SFC Path Tracer: A Troubleshooting Tool for Service Function Chaining

Rafael Anton Eichelberger*†, Tiago Ferreto*, Sebastien Tandel† and Pedro Arthur P. R. Duarte†
* PPGCC/PUCRS, Porto Alegre, Brazil
tiago.ferreto@pucrs.br
† Hewlett Packard Enterprise, Porto Alegre, Brazil.
reichelberger@hpe.com, sebastien.tandel@hpe.com, pedro.duarte@hpe.com

***Abstract*—Service Function Chaining (SFC) is a prominent research field in networking with several proposals from industry and academia. The lack of tools to check SFC path correctness forces network operators to spend a significant effort in guaranteeing that an SFC configuration meets their intent behavior. This work presents the SFC Path Tracer, a tool for troubleshooting SFC in NFV/SDN environments. The tool enables the identification of problems in an SFC configuration through the display of the whole path traversed by packets in a given SFC path. SFC Path Tracer is agnostic to the SFC encapsulation mechanism and incurs in low overhead on the SFC architecture.**

***Index Terms*—SFC, troubleshooting, trace, NFV, SDN**

I. Introduction

Common network topologies contain several middleboxes, proportional to the number of switches and routers [1]. These network devices perform a critical role in the deployment of network services, such as: firewalls, IDS (Intrusion Detection Systems), DPI (Deep Packet Inspection), Proxies, NAT (Network Address Translation), etc. Managing these devices is complex and require specialized technical support to reach the best performance among all aggregated functionalities.

The features delivered by Software-Defined Networking (SDN) and Network Function Virtualization (NFV) have leveraged the on-the-fly configuration of network functions interconnections known as Service Function Chaining (SFC), enabling then the addition or removal of those functions according to network requirements. SFC can be achieved following many proposals [2]–[4] from industry and academia. It is possible to implement SFC using a variety of technologies including SDN/NFV or even a new protocol stack.

Given the dynamic nature of SFC, its deployment may involve multiple configuration steps as functions might reside in different hosts or networks. Hence, configuration mistakes lead the systems to several erroneous behaviors ranging from wrong forwarding decisions to packet drop. Besides that, the lack of troubleshooting tools makes operators spend great efforts to ensure that the network meets its intent behavior [5]. To overcome this problem, researchers and practitioners proposed the SFC Operation, Administration and Maintenance Framework (SFC-OAM) [6], a Internet draft that discusses and proposes tools to aid SFC operation.

One of SFC-OAM discussions evolve around tracing tools. In traditional networks, tracing tools assist the detection of misbehavior such as packet drops and unwanted network hops. However, current tracing tools are unable to give precise information in SFC environments because these environments employ ad-hoc forwarding mechanisms that partially breaks their tracing techniques. Therefore, the main contribution of this work is the design of SFC Path Tracer, a troubleshooting tool for SFC environments that enables the visualization of the trace of network packets in the SFC domain.

II. Background

IETF (Internet Engineering Task Force) defines Service Function Chaining (SFC) [7] as an abstract view of network functions and the order in which they need to be applied. SFC forms traffic chains among service functions regarding a specific network service. SFC is instantiated from a set of network functions or service functions (SFs), which are placed in specific locations forming a forwarding graph, known as Service Function Path (SFP). In the SFC-enabled domain Service Function Forwarders (SFFs) are responsible for forwarding packets throughout SFs in the chain.

SFC enables the creation of composite network services that consist of an ordered set of SFs (e.g., SF1 $\Rightarrow$ SF2 $\Rightarrow$ SF3) that must be applied to packets selected as a result of classification. Each SF is referenced using an identifier that is unique within an SFC-enabled domain. SFC describes a method for deploying SFs in a way that enables dynamic ordering and topological independence of SFs, as well as the exchange of metadata between its components [8].

A. SFC troubleshooting

SFC configuration involves multiple steps, such as: defining a chain, translating the intent chain into network policies, installing switch flows rules and even configuring service functions. Any misconfiguration can lead the system to erroneous behavior, with packets being forwarded to wrong SFs or even dropped along its path.

