Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2015

J. Halpern, Ed.
Ericsson
C. Pignataro, Ed.
Cisco
March 5, 2015

Service Function Chaining (SFC) Architecture
draft-ietf-sfc-architecture-06

Abstract

This document describes an architecture for the specification, creation, and ongoing maintenance of Service Function Chains (SFC) in a network. It includes architectural concepts, principles, and components used in the construction of composite services through deployment of SFCs, with a focus on those to be standardized in the IETF. This document does not propose solutions, protocols, or extensions to existing protocols.

Table of Contents

1. Introduction

The delivery of end-to-end services often requires various service functions. These include traditional network service functions such as firewalls and traditional IP Network Address Translators (NATs), as well as application-specific functions. The definition and instantiation of an ordered set of service functions and subsequent

'steering' of traffic through them is termed Service Function Chaining (SFC).

This document describes an architecture used for the creation and ongoing maintenance of Service Function Chains (SFC) in a network. It includes architectural concepts, principles, and components, with a focus on those to be standardized in the IETF. Service function chains enable composite services that are constructed from one or more service functions.

An overview of the issues associated with the deployment of end-to-end service function chains, abstract sets of service functions and their ordering constraints that create a composite service and the subsequent "steering" of traffic flows through said service functions, is described in [I-D.ietf-sfc-problem-statement].

The current service function deployment models are relatively static, coupled to network topology and physical resources, greatly reducing or eliminating the ability of an operator to introduce new services or dynamically create service function chains. This architecture presents a model addressing the problematic aspects of existing service deployments, including topological independence and configuration complexity.

1.1. Scope

This document defines the architecture for Service Function Chaining (SFC) with minimum requirements on the physical topology of the network, as standardized in the IETF. In this architecture packets are classified on ingress for handling by the required set of Service Functions (SFs) in the SFC-enabled domain and are then forwarded through that set of functions for processing by each function in turn. Packets may be re-classified as a result of this processing.

The architecture described in this document is independent of the planned usage of the network and deployment context and thus, for example, is applicable to both fixed and mobile networks as well as being useful in many Data Center applications.

The architecture described herein is assumed to be applicable to a single network administrative domain. While it is possible for the architectural principles and components to be applied to inter-domain SFCs, these are left for future study.

1.2. Assumptions

The following assumptions are made:

- There is no standard definition or characterization applicable to all SFs, and thus the architecture considers each SF as an opaque processing element.
- There is no global or standard list of SFs enabled in a given administrative domain. The set of SFs enabled in a given domain is a function of the currently active services which may vary with time and according to the networking environment.
- There is no global or standard SF chaining logic. The ordered set of SFs that needs to be applied to deliver a given service is specific to each administrative entity.
- The chaining of SFs and the criteria to invoke them are specific to each administrative entity that operates an SF-enabled domain.
- Several SF chaining policies can be simultaneously applied within an administrative domain to meet various business requirements.
- The underlay is assumed to provide the necessary connectivity to interconnect the SFFs, but the architecture places no constraints on how that connectivity is realized other than it have the required bandwidth, latency, and jitter to support the SFC.
- No assumption is made on how FIBs and RIBs of involved nodes are populated.
- How to bind traffic to a given SF chain is policy-based.

1.3. Definition of Terms

Network Service: An offering provided by an operator that is delivered using one or more service functions. This may also be referred to as a composite service. The term "service" is used to denote a "network service" in the context of this document.

Note: Beyond this document, the term "service" is overloaded with varying definitions. For example, to some a service is an offering composed of several elements within the operator's network, whereas for others a service, or more specifically a network service, is a discrete element such as a "firewall". Traditionally, such services (in the latter sense) host a set of service functions and have a network locator where the service is hosted.

Classification: Locally instantiated matching of traffic flows against policy for subsequent application of the required set of network service functions. The policy may be customer/network/service specific.

Classifier: An element that performs Classification.

Service Function Chain (SFC): A service function chain defines an ordered set of abstract service functions (SFs) and ordering constraints that must be applied to packets and/or frames and/or flows selected as a result of classification. An example of an abstract service function is "a firewall". The implied order may not be a linear progression as the architecture allows for SFCs that copy to more than one branch, and also allows for cases where there is flexibility in the order in which service functions need to be applied. The term service chain is often used as shorthand for service function chain.

Service Function (SF): A function that is responsible for specific treatment of received packets. A Service Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers). As a logical component, a Service Function can be realized as a virtual element or be embedded in a physical network element. One or more Service Functions can be embedded in the same network element. Multiple occurrences of the Service Function can exist in the same administrative domain.

One or more Service Functions can be involved in the delivery of added-value services. A non-exhaustive list of abstract Service Functions includes: firewalls, WAN and application acceleration, Deep Packet Inspection (DPI), LI (Lawful Intercept), server load balancing, NAT44 [RFC3022], NAT64 [RFC6146], NPTv6 [RFC6296], HOST_ID injection, HTTP Header Enrichment functions, TCP optimizer.

An SF may be SFC encapsulation aware, that is it receives and acts on information in the SFC encapsulation, or unaware, in which case data forwarded to the SF does not contain the SFC encapsulation.

Service Function Forwarder (SFF): A service function forwarder is responsible for forwarding traffic to one or more connected service functions according to information carried in the SFC encapsulation, as well as handling traffic coming back from the SF. Additionally, a service function forwarder is responsible for delivering traffic to a classifier when needed and

supported, transporting traffic to another SFF (in the same or different type of overlay), and terminating the SFP.

