Recent Developments in Service Function Chaining

Patricia Horvath, Kilian Holzinger, Henning Stubbe*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: patricia.horvath@tum.de, holzinger@net.in.tum.de, stubbe@net.in.tum.de

***Abstract*—A network's infrastructure consists of multiple network functions some of which may include distinct intermediate steps that need to be applied in a particular order.**

In order to ensure the right order of application while still striving for goals such as improving the network's flexibility, improving its independence from physical structure and reducing infrastructure complexity, Service Function Chaining (SFC) is applied.

The resulting service function chains are comprised of an ordered set of network functions that are applied to data packets handled in the network. This technique is beneficial for use in fixed broadband networks, where it is deployed behind the broadband network gateway, as well as in mobile networks for optimization of services such as TCP, and in data centers. This paper gives an in-depth overview over the basic functionality of network function chains and compares the advantages and drawbacks of various implementations for different use cases.

***Index Terms*—network architecture, service function chaining, monitoring, load balancing, service function**

1. Introduction

The infrastructure of a network heavily relies on network services which are applied to the data packets that are being transmitted within the network. These network services are comprised of multiple network functions which may be hardware components or implemented as virtual components. Examples for such network services include firewalls, load balancing, parental control, network address translation and deep packet inspection to name a few.

Grouping these service functions in service function chains (SFC) ensures that the order in which the functions are applied remains unchanged. More specifically, using SFC also allows for an optimization of the network's configuration flexibility, for more independence from its hardware implementation and for less complexity, as the particular order of a set of network functions can be adjusted dynamically to adapt to any necessary changes or external influences, for example in the case of a malicious attack on the network.

In this paper, use cases of the SFC architecture as well as present available implementations, especially in the field of the Internet of Things (IoT), are discussed, while also giving an overview over possible challenges such as monitoring and load balancing and highlighting current trends. The remaining content of this paper is organized as follows: Section 2 provides the background on SFC architectures and describes their basic functionalities. Section 3 discusses use cases and available implementations of SFC in fixed broadband networks, in mobile networks and in data centers as well, while Section 4 examines possible challenges for SFC that are encountered by different implementations and their solutions. Finally, Section 5 summarizes the above content and concludes this paper.

2. Architectural Theory of SFC

SFCs are comprised of an ordered set of multiple service functions (SF) which are applied to packets and specify if packets should be, for example, directed to a firewall or a caching engine as described in RFC 7665 [1]. Additionally, the mechanism to express results of applying a more granular policy and constraints to the abstract constraints is called Service Function Path (SFP), some of which may be vague. It is noteworthy that an SF can be part of multiple SFCs and SFPs.

The logical core components of an SFC are classifiers, Service Function Forwarders (SFFs), the SFs themselves, and SFC proxies which are interconnected by SFC encapsulation which, although is not a transportation encapsulation itself, is the mechanism that allows for a SFP selection while sharing metadata or context information if required and carrying explicit information to identify the SFP.

The service functions a SFP contains may also be altered and result in selecting a new SFP or an update of the related metadata, which is the result of a process called "reclassification". If both of the above occur, this process is specified as "branching". Reclassification and branching are especially needed, if an attack on the traffic within the network has been detected. In this case, traffic can be rerouted to a new SF, e.g. a firewall, to be able to enforce security policies. Keep also in mind that any network transport may be used to carry SFC encapsulated traffic [1].

Beyond that, a tool called Service Function Forwarder is needed to control the flow of packets within the network. A SFF's responsibilities include forwarding packets and frames to one or more SFs associated with a given SFF by utilizing the information transmitted in the SFC encapsulation, terminating SFPs when all required SFs of a SFC have been passed and maintaining the flow state of an SFF, as they may be stateful. A SFC can be abstracted as a graph, where each node signifies a SF.

As such, SFCs may contain cycles and may be unidirectional, in which case the SFC has an ordered path,

doi: 10.2313/NET-2021-05-1_07

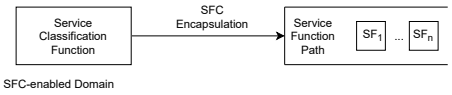

Figure 1: A diagram of a SFP as described in RFC 7665 [1].

or bidirectional, in which case the traffic is symmetric. Furthermore it is necessary to differentiate between SFC-aware and SFC-unaware SFs as SFC-unaware SFs need a SFC Proxy to function as a gateway to the SFC encapsulation by using a local attachment circuit to deliver packets to SFC-unaware SFs. In addition to enabling SFC-unaware SFs to be able to function in a SFC architecture, SFC proxies are also handling the removal of SFC encapsulation.