Different SFC approaches present a variety of techniques to implement dynamic function chaining which affects the complexity to properly evaluate such implementations. It is hard to evaluate or troubleshoot possible problems, regardless of the SFC technique used. If a reachability problem is detected in the chain path, it might be a massive work to find where the packet is being lost.

978-3-901882-89-0 @2017 IFIP

OAM for SFC [6] provides a reference framework for Operations, Administration and Maintenance (OAM) for SFC. It indicates the aspects of SFC that should be monitored. The monitoring element should have capabilities to monitor service function performance, forwarders and the classifier. OAM for SFC also discuss requirements for monitoring SFC path. It highlights that SFC path can be monitored using connectivity and trace functions. Packet trace function enables the detection of above common problems. Packet traces can confirm that the network traffic is traversing all configured service functions or pinpoint the problematic location when traffic does not reach its destination.

III. Related Work

Some tools and frameworks that generate packet traces may be leveraged by network operators to help identify recurrent problems in the SFC environment. SFC Traceroute [9] makes use of the metadata space from Network Service Header (NSH) [2] to store chain hop information and then generating a trace. NSH is a new protocol stack proposal to encapsulate SFC information. Tracebox [10] proposes an extension to the widely used traceroute tool, that is capable of detecting middleboxes by sending IP packets containing TCP segments with different TTL values. SDN Traceroute [11] installs high-priority rules in every switch of the network in order to allow these switches to trap probe packets and thus generate the trace.

IV. SFC Path Tracer

SFC Path Tracer provides a solution to generate packet traces in an SFC environment. It is agnostic to SFC encapsulation mechanism employed, enabling its utilization in different SFC implementations. SFC Path Tracer provides a trace which includes all network components in the SFC path.

A. Architecture

Figure 1 shows the architecture of SFC Path Tracer. *SFC Model* represents the SFC configuration including all its elements (SFF, SFs, SFP, etc). *SFC Driver* is responsible to read the SFC Model configuration and to install rules in the *Network Elements*. SFC Path Tracer watches all switch rules installation and based on that, it adds new rules to mirror network packets. The trace generation will be triggered by probe packets. Therefore, probe packets that traverse a target chain will be mirrored to the trace tool.

Probe packets will traverse a target chain as any other network packet would traverse. However, whenever a probe packet leaves a forwarder switch to its next hop, it is also mirrored to the trace tool. Probe packets can be identified in many ways, using optional header fields or encoding meaningful information in the L2 header. The probe packet is used to identify a single packet in the traffic and consequently get its trace. The use of probe packets allows a filter mechanism to restrict the steering of network packets to the SFC Path Tracer. Detailed probe packet flow can be seen in Figure 2.

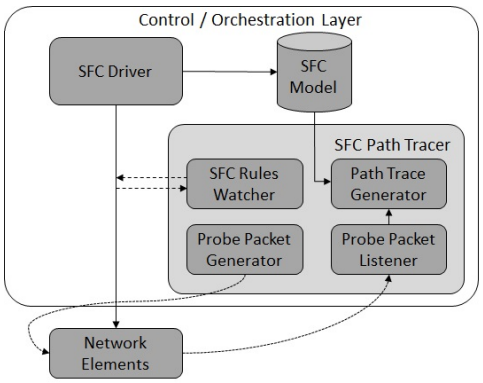


Fig. 1: SFC Path Tracer architecture

B. Implementation

SFC Path Tracer is implemented in the SDN controller layer. OpenDayLight (ODL) is the chosen platform. SFC Path Tracer is implemented as a plugin in the ODL controller under the ODL SFC project. In ODL, the OpenFlow configuration is handled by a plugin called OpenFlow Plugin. Therefore, OpenFlow Plugin is also used to install rules for tracing generation.

Since ODL and OpenFlow are used to evaluate SFC Path Tracer design, the OpenFlow channel is used to send mirrored packets to the controller where SFC Path Tracer is running. ODL abstraction layer is used to watch switch rules installation and read the SFC model configuration.

1) Probe Packet Generation: In this implementation, the probe packet is flagged by an IP header field that is not used for regular traffic and can also be used as a match field in OpenFlow rules. The used field is the 2-bit Explicit Congestion Notification (ECN). The ECN bits will trigger OpenFlow rules to send packets to the controller.