Metadata: provides the ability to exchange context information between classifiers and SFs and among SFs.

Service Function Path (SFP): The SFP provides a level of indirection between the fully abstract notion of service chain as a sequence of abstract service functions to be delivered, and the fully specified notion of exactly which SFF/SFs the packet will visit when it actually traverses the network. By allowing the control components to specify this level of indirection, the operator may control the degree of SFF/SF selection authority that is delegated to the network.

SFC Encapsulation: The SFC Encapsulation provides at a minimum SFP identification, and is used by the SFC-aware functions, such as the SFF and SFC-aware SFs. The SFC Encapsulation is not used for network packet forwarding. In addition to SFP identification, the SFC encapsulation carries metadata including data plane context information.

Rendered Service Path (RSP): The Service Function Path is a constrained specification of where packets assigned to a certain service function path must go. While it may be so constrained as to identify the exact locations, it can also be less specific. Packets themselves are of course transmitted from and to specific places in the network, visiting a specific sequence of SFFs and SFs. This sequence of actual visits by a packet to specific SFFs and SFs in the network is known as the Rendered Service Path (RSP). This definition is included here for use by later documents, such as when solutions may need to discuss the actual sequence of locations the packets visit.

SFC-enabled Domain: A network or region of a network that implements SFC. An SFC-enabled Domain is limited to a single network administrative domain.

SFC Proxy: Removes and inserts SFC encapsulation on behalf of an SFC-unaware service function. SFC proxies are logical elements.

2. Architectural Concepts

The following sections describe the foundational concepts of service function chaining and the SFC architecture.

Service Function Chaining enables the creation of composite (network) services that consist of an ordered set of Service Functions (SF)

that must be applied to packets and/or frames and/or flows selected as a result of classification. Each SF is referenced using an identifier that is unique within an SF-enabled domain.

Service Function Chaining is a concept that provides for more than just the application of an ordered set of SFs to selected traffic; rather, it describes a method for deploying SFs in a way that enables dynamic ordering and topological independence of those SFs as well as the exchange of metadata between participating entities.

2.1. Service Function Chains

In most networks, services are constructed as abstract sequences of SFs that represent SFCs. At a high level, an SFC is an abstracted view of a service that specifies the set of required SFs as well as the order in which they must be executed. Graphs, as illustrated in Figure 1, define an SFC, where each graph node represents the required existence of at least one abstract SF. Such graph nodes (SFs) can be part of zero, one, or many SFCs. A given graph node (SF) can appear one time or multiple times in a given SFC.

SFCs can start from the origination point of the service function graph (i.e., node 1 in Figure 1), or from any subsequent node in the graph. SFs may therefore become branching nodes in the graph, with those SFs selecting edges that move traffic to one or more branches. An SFC can have more than one terminus.

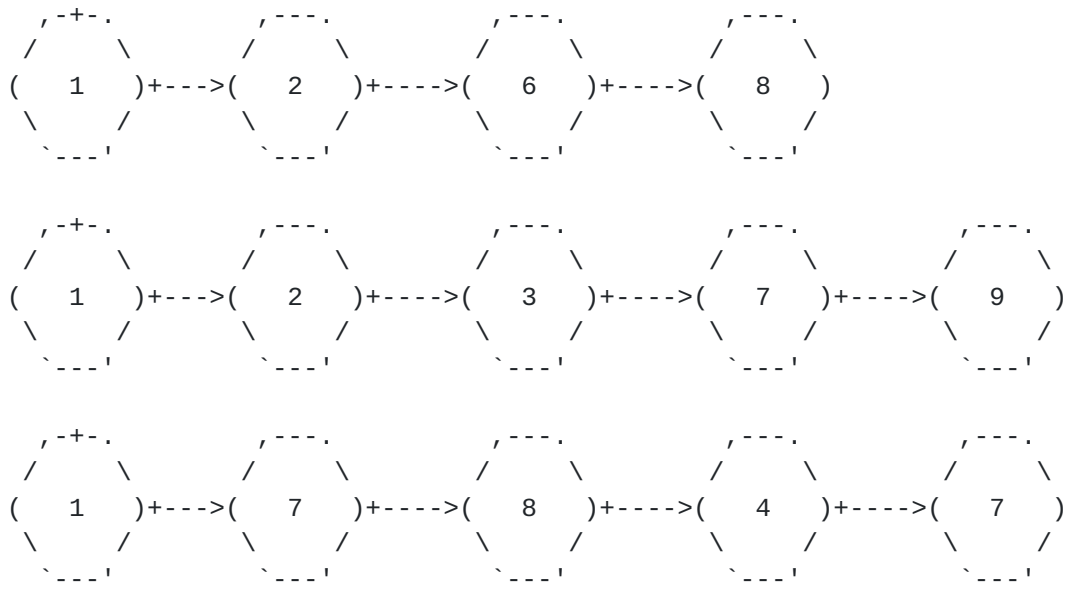

Figure 1: Service Function Chain Graphs

The concepts of classification, re-classification, and branching are covered in subsequent sections of this architecture (see Section 4.7 and Section 4.8).

2.2. Service Function Chain Symmetry

SFCs may be unidirectional or bidirectional. A unidirectional SFC requires that traffic be forwarded through the ordered SFs in one direction (SF1 -> SF2 -> SF3), whereas a bidirectional SFC requires a symmetric path (SF1 -> SF2 -> SF3 and SF3 -> SF2 -> SF1), and in which the SF instances are the same in opposite directions. A hybrid SFC has attributes of both unidirectional and bidirectional SFCs; that is to say some SFs require symmetric traffic, whereas other SFs do not process reverse traffic or are independent of the corresponding forward traffic.

