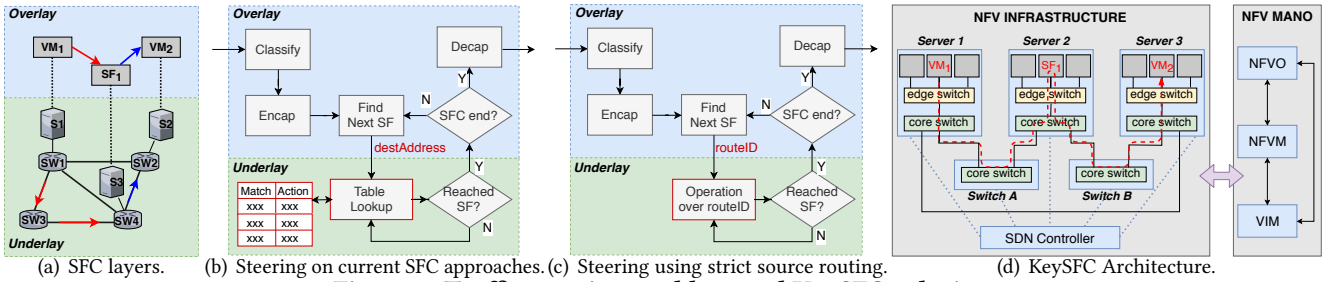
KeySFC: Agile Traffic Steering using Strict Source Routing

Cristina K. Dominicini, Gilmar Vassoler
Federal Institute of Education, Science and Technology of Espírito Santo (IFES)
Espírito Santo, Brazil
{cristina.dominicini,gilmarvassoler}@ifes.edu.br

Rodolfo Valentim, Rodolfo Villaça, Moisés R. N. Ribeiro, Magnos Martinello, Eduardo Zambon
Federal University of Espírito Santo (UFES)
Espírito Santo, Brazil



(a) SFC layers. (b) Steering on current SFC approaches. (c) Steering using strict source routing. (d) KeySFC Architecture.

Figure 1: Traffic steering problem and KeySFC solution.

CCS CONCEPTS

• **Networks** → **Network design principles**.

KEYWORDS

Service function chaining (SFC), source routing (SR).

1 INTRODUCTION

One of the main challenges in network functions virtualization (NFV) is how to dynamically steer traffic flows through a set of service functions (SFs). Fig. 1(a) shows the embedding of a service function chaining (SFC) request in a NFV Infrastructure (NFVI). The overlay layer represents logical connections between virtual machines (VMs), while the underlay layer represents connections between physical nodes.

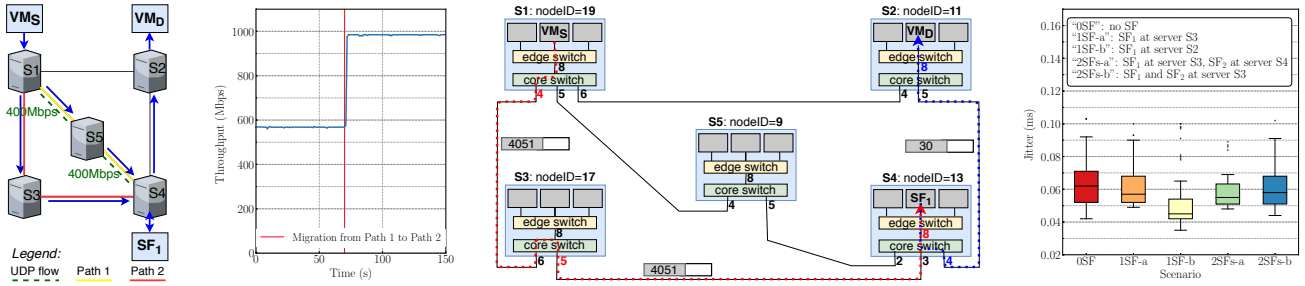
Fig. 1(b) shows how most of the current SFC solutions perform traffic steering. After classification, the SFC encapsulation carries information for defining the SFs to be executed. Then, the packet is delivered to the underlying routing mechanism (e.g., MPLS or IP) that executes per-hop table lookups based on the destination address until it reaches the first SF. At this moment, the encapsulation information is checked again to find the next SF and this loop proceeds until the chain ends, when the packet is decapsulated and delivered. The problems with this approach are: (i) the commonly adopted routing mechanisms cannot represent all the possible paths due to limited capacity of switch forwarding tables; (ii) the overhead to modify a path is high, because it may involve changes in all the nodes in the path; and (iii) the traffic engineering (TE) decisions are usually made in a decoupled way, considering placement, SFC, and routing.

These issues restrict how TE selects paths and performs load balance [3]. This work argues that the solution lies on strict source routing (SR) with algorithmic computation as shown in Fig. 1(c), in which a source specifies all forwarding nodes in the path and encapsulates this information in the packet header as a *routeID*, so each node forwards packets by executing a simple operation over this identifier [4]. Although other works proved the benefits of SR [3], very few apply SR to the SFC problem for data center networks (DCNs), and an efficient solution is yet to be presented. Related works either use a hybrid approach that still uses tables for routing [1] or cannot cope with tight performance requirements [5].

2 KEYSFC

We propose KeySFC, a traffic steering scheme that eliminates tables on the routing stage, using tables only to classify flows. To this end, we extend the idea of fabric [4] to server-based networking: (i) edge software switches (yellow in Fig. 1(d)) act as programmable SFC classifiers; and (ii) core switches (green in Fig. 1(d)), which can be software or hardware switches, perform efficient forwarding using the Residue Number System (RNS) [4]. Any SFC flow that traverses edge switches matches flow entries installed by

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SOSR '19, April 3–4, 2019, San Jose, CA, USA
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-6710-3/19/04.
https://doi.org/10.1145/3314148.3318048



(a) Test scenario. (b) Throughput at VM_D. (c) Detailed KeySFC operation using Path 2. (d) Jitter for SFC scenarios.

