEICE Trans. on Communications, vol.J97-B, no.4, pp.639–647, 2014. (in Japanese).
[26] "ns-3 [Online]," (2018, Aug. 10). https://www.nsnam.org
[27] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "MoonGen: A Scriptable High-Speed Packet Generator," Proc. ACM Internet Measurement Conference (IMC 2015), pp.275–287, 2015.
[28] "Testpmd Application User Guide [Online]," (2018, Aug. 10). Available: https://doc.dpdk.org/guides/testpmd_app_ug/index.html
[29] "Docker [Online]," (2018, Aug. 10). Available: https://www.docker.com
[30] L. Rizzo, "netmap: A Novel Framework for Fast Packet I/O," Proc. 21st USENIX Security Symposium 2012, WA, USA, pp.101–112, Aug. 2012.
[31] M. Miao, W. Cheng, F. Ren, and J. Xie, "Smart Batching: a Load-sensitive Self-tuning Packet I/O Using Dynamic Batch Sizing," Proc. 18th IEEE International Conference on High Performance Computing and Communications (HPCC 2016), Sydney, Australia, pp.726–733, Dec. 2016.

Yuki Taguchi was born in 1993. He received B.E and M.E degree in Engineering from Nagoya Institute of Technology in 2017 and 2019 respectively. His research interest is Network Functions Virtualization. He is a member of IEICE.

Ryota Kawashima was born in 1983. He received the M.S. degree from Iwate Prefectural University in 2007 and the Ph.D. degree from the Graduate University for Advanced Studies in 2010. He was a Software Engineer with Access Company, Ltd. and Stratosphere, Inc. In 2013, he became an Assistant Professor with the Nagoya Institute of Technology. His research interest is high-performance software-based packet processing. He was a recipient of Best Paper Awards for 2016 IEICE Communications Society, and for IEEE NFV-SDN 2018. He is a member of IEICE, IEEE, and ACM.

Hiroki Nakayama received B.E. and M.E. degree in Information and Communication Engineering from Osaka City University, Osaka, Japan, in 2012 and 2014. He is currently a senior manager principal researcher of BOSCO Technologies Inc. His research work is in the area of network traffic modeling, network management, software defined networks, and performance modeling on communication networks.

Tsunemasa Hayashi was born in 1968. He received his M.E. degree from Tokyo Institute of Technologies in 1994, and completed the Slone program of Executive-Management of Technologies at Massachusetts Institute of Technologies in 2006. His technology interest areas are network operation / management, network virtualization and high-speed parallel operation. He worked for NTT Laboratories from 1994 to 2006 and was awarded the best paper from APDAC'97. He is CEO / president of BOSCO Technologies Inc. from 2012, and a member of IEICE.

Hiroshi Matsuo was born in 1960. He received his M.S. in 1985 and also received Ph.D. degree in 1989 from Nagoya Institute of Technology. He became an assistant professor in 1989, lecturer in 1993, associate professor in 1995, and professor in 2003 at Nagoya Institute of Technology. His research interest is distributed cooperative system. He is a member of IEICE, IPSJ, and IEEE.