# Cutting Packet Fat in Shallow VNF Chain Processing

Swati Goswami[student], Nodir Kodirov[student], Ivan Beschastnikh
{sggoswam, knodir, bestchai}@cs.ubc.ca
University of British Columbia

## 1 Motivation

Virtual Network Functions (VNF) have stringent latency requirements. Datacenter (DC) operators aim to provide performance guarantees while simultaneously optimizing resource utilization. To this end, we propose SMP (Split Merge Payload), a header-payload decoupling mechanism. SMP cuts packet processing overhead by eliminating unnecessary payload transmission.

SMP is motivated by the two-fold observation that (a) a wide range of VNFs process only the packet header, and (b) by avoiding payload transmission to such VNFs, we can improve end-to-end packet processing latency and throughput. VNFs that process only the header are called *shallow processing VNFs*. Network Address Translator (NAT), firewall (FW), and load balancers (LB) are common shallow VNFs. The chain, `Router`→`NAT`→`LB`, in Metron [1] is an example of a shallow VNF chain.

Fig. 1 shows SMP's packet flow using a generalized shallow VNF chain: $F_1, F_2 ... F_n$. Since, this chain does not transform packet payload, we split the incoming packets and forward only the headers (H) to the VNF chain. After the chain has processed the header, we merge the output header (H') with the payload (P) of the original packet, before forwarding to the destination. Thus, we maintain compatibility between SMP and non-SMP chain processing for network nodes above SMP.

## 2 SMP Semantics and Design Options

**Semantics.** *Functional* and *operational* semantics of SMP are equivalent to packet processing semantics in non-SMP VNF chains. SMP does not change behaviour of individual VNFs and their relative ordering within a chain, thus ensuring functional equivalence. For example, VNF policy-driven packet drops are same in SMP and non-SMP settings. Operational equivalence ensures that packets dropped in non-SMP, due to failure scenarios, such as link failure, are dropped in SMP as well.

**Design Options.** In the SMP approach, there is a trade-off between the time saved by avoiding payload transmission ($T_{tx/rx}$) and the time required to perform split/merge operations ($T_{s/m}$). The user perceived latency reduction with SMP is $T_{saved} = T_{tx/rx} - T_{s/m}$. The significance of $T_{tx/rx}$ depends on SMP deployment specifics. As an example, consider the modern DC leaf-spine topology with three layers: ToR, spine, and core switches. We can also implement split/merge operations (see Fig. 1) at the server NIC. However, VNF frameworks already minimize the data copy between NIC and CPU with zero-copy techniques [2, 3, 1], and $T_{saved}$ of SMP deployed on the NIC will be marginal. Hence, we discount SMP deployment on the NIC. However, SMP deployed on the core switch maximizes $T_{saved}$ due to $T_{tx/rx}$ savings across multiple network hops. It also increases buffer memory requirements because core switches process significantly more packets than spine and ToR switch.

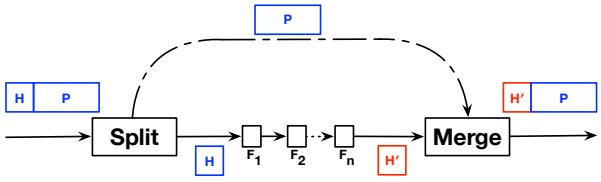


Figure 1: Packet processing in SMP. Header and payload are split before processing shallow VNF chain. Output header is merged with payload post-processing.

ToR switch nicely balances $T_{saved}$ and memory pressure. We plan to implement split/merge operations of Fig. 1 on the *same* ToR switch. When packets are received at the ToR, SMP executes the following steps:

- `Rx`: receive packets at the Rx port and tag the packet header to assist in merging packets.
- `Split`: split the packet payload and forward the tagged header to the server where the shallow VNF chain is allocated. Keep `(tag, payload)` tuple in the switch buffer, e.g., in a key-value store where tag is the *key* and payload is the *value*.
- `Merge`: receive the transformed header from VNF chain, and merge it back with the buffered payload.
- `Tx`: Trim the tag and send back the packet.

`Rx` and `Tx` steps are already performed by the ToR switches in modern DCs. The additional tagging/untagging operations can be performed at a line-rate similar to an in-dataplane VXLAN encap-

This NSDI'19 poster is supported in part by the Institute for Computing, Information and Cognitive Systems (ICICS) at UBC.

sulation/decapsulation done by VXLAN-aware ToR switches. The only overhead we need to address in our SMP prototype are the `Split` and `Merge` steps.

**Analysis**. To estimate $T_{saved}$, we evaluated $T_{tx/rx}$ on our commodity ToR switch (Arista 7050T-36 with 36 10GbE ports). We generated minimum (64 bytes) and maximum sized packets (1450 bytes[1]) with MoonGen [4] (pushing 5Gbps of throughput on 10Gbps NIC over 20s). The delta time ($T_{tx/rx}$) between minimum (16.02$\mu$s) and maximum sized packets (19.27$\mu$s) is $T_{tx/rx} = 3.25\mu$s on average[2]. This indicates that to reduce the packet processing latency with SMP ($T_{saved}$), split/merge operations ($T_{s/m}$) should complete in under 3.25$\mu$s. This is over three orders of magnitude lower than the 8ms control plane latency in literature [5]. Thus, split/merge operations should be done in the data plane.