Figure 2: KeySFC tests for chain $VM_S \rightarrow SF_1 \rightarrow VM_D$.

a software-defined networking (SDN) Controller. Such entries rewrite Ethernet MAC addresses in order to give a new meaning for that set of bits, called VMAC, that encapsulates a *routeID* and a segment identifier. The output port in a core switch is given by the *modulo* of the *routeID* of the packet by its *nodeID*. The *nodeID*s are pairwise co-prime numbers.

The KeySFC architecture, shown in Fig. 1(d), complies with the ETSI NFV standard [2]. The management and orchestration (MANO) block is composed by a virtual infrastructure manager (VIM), a virtual network function manager (VNFM), and a NFV Orchestrator (NFVO). The NFVI is composed by servers, switches, and an SDN Controller. The NFVO is responsible for TE, sending placement decisions to the VIM and SFC decisions to the SDN Controller. The later installs flow entries in edge switches to tag and steer SFC traffic. No entry is installed in core switches. Also, in contrast to traditional SFC solutions that focus on network-centric DCNs, KeySFC supports server-centric and hybrid DCNs, in which directly connected servers execute forwarding tasks.

We implemented KeySFC in a DCN prototype orchestrated by OpenStack. Edge and core switches were implemented as OvS bridges, and the RNS algorithm was implemented in the datapath module of core switches. To isolate the effects of the SFC mechanism, the SFs only receive packets and return them to the network. The topology, shown in Fig. 2(a), is composed by 5 servers with 1*Gbps* Ethernet NICs.

The first test in Fig. 2(a) aims to dynamically provide the maximum bandwidth for SFC $VM_S \rightarrow SF_1 \rightarrow VM_D$. There are two paths with length 2 that connect VM_S to SF_1. Initially, Path 1 is selected and presents a concurrent UDP traffic. VM_S sends a TCP flow to VM_D using `iperf` that uses all the available bandwidth. At 60*s*, the TE migrates the SFC to Path 2, causing the throughput at VM_D to increase from 560*Mbps* to 980*Mbps*, as shown in Fig. 2(b). The original flow entry at the edge switch of S1 matches the source IP address of VM_S and embeds VMAC1 for Path 1 with the action: "DstMAC=VMAC1; Output to core switch". To migrate the segment $VM_S \rightarrow SF_1$ to Path 2, the SDN Controller only modifies the field "DstMAC=VMAC2" in this entry to embed the new route through Path 2. On the other hand, in traditional SFC schemes, the path migration may involve changing flow entries in all the hops along the old and the new paths, which may lead to consistency problems and packet loss.

Fig. 2(c) details KeySFC operation when Path 2 is selected. The SDN controller assigns the *nodeID*s to servers (9, 11, 13, 17, 19), calculates $routeID1 = 4051$ for segment $VM_S \rightarrow SF_1$, and $routeID2 = 30$ for $SF_1 \rightarrow VM_D$. Then, it installs flow entries at the edge switches of S1 and S4 for encapsulation of $routeID1$ and $routeID2$, respectively. Also, it installs an entry at the edge switch of S2 for decapsulation. Then, the following *modulo* operations are executed in the packets of segment 1 that pass through core switches: in S1, $< 4051 >_{19} = 4$; in S3, $< 4051 >_{17} = 5$; and, in S4, $< 4051 >_{13} = 8$. Similarly, for segment 2: in S4, $< 30 >_{13} = 4$; and, in S2, $< 30 >_{11} = 8$.

In the test of Fig. 2(d), we compare jitter in a scenario with no SF ("0SF") to scenarios with one SF ("1SF-a" and "1SF-b") and two SFs ("2SFs-a" and "2SFs-b"). VM_S sends a 200*Mbps* UDP traffic to VM_D using path S_1-S_3-S_4-S_2. Even when two SFs are added in the chain, there is no significant degradation in jitter, which is small. Also, the distributions can be considered statistically equivalent for all scenarios, which can be explored by jitter sensitive applications.

3 CONCLUSION

KeySFC main contributions are: (i) use of RNS-based strict SR in core nodes to reduce control signaling, jitter, and latency; (ii) agile SFC path migration, demonstrated by an OpenStack prototype; and (iii) capacity to exploit all existing TE paths to maximize throughput [3]. Future works include performance tests in large DCNs and offload to SmartNICs with P4.

ACKNOWLEDGMENTS

This study was financed by CAPES (Finance Code 001), CNPq, FAPES, CTIC, RNP, and EU Horizon 2020 (688941-FUTEBOL).

REFERENCES

[1] F. Clad et al. 2018. *Segment Routing for Service Chaining.* Internet-Draft. IETF.
[2] ETSI ISG. 2014. NFV 002 V1.2.1. Network Functions Virtualisation (NFV); Architectural Framework.
[3] S. A. Jyothi et al. 2015. Towards a Flexible Data Center Fabric with Source Routing. In *SOSR*. ACM, New York, NY, USA, Article 10, 8 pages.
[4] M. Martinello et al. 2014. KeyFlow: a prototype for evolving SDN toward core network fabrics. *IEEE Network* 28, 2 (2014), 12–19.
[5] Y. Ren et al. 2018. On Scalable Service Function Chaining with O(1) Flowtable Entries. In *IEEE INFOCOM*. IEEE, 702–710.