The SFC-enabled domain contains all SFC Proxies and their corresponding SFC-unaware SFs. Furthermore, the SFC control plane is responsible for managing the resources of the SFC architecture. The responsibilities entail constructing SFPs, translating SFCs to forwarding paths and propagating path information to participating nodes, e.g. SFs.

Further utensils for SFC Operations, Administration and Maintenance (OAM) are subject to ensure fault detection and isolation, as well as performance management and are either in-band, meaning OAM packets are handled the same way user packets are handled, or out-of-band, meaning the tools function in a layer beyond the actual data plane [1].

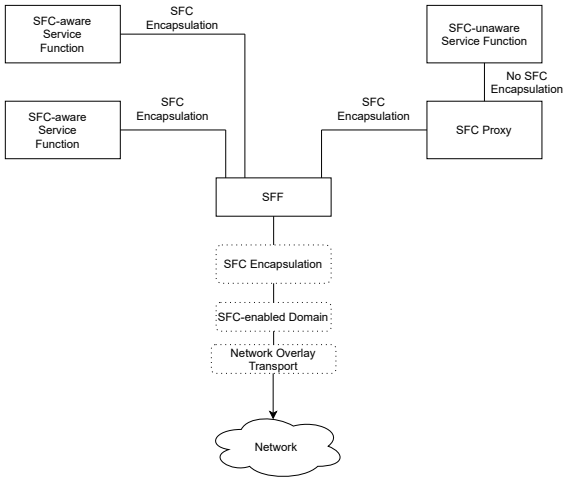

Figure 2: A diagram of the components of the SFC architecture after the initial classification as described in RFC 7665 [1].

3. Analysis of SFC

The following section introduces the reader to use cases of SFC and gives examples for available implementations.

3.1. Use Cases of SFC

The SFC architecture is used in multiple environments which include fixed broadband networks, data centers [2] and cloud customer premises equipment [3].

Fixed broadband network are accessed by their users commonly using technologies such as DSL, Ethernet or Passive Optical Networks, whereas in mobile networks, the Internet is accessed by using SFs (sometimes referred to as "enablers") [2].

In fixed broadband networks, SFC is responsible for services such as Deep Packet Inspection (DPI), NAT44, which is an extension to Network Address Translation, DS-Lite, a tool allowing applications which use IPv4 to access the internet via IPv6, NPTv6, a technology allowing IPv6 to IPv6 Network Prefix Translation, parental control, firewall, load balancer and cache. In mobile networks many SFs are implemented in the Gi interface, which is a reference point between the Gateway GPRS Support Node and an external Public Data Network [4]. Examples for SFs encompass functions such as DPI, billing and charging, TCP optimization, web optimization and video optimization. The answer as to why SFC are used is being able to facilitate resource optimization and a seamless service switchover from one network to the other. Additionally, SFCs are also used to facilitate addressing convergence needs [2]. These network services are comprised of multiple network functions which may be hardware components or implemented as virtual components. Using SFCs is also beneficial in data centers. Traffic flow in data centers can be categorized as either north-south traffic, where traffic originates from outside the data center, or as east-west traffic, where all traffic originates from within the data center [2].

A simple example for north-south traffic is given when a remote worker accesses a specific data center server resulting in incoming traffic for the data center. As you can see, north-south traffic generates the need for traffic analysis, identification of application and its users, authorization of transactions and mitigation and elimination of security threats since communication partners are outside of the data center and as such unknown and potentially dangerous. To be able to fulfill these needs, SFCs are implemented in permutations of service nodes through which the traffic has to flow. Permutations of the service nodes are necessary because not every present service function is suitable to be applied to a certain type of traffic and vice versa. For example, certain SFs are not able to be applied to virtual private network traffic. Furthermore within the network of a data center, SFCs can be either classified as Access SFCs or as Application SFCs depending on the destination of the data packets to whom the SFCs are applied, as highlighted in [2]. Access SFCs assist traffic which enters and leaves the data center, thus making such SFCs suitable for north-south traffic, whereas Application SFCs service traffing destined to applications, which is suitable for east-west traffic.

The following example helps illustrate east-west traffic: In a three-tiered architecture, requests come to the webserver which trigger interaction with the application servers. In turn, interaction with the database servers is triggered and SFs are then applied to enforce security policies between the tiers, while monitoring SFs enable visibility into the application traffic [2]. Along with the above mentioned use cases, there is also the scenario of a SFC architecture consisting of centralized value-added SFs, which are configured by subscribers and enabled in the network side, while the subscriber side box is limited

to only Layer 2 and Layer 3 functionalities, which is the case when using Cloud Customer Premises Equipment (CPE). In this case, Cloud CPE translates the subscribers' service requests into SFCs [3]. The ability to reconfigure SFPs as needed allows for pay-per-use and on-demand server-side services.