SFCs may contain cycles; that is traffic may need to traverse one or more SFs within an SFC more than once. Solutions will need to ensure suitable disambiguation for such situations.

The architectural allowance that is made for SFPs that delegate choice to the network for which SFs and/or SFFs a packet will visit creates potential issues here. A solution that allows such delegation needs to also describe how the solution ensures that those service chains that require service function chain symmetry can achieve that.

Further, there are state tradeoffs in symmetry. Symmetry may be realized in several ways depending on the SFF and classifier functionality. In some cases, "mirrored" classification (i.e., from Source to Destination and from Destination to Source) policy may be deployed, whereas in others shared state between classifiers may be used to ensure that symmetric flows are correctly identified, then steered along the required SFP. At a high level, there are various common cases. In a non-exhaustive way, there can be for example:

- A single classifier (or a small number of classifiers), in which case both incoming and outgoing flows could be recognized at the same classifier, so the synchronization would be feasible by internal mechanisms internal to the classifier.

- Stateful classifiers where several classifiers may be clustered and share state.

- Fully distributed classifiers, where synchronization needs to be provided through unspecified means.

- o A classifier that learns state from the egress packets/flows that is then used to provide state for the return packets/flow.

- o Symmetry may also be provided by stateful forwarding logic in the SFF in some implementations.

This is a non-comprehensive list of common cases.

2.3. Service Function Paths

A service function path (SFP) is a mechanism used by service chaining to express the result of applying more granular policy and operational constraints to the abstract requirements of a service chain (SFC). This architecture does not mandate the degree of specificity of the SFP. Architecturally, within the same SFC-enabled domain, some SFPs may be fully specified, selecting exactly which SFF and which SF are to be visited by packets using that SFP, while other SFPs may be quite vague, deferring to the SFF the decisions about the exact sequence of steps to be used to realize the SFC. The specificity may be anywhere in between these extremes.

As an example of such an intermediate specificity, there may be two SFPs associated with a given SFC, where one SFP specifies that any order of SFF and SF may be used as long as it is within data center 1, and where the second SFP allows the same latitude, but only within data center 2.

Thus, the policies and logic of SFP selection or creation (depending upon the solution) produce what may be thought of as a constrained version of the original SFC. Since multiple policies may apply to different traffic that uses the same SFC, it also follows that there may be multiple SFPs may be associated with a single SFC.

The architecture allows for the same SF to be reachable through multiple SFFs. In these cases, some SFPs may constrain which SFF is used to reach which SF, while some SFPs may leave that decision to the SFF itself.

Further, the architecture allows for two or more SFs to be attached to the same SFF, and possibly connected via internal means allowing more effective communication. In these cases, some solutions or deployments may choose to use some form of internal inter-process or inter-VM messaging (communication behind the virtual switching element) that is optimized for such an environment. This must be coordinated with the SFF so that the service function forwarding can properly perform its job. Implementation details of such mechanisms are considered out of scope for this document, and can include a spectrum of methods: for example situations including all next-hops

explicitly, others where a list of possible next-hops is provided and the selection is local, or cases with just an identifier, where all resolution is local.

This architecture also allows the same SF to be part of multiple SFPs.

3. Architecture Principles

Service function chaining is predicated on several key architectural principles:

1. Topological independence: no changes to the underlay network forwarding topology - implicit, or explicit - are needed to deploy and invoke SFs or SFCs.

2. Plane separation: dynamic realization of SFPs is separated from packet handling operations (e.g., packet forwarding).

3. Classification: traffic that satisfies classification rules is forwarded according to a specific SFP. For example, classification can be as simple as an explicit forwarding entry that forwards all traffic from one address into the SFP. Multiple classification points are possible within an SFC (i.e., forming a service graph) thus enabling changes/updates to the SFC by SFs.

   Classification can occur at varying degrees of granularity; for example, classification can use a 5-tuple, a transport port or set of ports, part of the packet payload, it can be the result of high-level inspections, or it can come from external systems.

4. Shared Metadata: Metadata/context data can be shared amongst SFs and classifiers, between SFs, and between external systems and SFs (e.g., orchestration).

   One use of metadata is to provide and share the result of classification (that occurs within the SFC-enabled domain, or external to it) along an SFP. For example, an external repository might provide user/subscriber information to a service chain classifier. This classifier could in turn impose that information in the SFC encapsulation for delivery to the requisite SFs. The SFs could in turn utilize the user/subscriber information for local policy decisions. Metadata can also share SF output along the SFP.

5. Service definition independence: The SFC architecture does not depend on the details of SFs themselves.

6. Service function chain independence: The creation, modification, or deletion of an SFC has no impact on other SFCs. The same is true for SFPs.

7. Heterogeneous control/policy points: The architecture allows SFs to use independent mechanisms (out of scope for this document) to populate and resolve local policy and (if needed) local classification criteria.