In order to generate an SFC trace for a specific chain, probe packets can be generated in many ways. The controller can artificially inject probe packet in the chain input to generate the trace. A client placed behind the Classifier, as chain input, can send probe packets. It is assumed that SFs being traced will not drop those probe packets.

2) Trace rules installation: The SFC implementations on ODL defines a pipeline of switch tables that will process network packets. SFC Path Tracer detects rules from egress OpenFlow table and adds the trace rules. The new trace rules are copies of the original egress rules, but with slightly modified matches, actions and a higher priority.

In the OpenFlow match, the ECN field is added in order to filter just probe packets. The OpenFlow action is changed to decrement IP TTL (time to live) field, write the output forward port into the OpenFlow metadata field, and add an action to forwarding the packet to the next table, defined as trace table. TTL is decremented to guarantee the trace ordering among hops. The output port is encapsulated into OpenFlow metadata in order to trace the next hop. On the trace table, the packet is matched again by ECN bits and sent to the ODL controller.

3) Probe packet listener: SFC Path Tracer must listen to packets coming into the ODL controller. From packet-in, it is possible to get information such as the switch that sent the packet to the controller. SFC Path Tracer uses this information to discover which forwarder handled the packet. From the output encapsulated in the metadata field, the next hop that the packet will be forwarded is detected. SFC Path Tracer knows the SFs and forwarders by looking the SFC data model and finding the SFF name and the SF connected to the specific switch port. Once the elements are found in the configured SFC model, SFC Path Tracer updates the packet trace. The IP identification field is used to disambiguate packets in order to handle concurrent probe packets.

It is worth noting that more information can be used from the income probe packet in the controller. For instance, if the port is not present in the SFC model, the search might be done looking for the MAC address from a specific SF.

C. Use case

Figure 2 shows a use case where a chain is configured to reach an IDS and Firewall. The client sends probe packets and, through trace rules installed by SFC Path Tracer, probe packets are mirrored to the controller in every chain hop.

The continuous line, shown on Figure 2, means the probe packet traversing the network elements, while the dotted line represents the mirrored probe packet being sent to the controller in order to generate the packet trace. Whenever the probe packet leaves a switch, it is also mirrored to the controller. The packet flow sequence is identified by numbers in the figure.

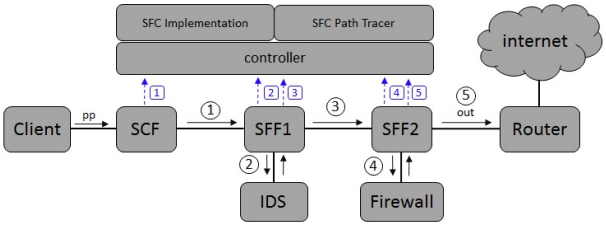


Fig. 2: SFC Path Tracer use case

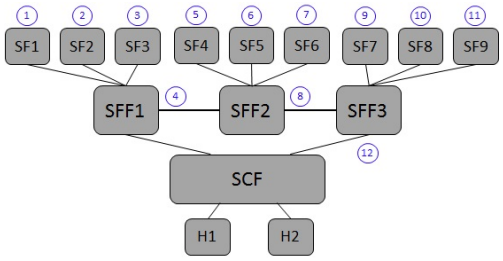
The resulted packet trace prints all SCF, SFs and SFFs reached by the probe packet. The trace output for the example in Figure 2 is $SCF \Rightarrow SFF1 \Rightarrow IDS \Rightarrow SFF1 \Rightarrow SFF2 \Rightarrow Firewall \Rightarrow SFF2 \Rightarrow Router$.

V. Evaluation

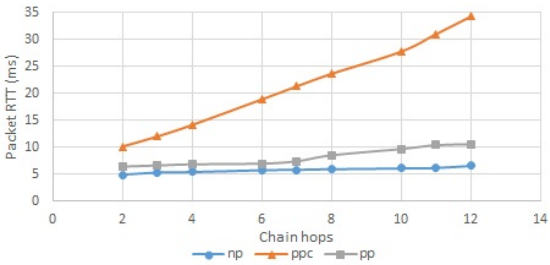
A. Probe packet delay

The SFC encapsulation technique used in the experiment is the SFCOFL2 plugin [12]. SFCOFL2 implements SFC for OpenFlow switches which are connected via layer 2 connectivity with the SFs.

