SLA-NFV: an SLA-aware High Performance Framework for Network Function Virtualization

Chen Sun[1], Jun Bi[1], Zhilong Zheng[1], Hongxin Hu[2],
[1]Institute for Network Sciences and Cyberspace, Tsinghua University
[1]Department of Computer Science, Tsinghua University
[1]Tsinghua National Laboratory for Information Science and Technology (TNList)
[2]Clemson University *

ABSTRACT

We propose SLA-NFV, a Service Level Agreement (SLA) aware framework, for building high-performance NFV, focusing on fulfilling SLAs of service subscribers (or tenants). SLA-NFV leverages a hybrid infrastructure with both software and programmable hardware to enhance NFV's capability with respect to various SLAs. Evaluations show that a hybrid service chain could reduce latency by up to 60% compared with a pure software service chain.

CCS Concepts

•**Networks** → **Middle boxes / network appliances;** *Programmable networks;* Control path algorithms;

1. INTRODUCTION

Network Functions Virtualization (NFV) was recently introduced to address the limitations of dedicated middleboxes and offer the potential for both enhancing service delivery flexibility and reducing overall costs. Many efforts have been devoted to exploiting the flexibility of virtualization and providing more scalable services for NFV. For example, Gember et al. [1] proposed OpenNF to help NFV network operators monitor and manipulate network state. However, current research mainly focuses on enhancing and optimizing NFV from the perspectives of service providers and network operators.

In contrast, we provide a novel aspect to recognize and enhance NFV from the perspective of service subscribers (or tenants). What a tenant cares is the fulfillment of functionality requirements, as well as *performance* requirements, such as processing latency and throughput, which could be summarized as Service Level Agreements (SLAs). Regarding functionality, current software-based NFV could provide rich functionality and flexibility through its virtualization techniques. However, packet processing in software-based Virtualized Network Functions (VNFs) would introduce significant performance overhead (e.g., Ananta SMux at 100K pps can add from 200μsec to 1ms of forwarding latency), which could be unacceptable for applications, such as algorithmic stock trading and high performance distributed memory caches that demand ultra-low (a few μs) latency within the NFV network. Moreover, current service chaining mechanisms for NFV [2] simply chain required VNFs, without considering performance requirements from NFV tenants.

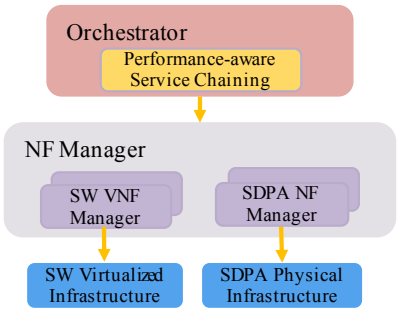

Figure 1: SLA-NFV network abstraction

To address above problems, in this poster, we propose SLA-NFV, a framework targeting on SLA fulfillment for NFV tenants. We integrate a stateful and NetFPGA accelerated data plane named SDPA [3] into traditional software NFV environment, leveraging both high performance of hardware and rich functionality and flexibility of software. Inside the NFV orchestrator, we design a performance-aware service chaining algorithm to fulfill both functionality and performance requirements with respect to SLAs. We implement an SLA-NFV prototype based on OpenStack and NetFPGA. Experimental results show that a hybrid service chain could reduce the forwarding latency by 60% compared with a pure soft-

*This research is supported by the National Natual Science Foundation of China (No.61472213 and No.6150-2267). Jun Bi is the corresponding author.

SIGCOMM'16, August 22–26, 2016, Florianopolis, Brazil.

ACM ISBN 978-1-4503-4193-6/16/08...
DOI: http://dx.doi.org/10.1145/2934872.2959058

ware service chain, and SLA-NFV could quickly build a service chain that fulfills SLAs in $1.0ms$.

2. DESIGN

As shown in Figure 1, we abstract the NFV network into three layers. The orchestrator performs service chaining and determines VNF deployment, destruction and migration. NF managers control NF life cycle. We combine programmable SDPA hardware with NFV software to achieve both high performance and flexibility.