4. Core SFC Architecture Components

At a very high level, the logical architecture of an SFC-enabled Domain comprises:

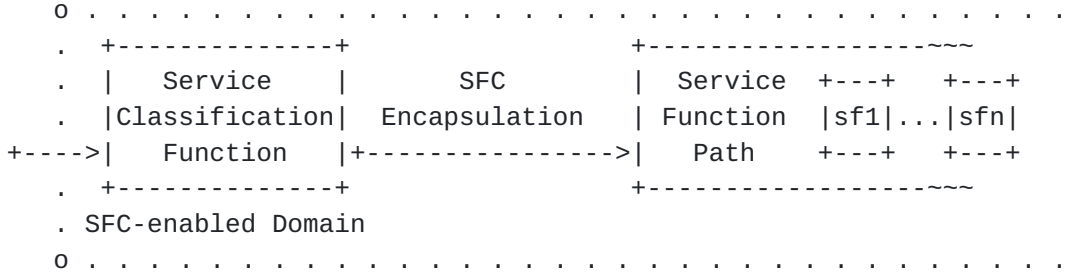

Figure 2: Service Function Chain Architecture

The following sub-sections provide details on each logical component that form the basis of the SFC architecture. A detailed overview of how each of these architectural components interact is provided in Figure 3:

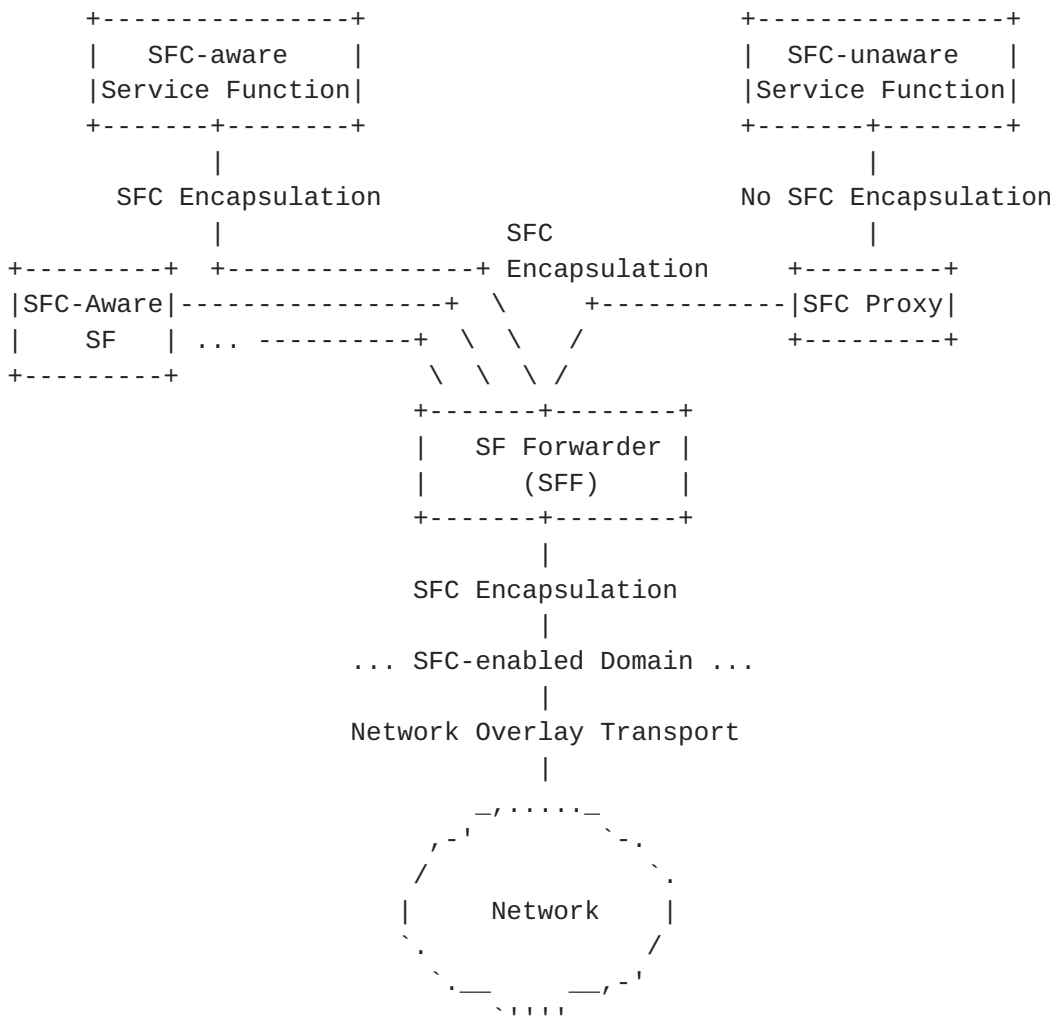

Figure 3: Service Function Chain Architecture Components

4.1. SFC Encapsulation

The SFC encapsulation enables service function path selection. It also enables the sharing of metadata/context information when such metadata exchange is required.

The SFC encapsulation carries explicit information used to identify the SFP. However, the SFC encapsulation is not a transport encapsulation itself: it is not used to forward packets within the network fabric. If packets need to flow between separate physical platforms, the SFC encapsulation therefore relies on an outer network transport. Transit forwarders -- such as router and switches -- forward SFC encapsulated packets based on the outer (non-SFC) encapsulation.

One of the key architecture principles of SFC is that the SFC encapsulation remain transport independent. As such any network transport protocol may be used to carry the SFC encapsulated traffic.

4.2. Service Function (SF)

The concept of an SF evolves; rather than being viewed as a bump in the wire, an SF becomes a resource within a specified administrative domain that is available for consumption as part of a composite service. SFs send/receive data to/from one or more SFFs. SFC-aware SFs receive this traffic with the SFC encapsulation.

