VNF Chain Abstraction for Cloud Service Providers

Nodir Kodirov[♣] Sam Bayless[♣] Fabian Ruffy[♣] Ivan Beschastnikh[♣] Holger H. Hoos[⋆♣] Alan J. Hu[♣]
[♣]University of British Columbia, Canada [⋆]Universiteit Leiden, The Netherlands
{knodir, sbayless, fruffy, bestchai}@cs.ubc.ca, hh@liacs.nl, ajh@cs.ubc.ca

Abstract

We propose VNF chain abstraction to decouple a tenant's view of the VNF chain from the cloud provider's implementation. We motivate the benefits of such an abstraction for the cloud provider as well as the tenants, and outline the challenges a cloud provider needs to address to make the chain abstraction practical. We describe the design requirements and report on our initial prototype.

1 INTRODUCTION

Network Functions (NF) are an important part of the enterprise networks. A large-scale survey by Sherry et al. [10] reveals that on a typical enterprise the number of NFs and traditional L2/L3 routers are comparable. The authors also propose to outsource the network processing of an enterprise as a Virtual Network Function (VNF) deployed in the cloud. Recent work extends this model with a *VNF chain* where network traffic can be processed by multiple VNFs [3].

However, VNF chain allocation and management at scale remains under-explored. In particular, only a few papers address the scaling of VNF chains in a *resource-agnostic way* [2, 6], but these do not apply to the cloud setting where tenant isolation and SLA guarantees are a primary concern. Consider a VNF chain with 8 Gbps bandwidth shown in Fig. 1(a). The actual bandwidth the cloud allocates is 16 Gbps: sum of ingress and egress traffic. How can this VNF chain be allocated on the resources in Fig. 1(c)? Answer: it cannot, because there is only 10 Gbps of servers-to-ToR uplink bandwidth available in Fig. 1(c).

We propose an *abstract-concrete decoupling*, or *abstraction* for short, of VNF chains to (1) free cloud service offerings from infrastructure limitations, and (2) enable high resource utilization to the cloud provider. Such decoupling splits VNF chains into *abstract* and *concrete* chains. An abstract chain is the one that a tenant requests to allocate. It describes NF elements, their topology, and SLA constraints. A cloud operator *realizes* the abstract chain using one or more *concrete chains*. Fig. 1(b) shows two concrete chains that correspond to the single abstract chain from Fig. 1(a).

Unlike cloud resources consumed by VMs today, where VM flavor cannot exceed compute and memory available on an individual server, an abstract chain (or an abstract VNF) is resource-agnostic, i.e., not restricted by an individual server's capacity. Since the abstract chain can be realized using multiple concrete chains, the cloud operator can multiplex the throughput of several concrete chains to match the abstract chain capacity, e.g., the aggregate bandwidth of the `concrete-chain-1` and `concrete-chain-2` are multiplexed to provide 16 Gbps bandwidth of the abstract chain in Fig. 1(a). Such multiplexing not only frees the cloud provider from bounding their services because of the infrastructure limitations, but multiplexing also simplifies the tenant experience. Tenants manage their VNF chain using the same API regardless of the resource footprint of the VNF chain. Moreover, chain abstraction allows for high data center (DC) utilization by enabling finer grained resource allocation. Cloud operators can achieve higher density packing of the concrete chains as resource requirements of the concrete chains make a subset of the abstract chains.

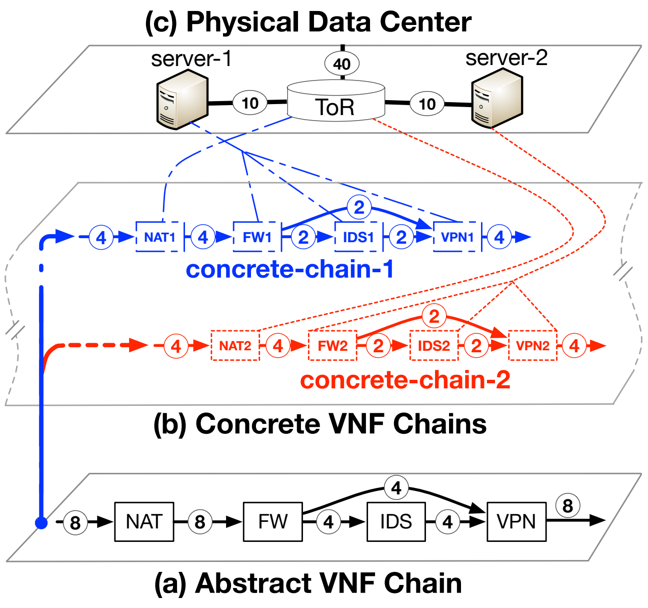

Figure 1: VNF chain decoupling: (a) an abstract chain the tenant requests to allocate, (b) the abstract chain is split into two concrete chains, and (c) cloud resources where concrete chain NFs get placed. Note that `NAT1` and `NAT2` are placed on `ToR`. The other NFs of the `concrete-chain-1` are placed on `server-1`, and NFs of the `concrete-chain-2` are placed on `server-2`.

ACM ISBN 978-1-4503-5902-3/18/07.
https://doi.org/10.1145/3230718.3232109

However, there are multiple challenges associated with making chain abstraction practical. These include (1) state synchronization between multiple concrete chains, (2) guaranteeing low-latency packet processing across concrete chains, (3) packet loss during normal operation and scaling up/down of an abstract chain, and (4) potential efficiency losses due to splitting a single abstract chain into multiple concrete chains. Next, we describe our approach to handle these challenges.