Programmable switches have the ability and capacity to implement split/merge operations. As an example, Arista 7170-32C ToR switches with Barefoot Tofino chips [6] already allow the control plane to apply packet parsing, reassembly, and other non-forwarding functions to every packet at near line-rate speed [7]. Tofino chips also have the capacity to provide a key-value store interface. When SMP is deployed at rack scale, SMP has to buffer payload in a similar key-value store. At full saturation, our Arista 7050T-36 processes 360Gbps. With a round trip latency of 16.02$\mu$s, and 73$\mu$s packet processing latency of Snort (a heavy VNF) [8], we have to keep the payload in the buffer for 89.02$\mu$s (the expected processing time in a shallow VNF is likely to be less than Snort's latency, further relaxing memory pressure). With 1450 byte payload[3] and a 64 bytes header, we will require a 4.18 MB buffer (31Mpps for 360Gbps). This is almost 2$\times$ smaller than the buffer space used by NetCache [9] and is within the available buffer size of the Arista 7170-32C switch.

In summary, given different design options, SMP deployed on a ToR switch uniquely trade-offs the latency and memory overhead of split/merge operations. In addition to these design options, we need to address several other challenges described in the following section.

## 3 Discussion

Ensuring consistent packet drop behaviour in SMP and non-SMP is part of functional and operational equivalence. There are two sources of packet drops: policy driven (due to VNFs such as FW) and failure induced (due to events such as link failure and congestion). Policy-driven packet drops are handled by short-circuiting the remainder of the chain and signalling the switch to drop the buffered payload. Failure induced packet drops are challenging because unsignalled packet drops will increasingly consume limited switch buffer memory. We address this memory overconsumption using a watchdog mechanism. The factors influencing the purge threshold include available buffer space, processing time of VNFs, workload characteristics, and SLAs.

Existing work addresses latency reduction with different techniques. For example, netmap [10] accelerates packet processing in end-host, PacketShader [11] exploits parallelism using GPUs, and lightweight-DPI [12] analyzes a subset of payload to improve processing latency. Such techniques are complementary to SMP and can further optimize performance. SMP is partly inspired by Cut Payload that *drops* packet payload to accelerate TCP congestion-detection [13]. SMP differs because it *buffers* the payload as opposed to dropping it completely. To the best of our knowledge, we are the first to leverage *header-payload decoupling* to reduce latency in VNF chains. Moreover, SMP can be extended to chains with payload processing VNF(s) by applying SMP to the shallow subset of the VNF chain.

In conclusion, we present SMP, an approach to improve shallow VNF chain performance. We will evaluate SMP under varying workloads and VNF chains of different length and depth. Our initial analysis indicates that SMP reduces tenant perceived latency when deployed in the cloud and private enterprise setting.

[1] We do not use MTU size (1500 bytes) to avoid fragmentation.

[2] Unless otherwise indicated, all numbers are averages of three runs.

[3] For example, NetCache suggests that multiple register arrays, packet mirroring, and multiple piping rounds can be used to accommodate MTU sized values in Tofino chip's key-value store.

## References


[1] G. P. Katsikas et al. Metron: NFV Service Chains at the True Speed of the Underlying Hardware. In *NSDI*, 2018.

[2] J. Hwang et al. Netvm: High performance and flexible networking using virtualization on commodity platforms. In *NSDI*, 2014.

[3] W. Zhang et al. Opennetvm: A platform for high performance network service chains. In *HotMIddlebox*, 2016.

[4] P. Emmerich et al. Moongen: A scriptable high-speed packet generator. In *IMC*, 2015.

[5] K. He et al. Measuring control plane latency in sdn-enabled switches. In *SOSR*, 2015.

[6] Nextplatform.com. Arista runs barefoot with tofino programmable switch chips, 2018. [Online at nextplatform.com/2018/06/12/arista-runs-barefoot-with-tofino-programmable-switch-chips; accessed 01-12-2019].

[7] C. Kim. Programming the network data plane, 2016. [Online at conferences.sigcomm.org/sigcomm/2016/files/program/netpl/netpl16-kim.pdf; accessed 01-12-2019].

[8] G. Liu et al. Design Challenges for High Performance, Scalable NFV Interconnects. In *KBNets*, 2017.

[9] X. Jin et al. Netcache: Balancing key-value stores with fast in-network caching. In *SOSP*. ACM, 2017.

[10] L. Rizzo. netmap: a novel framework for fast packet I/O. In *USENIX Annual Technical Conference*, 2012.

[11] S. Han et al. Packetshader: a gpu-accelerated software router. In *SIGCOMM*, 2010.

[12] S. Fernandes et al. Slimming down deep packet inspection systems. In *IEEE INFOCOM Workshops*, 2009.

[13] P. Cheng et al. Catch the Whole Lot in an Action: Rapid Precise Packet Loss Notification in Data Center. In *NSDI*, 2014.