While the SFC architecture defines the concept and specifies some characteristics of a new encapsulation - the SFC encapsulation - and several logical components for the construction of SFCs, existing SF implementations may not have the capabilities to act upon or fully integrate with the new SFC encapsulation. In order to provide a mechanism for such SFs to participate in the architecture, an SFC proxy function is defined (see Section 4.6). The SFC proxy acts as a gateway between the SFC encapsulation and SFC-unaware SFs. The integration of SFC-unaware service functions is discussed in more detail in the SFC proxy section.

This architecture allows an SF to be part of multiple SFPs and SFCs.

4.3. Service Function Forwarder (SFF)

The SFF is responsible for forwarding packets and/or frames received from the network to one or more SFs associated with a given SFF using information conveyed in the SFC encapsulation. Traffic from SFs eventually returns to the same SFF, which is responsible for injecting traffic back onto the network. Some SFs, such as firewalls, could also consume a packet.

The collection of SFFs and associated SFs creates a service plane overlay in which SFC-aware SFs, as well as SFC-unaware SFs reside. Within this service plane, the SFF component connects different SFs that form a service function path.

SFFs maintain the requisite SFP forwarding information. SFP forwarding information is associated with a service path identifier that is used to uniquely identify an SFP. The service forwarding state enables an SFF to identify which SFs of a given SFP should be applied, and in what order, as traffic flows through the associated SFP. While there may appear to the SFF to be only one available way to deliver the given SF, there may also be multiple choices allowed by the constraints of the SFP.

If there are multiple choices, the SFF needs to preserve the property that all packets of a given flow are handled the same way, since the SF may well be stateful. Additionally, the SFF may preserve the handling of packets based on other properties on top of a flow, such as a subscriber, session, or application instance identification.

The SFF also has the information to allow it to forward packets to the next SFF after applying local service functions. Again, while there may be only a single choice available, the architecture allows for multiple choices for the next SFF. As with SFs, the solution needs to operate such that the behavior with regard to specific flows (see the Rendered Service Path) is stable. The selection of available SFs and next SFFs may be interwoven when an SFF supports multiple distinct service functions and the same service function is available at multiple SFFs. Solutions need to be clear about what is allowed in these cases.

Even when the SFF supports and utilizes multiple choices, the decision as to whether to use flow-specific mechanisms or coarser grained means to ensure that the behavior of specific flows is stable is a matter for specific solutions and specific implementations.

The SFF component has the following primary responsibilities:

1. SFP forwarding : Traffic arrives at an SFF from the network. The SFF determines the appropriate SF the traffic should be forwarded to via information contained in the SFC encapsulation. Post-SF, the traffic is returned to the SFF, and if needed forwarded to another SF associated with that SFF. If there is another non-local (i.e., different SFF) hop in the SFP, the SFF further encapsulates the traffic in the appropriate network transport protocol and delivers it to the network for delivery to the next SFF along the path. Related to this forwarding responsibility, an SFF should be able to interact with metadata.

2. Terminating SFPs : An SFC is completely executed when traffic has traversed all required SFs in a chain. When traffic arrives at the SFF after the last SF has finished processing it, the final SFF knows from the service forwarding state that the SFC is complete. The SFF removes the SFC encapsulation and delivers the packet back to the network for forwarding.

3. Maintaining flow state: In some cases, the SFF may be stateful. It creates flows and stores flow-centric information. This state information may be used for a range of SFP-related tasks such as ensuring consistent treatment of all packets in a given flow, ensuring symmetry or for state-aware SFC Proxy functionality (see Section 4.8).

4.3.1. Transport Derived SFF

Service function forwarding, as described above, directly depends upon the use of the service path information contained in the SFC encapsulation. However, existing implementations may not be able to act on the SFC encapsulation. These platforms may opt to use existing transport information if it can be arranged to provide explicit service path information.

This results in the same architectural behavior and meaning for service function forwarding and service function paths. It is the responsibility of the control components to ensure that the transport path executed in such a case is fully aligned with the path identified by the information in the service chaining encapsulation.

4.4. SFC-Enabled Domain

Specific features may need to be enforced at the boundaries of an SFC-enabled domain, for example to avoid leaking SFC information. Using the term node to refer generically to an entity that is performing a set of functions, in this context, an SFC Boundary Node denotes a node that connects one SFC-enabled domain to a node either located in another SFC-enabled domain or in a domain that is SFC-unaware.

An SFC Boundary node can act as egress or ingress. An SFC Egress Node denotes a SFC Boundary Node that handles traffic leaving the SFC-enabled domain the Egress Node belongs to. Such a node is required to remove any information specific to the SFC Domain, typically the SFC Encapsulation. Further, from a privacy perspective, an SFC Egress Node is required to ensure that any sensitive information added as part of SFC gets removed. An SFC Ingress Node denotes an SFC Boundary Node that handles traffic entering the SFC-enabled domain. In most solutions and deployments this will need to include a classifier, and will be responsible for adding the SFC encapsulation to the packet.

4.5. Network Overlay and Network Components

Underneath the SFF there are components responsible for performing the transport (overlay) forwarding. They do not consult the SFC encapsulation or inner payload for performing this forwarding. They only consult the outer-transport encapsulation for the transport (overlay) forwarding.

4.6. SFC Proxy

In order for the SFC architecture to support SFC-unaware SFs (e.g., legacy service functions) a logical SFC proxy function may be used. This function sits between an SFF and one or more SFs to which the SFF is directing traffic (see Figure 3).