3.2. Available Implementations

Various frameworks and policies exist where SFC is used. The following section highlights some of these to give some insight into networks with SFC architecture.

One growing field where SFC can be applied to is IoT. In [5], the authors propose using the SFC orchestration system PRSFC-IoT to meet IoT providers' demand of being able to process both the increasing traffic amount and the increasing amount of varied IoT traffic requirements. SFC orchestration can be beneficial to optimize performance and resource consumption. To fulfill the above goals, the authors suggest the use of PRSFC-IoT considering that it encompasses functionalities that enable meeting deadlines while guaranteeing a stable packet rate including efficient resource management.

SFCs are also used in edge computing where high flexibility and low latency are important in regards to the quick execution of various functionalities. In [6] an online orchestration framework for cross-edge SFCs, which strives to improve cost efficiency by improving both resource provisioning and traffic routing, is introduced. In this approach, SFs are split into edge clouds to mitigate the cost of dynamically launching new SFs.

To summarize, SFC is highly beneficial to environments where networks need to be able to adapt to changes quickly and dynamically.

4. Challenges in SFC Architectures

This section discusses some of the prevalent challenges in various SFC architectures and possible solutions.

4.1. Flexibility

Scalability, which results in more flexibility, is an import requirement for SFC architectures as it is needed to be able to accommodate changing demands of the user side, for example in data centers where the number of clients accessing server-side services may change drastically. When using static SFCs, there may be a need to readjust the service nodes by adding or removing some to be able to accommodate new obligations. In other words, static SFCs cannot scale well. Furthermore, SFCs cannot pass metadata which is needed to enforce policies consistently across all of the network. Beyond the above problem, physical and static SFC mechanisms cannot be mixed with virtual and dynamic SFC mechanisms which is problematic as one cannot use the benefits of both implementations at the same time [2].

Another issue regarding flexibility occurs when managing large scale, multi-region data centers with multiple operational teams. In [7], the authors propose an implementation of hierarchical SFC using OpenDaylight platform in a SDN environment to fulfill this demand. The

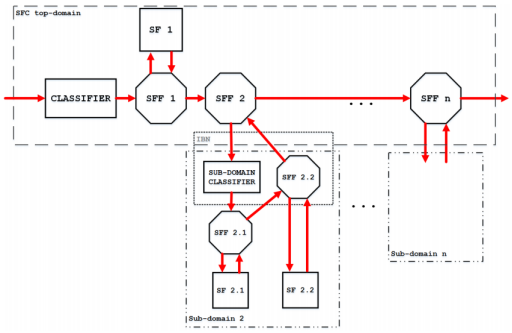

Figure 3: A diagram of a hierarchical SFC architecture [7].

OpenDaylight platform is an open-source project which enables modularity, flexibility, scalability in a multiprotocol SDN controller infrastructure.

4.2. Monitoring

Another functionality which plays a big role in SFC deployments is the monitoring of traffic as it is needed for quality and congestion control as well as anomaly detection and capacity planning among other things.

As there is no standard method of monitoring and detecting failure, the authors in [8] have implemented an alarm-based monitoring with a focus on high availability in SFC for cloud environments using the OpenStack project API and the ODL driver in Tacker. In general, a Virtual Network Function (VNF) Manager's tasks include VNF instantiation, updating and upgrading software, modification, collecting performance measurement results as well as information regarding events and faults, terminating instances when needed and lastly managing the integrity of the VNF instance. Tacker, which is a Generic VNF Manager and NFV Orchestrator and OpenStack project based on ETSI MANO Architectural Framework, deploys and operates Network Services and VNFs and can add functions such as monitoring and auto healing [9]. As Tacker needs a monitoring tool to be able to perform the above introduced functionalities, an alarm-based monitoring tool is needed. Due to alarms being related to hardware resources such as CPU and memory usage, Tacker is used in combination with Openstack Ceilometer, which is responsible for collecting measurements within OpenStack [8].

VNFs are built in such a way that each VNF can be identified with an unique VNF ID which aids in VNF failure detection. Additionally, SFC reliability can be achieved by means of using Ceilometer as a monitoring driver which sends a notification message to the SFC driver after analyzing the alarm message [8].