Figure 3a shows the largest topology used in the evaluation. This topology is formed by a classifier and three forwarders, connecting three SFs each. In order to adjust the number of hops in the chain, SFs were removed for each measurement



(a) Experiment topology



(b) Probe packet delay

Fig. 3: Packet delay with a varying number of chain hops: normal packets (np), probe packets (pp) and probe packet with controller in the path (ppc)

until the smallest chain were formed just by *SFF1* and *SF1*. The numbers in Figure 3a represent the chain hops, which are changed to run the experiments.

Probe packets were generated using *hping3* [1] tool. This tool can create custom packets to ping devices via UDP, TCP or ICMP. In the context of the experiments, *hping3* is used to create probe packets, *i.e.* IP headers with ECN bit set to one. Besides that, the probe packet carries a UDP payload that targets a closed port. According to RFC 792, if a source cannot deliver a datagram to a protocol module or process port, the destination may send an ICMP Port Unreachable back to the source. The experiments use this message to calculate the Round-Trip Time (RTT).

In order to measure the delay, a UDP packet is sent through a unidirectional chain from $H1$ to $H2$. Figure 3b shows the RTT for normal packets (np) that were sent over the chain with ECN untouched and probe packets (pp) with ECN bits set. It was also evaluated the approach used by SDN Traceroute [11] (ppc). In this approach, probe packets are sent to the controller and forwarded back to the switches. Therefore, the controller is located in the direct path of probe packets, since they are not mirrored. The amount of time for packets traverse the chain is computed collecting RTT measurements from an average of 10 cycles of 100 pings.

The additional delay of pp, comparing with np, represents the cost of copying those packets in the switches in order to send it to the controller. Since this evaluation runs on virtual switches, this copying is notable. On hardware switches, the probe packet mirroring would have significant smaller cost.

[1] http://www.hping.org/

Although there is a small extra delay for *pp*, this delay is significant smaller comparing with SDN Traceroute approach (*ppc*). While the average of additional delay for *ppc* is around 2.3 ms per hop, for *pp* is approximately 0.3 ms per hop. This enables the possibility to flag normal network packets, generating traces for real traffic.

B. Troubleshooting evaluation

SFC Path Tracer tool is useful to troubleshoot common SFC problems such as misconfigured policies and SFs connectivity issues. For instance, while setting-up the test environment, SFC Path Tracer made explicit several errors such as bogus SFs, wrong OpenFlow rule sets installation, and dataplane misconfiguration (*e.g.*, disabled ports). With the use of SFC Path Tracer, such problems were easily identified observing the SFC trace output.

VI. Discussion

OAM for SFC [6] discusses tool gaps to perform OAM function on an SFC. The gaps are related to verifying that the connectivity exists between network elements and the continuity of elements, which is a model where OAM messages are sent periodically to validate or verify the reachability to a given SF or SFC. Performance and trace are other OAM functions analyzed. Some of these gaps can be filled by SFC Path Tracer. Table I shows this gap analyses and where SFC Path Tracer (SFC PT) is fitted. Existing tools used for network overlay does not work within the SFC environment.

TABLE I: OAM Tool gap Analysis [6].

Layer	Connectivity	Continuity	Trace	Performance
Network Overlay	Ping	BFD [13]	Traceroute	IPPM [14]
SF	SFC PT	SFC PT	SFC PT	None
SFC	SFC PT	SFC PT	SFC PT	None

A. SFC encapsulation methods

SFC Path Tracer was evaluated on ODL's SFCOFL2, an SFC implementation that stores chain identification in DSCP field of IP headers. However, SFC Path Tracer design can be applied to other SFC encapsulation methods. SFC Path Tracer was also tested using NSH. In this case, SFC Path Tracer extracts NSH field to get chain and hop information to generate the trace. In cases where SFC is encapsulated in L2 headers, MAC address fields could encode probe packets flagging information. SFC technique that relies on traditional switch routing protocols such as 802.1q (VLAN) and MPLS may not be trivial to adapt the SFC Path Tracer.