The proxy accepts packets from the SFF on behalf of the SF. It removes the SFC encapsulation, and then uses a local attachment circuit to deliver packets to SFC unaware SFs. It also receives packets back from the SF, reapplies the SFC encapsulation, and returns them to the SFF for processing along the service function path.

Thus, from the point of view of the SFF, the SFC proxy appears to be part of an SFC aware SF.

Communication details between the SFF and the SFC Proxy are the same as those between the SFF and an SFC aware SF. The details of that are not part of this architecture. The details of the communication methods over the local attachment circuit between the SFC proxy and the SFC-unaware SF are dependent upon the specific behaviors and capabilities of that SFC-unaware SF, and thus are also out of scope for this architecture.

Specifically, for traffic received from the SFF intended for the SF the proxy is representing, the SFC proxy:

- Removes the SFC encapsulation from SFC encapsulated packets.
- Identifies the required SF to be applied based on available information including that carried in the SFC encapsulation.
- Selects the appropriate outbound local attachment circuit through which the next SF for this SFP is reachable. This is derived from the identification of the SF carried in the SFC encapsulation, and may include local techniques. Examples of a local attachment circuit include, but are not limited to, VLAN, IP-in-IP, L2TPv3, GRE, VXLAN.
- Forwards the original payload via the selected local attachment circuit to the appropriate SF.

When traffic is returned from the SF:

- Applies the required SFC encapsulation. The determination of the encapsulation details may be inferred by the local attachment circuit through which the packet and/or frame was received, or via

packet classification, or other local policy. In some cases, packet ordering or modification by the SF may necessitate additional classification in order to re-apply the correct SFC encapsulation.

- o Delivers the packet with the SFC Encapsulation to the SFF, as would happen with packets returned from an SFC-aware SF.

4.7. Classification

Traffic from the network that satisfies classification criteria is directed into an SFP and forwarded to the requisite service function(s). Classification is handled by a service classification function; initial classification occurs at the ingress to the SFC domain. The granularity of the initial classification is determined by the capabilities of the classifier and the requirements of the SFC policy. For instance, classification might be relatively coarse: all packets from this port are subject to SFC policy X and directed into SFP A, or quite granular: all packets matching this 5-tuple are subject to SFC policy Y and directed into SFP B.

As a consequence of the classification decision, the appropriate SFC encapsulation is imposed on the data, and a suitable SFP is selected or created. Classification results in attaching the traffic to a specific SFP.

4.8. Re-Classification and Branching

The SFC architecture supports re-classification (or non-initial classification) as well. As packets traverse an SFP, re-classification may occur - typically performed by a classification function co-resident with a service function. Reclassification may result in the selection of a new SFP, an update of the associated metadata, or both. This is referred to as "branching".

For example, an initial classification results in the selection of SFP A: DPI_1 --> SLB_8. However, when the DPI service function is executed, attack traffic is detected at the application layer. DPI_1 re-classifies the traffic as attack and alters the service path to SFP B, to include a firewall for policy enforcement: dropping the traffic: DPI_1 --> FW_4. Subsequent to FW_4, surviving traffic would be returned to the original SFF. In this simple example, the DPI service function re-classifies the traffic based on local application layer classification capabilities (that were not available during the initial classification step).

When traffic arrives after being steered through an SFC-unaware SF, the SFC Proxy must perform re-classification of traffic to determine

the SFP. The SFC Proxy is concerned with re-attaching information for SFC-unaware SFs, and a stateful SFC Proxy simplifies such classification to a flow lookup.

4.9. Shared Metadata

Sharing metadata allows the network to provide network-derived information to the SFs, SF-to-SF information exchange and the sharing of service-derived information to the network. Some SFCs may not require metadata exchange. SFC infrastructure enables the exchange of this shared data along the SFP. The shared metadata serves several possible roles within the SFC architecture:

- Allows elements that typically operate as ships in the night to exchange information.
- Encodes information about the network and/or data for post-service forwarding.
- Creates an identifier used for policy binding by SFs.

Context information can be derived in several ways:

- External sources
- Network node classification
- Service function classification

5. Additional Architectural Concepts

There are a number of issues which solutions need to address, and which the architecture informs but does not determine. This section lays out some of those concepts.

5.1. The Role of Policy

Much of the behavior of service chains is driven by operator and per-customer policy. This architecture is structured to isolate the policy interactions from the data plane and control logic.

Specifically, it is assumed that the service chaining control plane creates the service paths. The service chaining data plane is used to deliver the classified packets along the service chains to the intended service functions.

Policy, in contrast, interacts with the system in other places. Policies and policy engines may monitor service functions to decide

if additional (or fewer) instances of services are needed. When applicable, those decisions may in turn result in interactions that direct the control logic to change the SFP placement or packet classification rules.

Similarly, operator service policy, often managed by operational or business support systems (OSS or BSS), will frequently determine what service functions are available. Operator service policies also determine which sequences of functions are valid and are to be used or made available.

The offering of service chains to customers, and the selection of which service chain a customer wishes to use, are driven by a combination of operator and customer policies using appropriate portals in conjunction with the OSS and BSS tools. These selections then drive the service chaining control logic, which in turn establishes the appropriate packet classification rules.

5.2. SFC Control Plane

This is part of the overall architecture but outside the scope of this document.

The SFC control plane is responsible for constructing SFPs, translating SFCs to forwarding paths and propagating path information to participating nodes to achieve requisite forwarding behavior to construct the service overlay. For instance, an SFC construction may be static; selecting exactly which SFFs and which SFs from those SFFs are to be used, or it may be dynamic, allowing the network to perform some or all of the choices of SFF or SF to use to deliver the selected service chain within the constraints represented by the service path.