Another approach to improve network monitoring with the goal for this implementation to be adjustable to recent and future technologies is proposed in [10]. The birth of trends such as network function virtualization (NFV), software-defined networking (SDN) and cloud computing, led to the demand to monitor different planes inside a network, such as the (virtual) infrastructure plane, the user plane and the service plane, having grown considerably. As proposed in [10], using a monitoring framework, that

is able to control monitoring in a network as well as in a cloud infrastructure, is favorable to combine the operators' need to monitor end user subscribers' traffic as well as tenant customers' traffic.

As a result, monitoring is not hindered by cloud, physical and virtual boundaries anymore. Moreover, the authors in [10] also recommend classifying traffic flows based on Layer 4-7 information and controlling these flows with a different monitoring approach with the assistance of the new network service header (NSH) and the metadata it carries. The monitoring process of the above mentioned framework is logically centralized. Furthermore, the framework enables for probes to be consolidated based on monitoring requests and for the monitoring of rule consolidations which possibly facilitates the use of monitoring resources in a more efficient manner. Moreover, the authors suggest using Layer 4-7 information to use classifiers to be able to mark packets that pass along the SFPs. The classifier then has the ability to modify the packet's NSH.

In addition, the authors in [10] introduce three possible implementations for the markers: Using coloring markers, which requires a new metadata header that is capable of carrying the color of the packet and the point of time when the marking was done, or timestamp markers, which entails the NHS to carry a new metadata type-length-value that is used to save the timestamps from each probe, and finally, using interception markers, where the intercept metadata is used to gather parts of the packet headers while on their packet data paths. Also, the framework is able to use SFs as well to re-classify or re-mark packets and to modify the metadata field of the NSH. Further, the authors in [10] discuss the classifier function to be adjustable in regards to the deep packet inspection (DPI) function being applied at all. If there is no DPI function, a separate classifier is responsible for a shallow packet inspection, meaning a Layer 2-4 inspection. Otherwise, there is the possibility of using indirect DPI-classifier communication, direct DPI-classifier communication or using an integrated DPI-classifier, where the classifier is part of the DPI engine and the Layer 4-7 classification and marking of the packets is done by the same entity.

Yet another approach to improve monitoring is introduced in [11] which proposes employing in-band network telemetry optimization for NFV service chain monitoring, specifically using a scalable telemetry system called IntOpt that uses active probing, thus making this technique especially effective in dynamic service chaining architectures. Furthermore, this telemetry system also enables specifying monitoring requirements for individual service chains, which are mapped to telemetry item collection jobs. A SDN controller creates the minimal number of monitoring flows needed to monitor the deployed service chains as per the telemetry demands. Moreover, the simulated annealing based random greedy metaheuristic (SARG) is utilized to minimize the overhead caused by active probing and collecting of telemetry items. The IntOpt controller then determines the set of optimal monitoring flows which minimize the total overhead of the network monitoring through use of the SARG approach. In this way, the optimal probing frequency as well as the total number of telemetry items to be monitored for each link in order to cover all service flows with minimal overhead at the data plane as well at the controller is calculated. Reducing the overhead caused by the monitoring allows for dynamic service chaining architectures to function more efficiently as less resources are pointlessly used which can the be rather used by other important tasks so the network retains its fast response time. Next, the the proper telemetry sources, forwarders as well as sinks are calculated by the controller, thus filling in the flow tables. The authors have ascertained that using the above heuristic considerably reduces the total monitoring overhead, including the delays induced by the telemetry operations. Moreover, [11] explains that such a systematic technique can be incorporated with the existing monitoring frameworks to achieve high scalability without losing the generality and expressiveness of the systems.

4.3. Load Balancing

Another important issue that has to be accounted for is load balancing for optimizing response times and making the network work more efficiently. As shown in [12], a joint network and server load balancing algorithm for chaining VNFs is proposed for this purpose. As network load balancing and server load balancing, e.g. in data center environments, are both two separate issues that need to be addressed, the authors propose an algorithm called Nearest First and Local-Global Transformation (NF-LGT) which can address both issues. The above algorithm consists of two phases that are executed consecutively. The first phase involves constructing service chains by a greedy strategy which considers both network latency and server latency. The strategy encompasses choosing the VNF whose latency from the current location is the smallest as the next destination, repeating this process iteratively, until the service chain includes all required NFs. Afterwards, the second phase commences which involves applying a searching technique to improve the result of phase 1. This is accomplished by attempting to find a better service chain, in regards to possessing smaller latency, by replacing a selected VNF with another candidate and swapping the order of VNFs in the SFC. The authors propose using a SDN/OpenFlow concept to implement the above explained algorithm, specifically separating the control plane and data plane from each other. To provide evidence for the superiority of this approach, [12] shows the results of benchmarking tests demonstrating that NF-LGT improves the system bandwidth utilization by up to 45%.