2.1 Hybrid Infrastructure

The key challenge of integrating SDPA into NFV is to maintain NFV's flexibility of NF deployment, destruction and migration in SDPA. For NF deployment, SDPA NF manager could easily configure the infrastructure to support a specific network function. For NF destruction, traffic can be steered away from the instance and the manager can re-configure the old instance into a new one. However, previous work lacks a mechanism to support the migration between SDPA NF instances. Therefore, we extend SDPA infrastructure with state-relevant interfaces to migrate state of any **type** for any **flow**:

`readStateFromInstance(`**type**`, List<`**flow**`>)`
`writeStateToInstance (`**type**`, List<`**entry**`>)`

2.2 Performance-aware Service Chaining

As there exist several instances of the same NF in the network, the algorithm addresses the challenge of quickly choosing the best path to meet requirements of functionality, latency and throughput. The orchestrator should periodically query latest performance of VNFs through NF managers as input of the algorithm.

Step 1: Latency fulfillment. The latency mainly indicates the processing latency of NF instances. Provided that *Functions* $1, 2, 3$ are required. We assume *Function* 1 has an instance #1, *Function* 2 has instances #2, #4 and *Function* 3 has an instance #3. The possible paths of this requirement can be expressed as:

$Begin \rightarrow (\#1) \rightarrow (\#2, \#4) \rightarrow (\#3) \rightarrow End.$

We take the expression as three *stages* and use a *Greedy Policy* to *quickly* find the path with the lowest latency. For each instance $N_{x,i}$ in stage i, we pick the instance with the lowest latency and finally find the fastest path.

Step 2: Throughput fulfillment. The algorithm checks whether all instances of the fastest path can provide sufficient throughput. If so, the algorithm finishes. If some instances fail the check, the algorithm marks their latency as *infinite* and goes back to Step 1. The cycle continues until a qualified path is found or no path is picked, in which case new instances will be deployed.

3. IMPLEMENTATION

We have implemented hardware NFs based on SDPA architecture on NetFPGA, and software NFs on Dell PowerEdge R730 servers with 40 Intel Xeon CPUs (3.00 GHZ), 256G memory and 10G NICs under the management of OpenStack. We have implemented hardware and software versions of NAT, stateful firewall, elephant flow detection and a software DPI. In SDPA, NFs are abstracted into state machines and we can easily program an NF by assigning state transition policies and actions under different states (stateful firewall: <100 LoC). We issued 100 flow filtering rules to the stateful firewall and 100 regular expressions to DPI to match on packet payloads. We have also implemented the performance-aware service chaining algorithm in Floodlight (∼1.5K LoC).

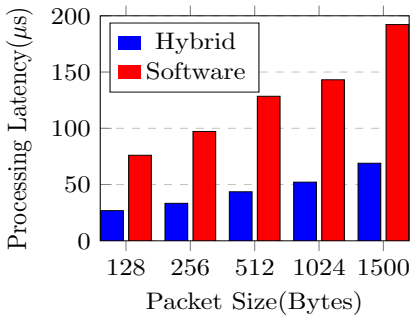

Figure 2: Service chain forwarding latency

We form two service chains. The first one contains 3 hardware instances and software DPI, while the second chain contains 4 software instances as introduced above. Figure 2 shows that forwarding latency of a hybrid service chain decreases by up to 60% compared with pure software implementation. We run the performance-aware service chaining algorithm to process 10,000 tenant requirements, each containing functionality, latency and throughput demands. Results demonstrate that the algorithm can form a qualified chain in an average of $1.0ms$, which could quickly adapt to SLA requirements.

4. CONCLUSION

In this poster, we have proposed SLA-NFV framework targeting on fulfilling tenants' SLAs by supporting NFs with both hardware and software infrastructures. We have implemented a performance-aware service chaining algorithm to fulfill SLAs.

5. REFERENCES

[1] Gember-Jacobson, A., Viswanathan, R., Prakash, C., Grandl, R., Khalid, J., Das, S., and Akella, A. Opennf: Enabling innovation in network function control. In *Proceedings of the 2014 ACM conference on SIGCOMM* (2014), ACM, pp. 163–174.

[2] Halpern, J., and Pignataro, C. Service function chaining (sfc) architecture. *draft-ietf-sfc-architecture-07 (in progress)* (2015).

[3] Zhu, S., Bi, J., Sun, C., Wu, C., and Hu, H. Sdpa: Enhancing stateful forwarding for software-defined networking. In *Proceedings of the 23rd IEEE International Conference on Network Protocols (ICNP 2015)*, pp. 323–333.