In the SFC architecture, SFs are resources; the control plane manages and communicates their capabilities, availability and location in fashions suitable for the transport and SFC operations in use. The control plane is also responsible for the creation of the context (see below). The control plane may be distributed (using new or existing control plane protocols), or be centralized, or a combination of the two.

The SFC control plane provides the following functionality:

1. An SFC-enabled domain wide view of all available service function resources as well as the network locators through which they are reachable.

2. Uses SFC policy to construct service function chains, and associated service function paths.

3. Selection of specific SFs for a requested SFC, either statically (using specific SFs) or dynamically (using service explicit SFs at the time of delivering traffic to them).

4. Provides requisite SFC data plane information to the SFC architecture components, most notably the SFF.

5. Provide the metadata and usage information classifiers need so that they in turn can provide this metadata for appropriate packets in the data plane.

6. When needed, provide information including policy information to other SFC elements to be able to properly interpret metadata.

5.3. Resource Control

The SFC system may be responsible for managing all resources necessary for the SFC components to function. This includes network constraints used to plan and choose network path(s) between service function forwarders, network communication paths between service function forwarders and their attached service functions, characteristics of the nodes themselves such as memory, number of virtual interfaces, routes, and instantiation, configuration, and deletion of SFs.

The SFC system will also be required to reflect policy decisions about resource control, as expressed by other components in the system.

While all of these aspects are part of the overall system, they are beyond the scope of this architecture.

5.4. Infinite Loop Detection and Avoidance

This SFC architecture is predicated on topological independence from the underlying forwarding topology. Consequently, a service topology is created by Service Function Paths or by the local decisions of the Service Function Forwarders based on the constraints expressed in the SFP. Due to the overlay constraints, the packet-forwarding path may need to visit the same SFF multiple times, and in some less common cases may even need to visit the same SF more than once. The Service Chaining solution needs to permit these limited and policy-compliant loops. At the same time, the solutions must ensure that indefinite and unbounded loops cannot be formed, as such would consume unbounded resources without delivering any value.

In other words, this architecture requires the solution to prevent infinite Service Function Loops, even when Service Functions may be invoked multiple times in the same SFP.

5.5. Load Balancing Considerations

Supporting function elasticity and high-availability should not overly complicate SFC or lead to unnecessary scalability problems.

In the simplest case, where there is only a single function in the SFP (the next hop is either the destination address of the flow or the appropriate next hop to that destination), one could argue that there may be no need for SFC.

In the cases where the classifier is separate from the single function or a function at the terminal address may need sub-prefix (e.g., finer grained address information) or per-subscriber metadata, a single SFP exists (i.e., the metadata changes but the SFP does not), regardless of the number of potential terminal addresses for the flow. This is the case of the simple load balancer. See Figure 4.

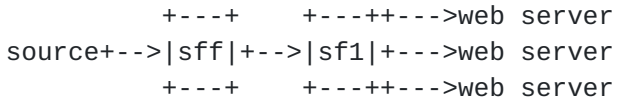

Figure 4: Simple Load Balancing

By extrapolation, in the case where intermediary functions within a chain had similar "elastic" behaviors, we do not need separate chains to account for this behavior - as long as the traffic coalesces to a common next-hop after the point of elasticity.

In Figure 5, we have a chain of five service functions between the traffic source and its destination.

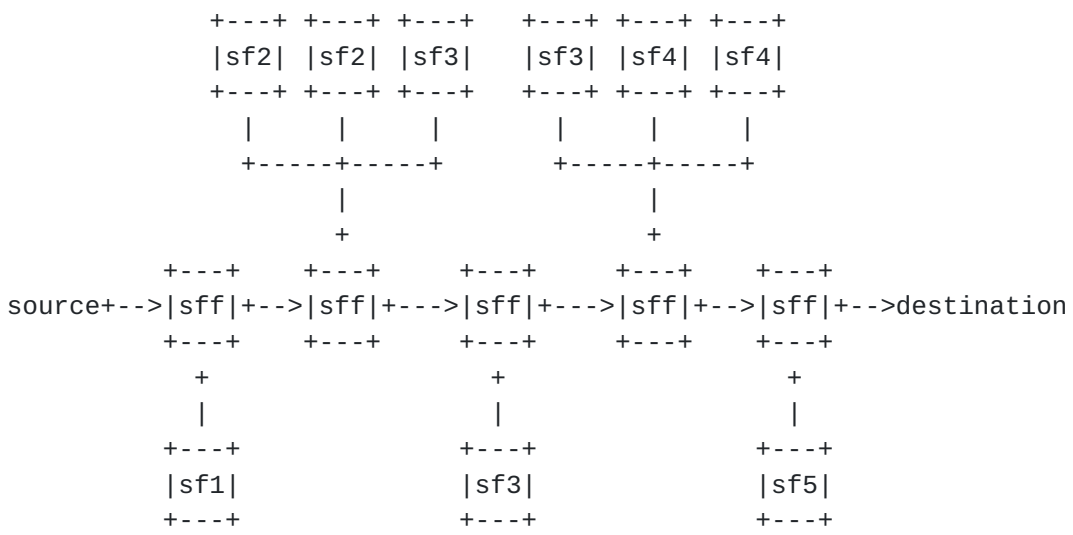

Figure 5: Load Balancing

This would be represented as one service function path: sf1->sf2->sf3->sf4->sf5. The SFF is a logical element, which may be made up of one or multiple components. In this architecture, the SFF may handle load distribution based on policy.