In conclusion, to be able to fully be advantageous, especially in a dynamic context where high and fast adaptability is needed, the main issues in SFC architectures that need to be tackled are monitoring and load balancing as described above. Please keep in mind that the above highlighted implementations are merely an overview over possible solutions and that research in regards to these matters is still being conducted while also touching on adjacent topics that surpass the limits of this paper.

5. Conclusion

In this paper, I presented the fundamentals of service function chains and their composition. SFCs consist of ordered service functions that are consecutively applied

to the frames that are being passed through the network. These functions include vital services such as firewalls and caching. SFCs are hugely beneficial to (mobile) networks as well as data centers in particular because they improve latency and promote efficient resource management, modularity and scalability, which are indispensable attributes for managing high volumes of traffic.

Indeed, service function chaining including network function virtualization is because of its dynamic properties of high value for many different application implementations but using service function chains also comes with challenges that need to be addressed to achieve the full potential of the above-mentioned benefits in dynamic environments. In particular, optimizing the load balancing of traffic is necessary, even more so to be able to process the aforementioned high volumes of traffic, to be able to retain response times that are as short as possible, as well as being able to efficiently monitor the network traffic, which is an imperative requisite to be able to properly analyze traffic in order to adapt accordingly to any changes.

References

[1] J. Halpern and C. Pignataro, "Service function chaining (sfc) architecture," Internet Requests for Comments, RFC Editor, RFC 7665, October 2015. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7665.txt

[2] S. Kumar, M. Tufail, S. Majee, C. Captari, and S. Homma, "Service function chaining use cases in data centers," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-sfc-dc-use-cases-06, February 2017, http://www.ietf.org/internet-drafts/draft-ietf-sfc-dc-use-cases-06.txt. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-sfc-dc-use-cases-06.txt

[3] W. Liu, H. Li, O. Huang, M. Boucadair, N. Leymann, Q. Fu, Q. Sun, C. Pham, C. Huang, J. Zhu, and P. He, "Service function chaining (sfc) general use cases," Working Draft, IETF Secretariat, Internet-Draft draft-liu-sfc-use-cases-08, September 2014, http://www.ietf.org/internet-drafts/draft-liu-sfc-use-cases-08.txt. [Online]. Available: http://www.ietf.org/internet-drafts/draft-liu-sfc-use-cases-08.txt

[4] J. Korhonen, J. Soininen, B. Patil, T. Savolainen, G. Bajko, and K. Iisakkila, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)," Internet Requests for Comments, RFC Editor, RFC 6459, January 2012.

[5] J. Wang, H. Qi, K. Li, and X. Zhou, "PRSFC-IoT: A Performance and Resource Aware Orchestration System of Service Function Chaining for Internet of Things," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1400–1410, 2018.

[6] Z. Zhou, Q. Wu, and X. Chen, "Online Orchestration of Cross-Edge Service Function Chaining for Cost-Efficient Edge Computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1866–1880, 2019.

[7] A.-V. Vu and Y. Kim, "An Implementation of Hierarchical Service Function Chaining using OpenDaylight Platform," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. IEEE, 2016, pp. 411–416.

[8] L. Yang, D. Van Tung, M. Kim, and Y. Kim, "Alarm-based Monitoring for High Availability in Service Function Chain," in *2016 International Conference on Cloud Computing Research and Innovations (ICCCRI)*. IEEE, 2016, pp. 86–91.

[9] S. Ramaswamy and Brocade. (2015) Tacker: VNF Lifecycle Management and Beyond. IETF. [Online]. Available: https://www.ietf.org/proceedings/93/slides/slides-93-nfvrg-25.pdf

[10] M. Shirazipour, H. Mahkonen, M. Xia, R. Manghirmalani, A. Takacs, and V. S. Vega, "A Monitoring Framework at Layer4–7 Granularity Using Network Service Headers," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. IEEE, 2015, pp. 54–60.

[11] D. Bhamare, A. Kassler, J. Vestin, M. A. Khoshkholghi, and J. Taheri, "IntOpt: In-band Network Telemetry Optimization for NFV Service Chain Monitoring," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[12] M.-T. Thai, Y.-D. Lin, and Y.-C. Lai, "A Joint Network and Server Load Balancing Algorithm for Chaining Virtualized Network Functions," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.