2 CHALLENGES

State synchronization between multiple concrete NFs introduces a trade-off between consistency and latency. As reported by earlier work, e.g., E2 [6], designs with full consistency are computationally expensive, operationally complex, and incur high latency [4, 8]. As we target cloud scale deployment with latency SLAs, the challenge is to *design a low-latency state synchronization mechanism without violating correct functionality across multiple concrete chains.*

Low-latency. In addition to latency inflation because of state synchronization, another latency contributing factor across multiple concrete chains is the network distance between physical servers where chains are deployed. For example, when all concrete chains of an abstract chain get deployed on the same server the packet processing latency across these concrete chains is going to be comparable and minimal. However, if one concrete chain gets deployed across two servers of the same rack, that concrete chain instance will incur extra rack-level round trip latency. This latency impacts a subset of the flows and must not violate the abstract chain SLA. As the distance is an artifact of the scheduler the challenge is to *design a distance-minimizing chain scheduler.*

Packet loss rate is specified by tenants as part of the SLA. The cloud provider has to ensure that the packet loss during normal operation and scale up/down operations do not violate the requested SLA. Scaling down the throughput of an (already deployed) abstract chain, in particular, is susceptible to packet loss, as the number of concrete chains has to be decreased in response to the decreasing load. If a concrete chain has a long-lived flow, deallocating that concrete chain introduces a packet loss. Thus, the challenge is to *design a chain scale up/down mechanism with minimal packet loss.*

Efficiency loss. Consolidating VNFs saves server CPU and RAM resources [9]. Proposed chain abstraction goes against such optimization as we not only prevent consolidation between different types of NFs, but also split a single abstract NF into multiple concrete NFs. Although we argue that VNF consolidation is not practical in the cloud setting due to the tenant-level isolation requirements, such as security and predictable performance expectations, the challenge is to *design abstract-concrete chain decoupling mechanism with minimum compute and memory overhead.*

3 PROTOTYPE AND FUTURE WORK

Our initial prototype confirms the feasibility of chain abstraction [5]. We used the Sonata framework [7] to build and deploy VNF chains. We used an Azure VM with 64 cores and 432 GB RAM to create 50 virtual hosts emulating a rack-scale deployment. The tenant requested allocation of the Fig. 1(a) abstract chain with the maximum bandwidth. We configured each concrete chain to support 10 Mbps bandwidth (due to VM resource limitations) and our algorithm allocated a total of 75 concrete chains, which is the maximum possible. We successfully passed 10 Mbps of traffic through each concrete chain with full performance isolation. We also performed large scale simulations on Facebook's recently published [1] DC topology. In a pod with 768 servers we achieved 100% server-locality (i.e., all NFs of the concrete chains are co-located) by allocating a total of 480 concrete chains, which is the maximum possible.

Although the preliminary results show that chain abstraction can indeed facilitate high DC utilization with a simple tenant-facing chain management API, we are yet to scale an abstract chain across multiple physical servers. We plan to overcome Sonata's single-server runtime limitation by building an E2-like system and by integrating a low-latency state synchronization mechanism, designing scale up/down mechanism with minimal packet loss, and implementing a lightweight abstract-concrete chain decoupling mechanism to thoroughly evaluate the benefits of our chain abstraction.

REFERENCES

[1] Alexey Andreyev. 2014. Introducing data center fabric, the next-generation Facebook data center network. (2014). [Online at code.facebook.com/posts/360346274145943; accessed 05-31-2018].

[2] Bilal Anwer, Theophilus Benson, Nick Feamster, and Dave Levin. 2015. Programming Slick Network Functions. In *SOSR*.

[3] ETSI. 2013. NFV Whitepaper. (2013). [Online at portal.etsi.org/NFV/NFV_White_Paper2.pdf; accessed 05-31-2018].

[4] Aaron Gember-Jacobson, Raajay Viswanathan, Chaithan Prakash, Robert Grandl, Junaid K., Sourav Das, and Aditya Akella. 2014. OpenNF: Enabling Innovation in Network Function Control. In *SIGCOMM*.

[5] Nodir Kodirov, Sam Bayless, Fabian Ruffy, Ivan Beschastnikh, Holger H. Hoos, and Alan J. Hu. 2018. VNF Chain Allocation and Management at Data Center Scale. In *ANCS*.

[6] Shoumik Palkar, Chang Lan, Sangjin Han, Keon Jang, Aurojit Panda, Sylvia Ratnasamy, Luigi Rizzo, and Scott Shenker. 2015. E2: A Framework for NFV Applications. In *SOSP*.

[7] Manuel Peuster, Holger Karl, and Steven van Rossem. 2017. Sonata NFV SDK. (2017). [github.com/sonata-nfv/son-emu; accessed 05-31-2018].

[8] Shriram Rajagopalan, Dan Williams, Hani Jamjoom, and Andrew Warfield. 2013. Split/Merge: System Support for Elastic Execution in Virtual Middleboxes. In *NSDI*.

[9] Vyas Sekar, Norbert Egi, Sylvia Ratnasamy, Michael K. Reiter, and Guangyu Shi. 2012. Design and Implementation of a Consolidated Middlebox Architecture. In *NSDI*.

[10] Justine Sherry, Shaddi Hasan, Colin Scott, Arvind Krishnamurthy, Sylvia Ratnasamy, and Vyas Sekar. 2012. Making Middleboxes Someone else's Problem: Network Processing As a Cloud Service. In *SIGCOMM*.