B. Service Functions compatibility

SFC Path Tracer does not require any agent from SF side. SFC Path Tracer technique allows the utilization of any IP packet with probe flag (ECN) to be traced, which can be used in most transport protocols. SFC Path Tracer just relies on SFs not dropping those packets otherwise, the probe packet flagging must be done using other fields such as MAC addresses.

SFC Path Tracer may present compatibility issues with SFs which does not keep the IP header by any reason or opens new output connections such as TCP proxies. In this case, the ECN flags from packets will be lost and the next SFF will not recognize it as a probe packet.

VII. Conclusions

This paper presented the SFC Path Tracer, a troubleshooting tool for SFC environments. SFC Path Tracer proved to be useful for identifying problems in SFC paths configuration. Packet trace information reduces the debug time by pinpointing the origin of a possible problem. The strategy of decrementing TTL field from IP packets ensures correct ordering of network elements in the trace.

SFC Path Tracer can be extended to add new features to collect other measurements and become a more generic SFC monitoring tool. SFC Path Tracer can be leveraged to measure chain performance. This can be achieved by collecting timestamps throughout a chain in order to detect congestion paths or overloaded SFs.

References

[1] V. Sekar, S. Ratnasamy, M. K. Reiter, N. Egi, and G. Shi, "The middlebox manifesto: Enabling innovation in middlebox deployment." ACM, 2011.

[2] P. Garg, P. Quinn, R. Manur, J. Guichard, S. Kumar, A. Chauhan, B. McConnell, M. Smith, C. Wright, U. Elzur, J. M. Halpern, W. Henderickx, T. Nadeau, S. Majee, D. T. Melman, K. Glavin, and P. Agarwal, "Network Service Header," Internet Engineering Task Force, Internet-Draft draft-quinn-sfc-nsh-07, Feb. 2015, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-quinn-sfc-nsh-07

[3] P. Bottorff, don.fedyk@hpe.com, and H. Assarpour, "Ethernet MAC Chaining," Internet Engineering Task Force, Internet-Draft draft-fedyk-sfc-mac-chain-01, Jan. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-fedyk-sfc-mac-chain-01

[4] D. Dolson, "Vlan service function chaining," *IETF (Internet Engineering Task Force) Internet-Draft, draft-dolson-sfc-vlan-00*, 2014.

[5] S. K. Fayaz, T. Yu, Y. Tobioka, S. Chaki, and V. Sekar, "Buzz: testing context-dependent policies in stateful networks," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, 2016.

[6] S. Aldrin, C. Pignataro, R. R. Krishnan, A. Ghanwani, and N. Akiya, "Service Function Chaining Operation, Administration and Maintenance Framework," Internet Engineering Task Force, Internet-Draft draft-aldrin-sfc-oam-framework-02, Jul. 2015, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-aldrin-sfc-oam-framework-02

[7] A. Farrel, "Service function chaining," IETF, Tech. Rep., 2013. [Online]. Available: http://datatracker.ietf.org/doc/charter-ietf-sfc/

[8] J. Halpern and C. Pignataro, "Service function chaining (sfc) architecture," IETF, RFC, 2015.

[9] X. Yang, L. Zhu, and G. Karagiannis, "SFC Trace Issue Analysis and Solutions," Internet Engineering Task Force, Internet-Draft draft-yang-sfc-trace-issue-analysis-01, Feb. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-yang-sfc-trace-issue-analysis-01

[10] G. Detal, B. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet, "Revealing middlebox interference with tracebox," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013.

[11] K. Agarwal, E. Rozner, C. Dixon, and J. Carter, "Sdn traceroute: Tracing sdn forwarding without changing network behavior," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014.

[12] "Service function chaining:lithium user facing features - opendaylight project," https://wiki.opendaylight.org/view/Service_Function_Chaining:Lithium_User_Facing_Features#SFCOFL2, (Accessed on 05/18/2016).

[13] D. Katz and D. Ward, "Bidirectional forwarding detection (bfd)," 2010.

[14] D. G. T. Almes, J. Mahdavi, M. Mathis, and D. V. Paxson, "Framework for IP Performance Metrics," RFC 2330, Mar. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc2330.txt