It can also be seen in the above that the same service function may be reachable through multiple SFFs, as discussed earlier. The selection of which SFF to use to reach SF3 may be made by the control logic in defining the SFP, or may be left to the SFFs themselves, depending upon policy, solution, and deployment constraints. In the latter case, it needs to be assured that exactly one SFF takes responsibility to steer traffic through SF3.

5.6. MTU and Fragmentation Considerations

This architecture prescribes additional information being added to packets to identify service function paths and often to represent metadata. It also envisions adding transport information to carry packets along service function paths, at least between service function forwarders. This added information increases the size of the packet to be carried by service chaining. Such additions could potentially increase the packet size beyond the MTU supported on some or all of the media used in the service chaining domain.

Such packet size increases can thus cause operational MTU problems. Requiring fragmentation and reassembly in an SFF would be a major processing increase, and might be impossible with some transports. Expecting service functions to deal with packets fragmented by the SFC function might be onerous even when such fragmentation was possible. Thus, at the very least, solutions need to pay attention

to the size cost of their approach. There may be alternative or additional means available, although any solution needs to consider the tradeoffs.

These considerations apply to any generic architecture that increases the header size. There are also more specific MTU considerations: Effects on Path MTU Discovery (PMTUD) as well as deployment considerations. Deployments within a single administrative control or even a single Data Center complex can afford more flexibility in dealing with larger packets, and deploying existing mitigations that decrease the likelihood of fragmentation or discard.

5.7. SFC OAM

Operations, Administration, and Maintenance (OAM) tools are an integral part of the architecture. These serve various purposes, including fault detection and isolation, and performance management. For example, there are many advantages of SFP liveness detection, including status reporting, support for resiliency operations and policies, and an enhanced ability to balance load.

Service Function Paths create a services topology, and OAM performs various functions within this service layer. Furthermore, SFC OAM follows the same architectural principles of SFC in general. For example, topological independence (including the ability to run OAM over various overlay technologies) and classification-based policy.

We can subdivide the SFC OAM architecture in two parts:

- In-band: OAM packets follow the same path and share fate with user packets, within the service topology. For this, they also follow the architectural principle of consistent policy identifiers, and use the same path IDs as the service chain data packets. Load balancing and SFC encapsulation with packet forwarding are particularly important here.

- Out-of-band: reporting beyond the actual data plane. An additional layer beyond the data-plane OAM allows for additional alerting and measurements.

This architecture prescribes end-to-end SFP OAM functions, which implies SFF understanding of whether an in-band packet is an OAM or user packet. However, service function validation is outside of the scope of this architecture, and application-level OAM is not what this architecture prescribes.

Some of the detailed functions performed by SFC OAM include fault detection and isolation in a Service Function Path or a Service

Function, verification that connectivity using SFPs is both effective and directing packets to the intended service functions, service path tracing, diagnostic and fault isolation, alarm reporting, performance measurement, locking and testing of service functions, validation with the control plane (see Section 5.2), and also allow for vendor-specific as well as experimental functions. SFC should leverage, and if needed extend relevant existing OAM mechanisms.

5.8. Resilience and Redundancy

As a practical operational requirement, any service chaining solution needs to be able to respond effectively, and usually very quickly, to failure conditions. These may be failures of connectivity in the network between SFFs, failures of SFFs, or failures of SFs. Per-SF state, as for example stateful-firewall state, is the responsibility of the SF, and not addressed by this architecture.

Multiple techniques are available to address this issue. Solutions can describe both what they require and what they allow to address failure. Solutions can make use of flexible specificity of service function paths, if the SFF can be given enough information in a timely fashion to do this. Solutions can also make use of MAC or IP level redundancy mechanisms such as VRRP. Also, particularly for SF failures, load balancers co-located with the SFF or as part of the service function delivery mechanism can provide such robustness.

Similarly, operational requirements imply resilience in the face of load changes. While mechanisms for managing (e.g., monitoring, instantiating, loading images, providing configuration to service function chaining control, deleting, etc.) virtual machines are out of scope for this architecture, solutions can and are aided by describing how they can make use of scaling mechanisms.

6. Security Considerations

Security considerations apply to the realization of this architecture, in particular to the documents that will define protocols. Such realization ought to provide means to protect against security and privacy attacks in the areas hereby described.

Following the categorization of [I-D.ietf-sfc-problem-statement], we can largely divide the security consdierations in three areas:

Service Overlay: Underneath the Service Function Forwarders, the components that are responsible for performing the transport forwarding consult the outer-transport encapsulation for underlay forwarding. Used transport mechanisms should satisfy the security requirements of the specific SFC deployment. These

requirements typically include varying degrees of traffic separation, protection against different attacks (e.g., spoofing, man-in-the-middle, brute-force, or insertion attacks), and can also include authenticity and integrity checking, and/or confidentiality provisions.

Classification: Specific requirements may need to be enforced at the boundaries of an SFC-enabled domain. These include, for example, to avoid leaking SFC information, and to protect its borders against various forms of attacks, including DDoS attacks. Classification is used at the ingress edge of an SFC-enabled domain. Policy for this classification is done using a plurality of methods. Whatever method is used needs to consider a range of security issues. These include appropriate authentication and authorization of classification policy, potential confidentiality issues of that policy, protection against corruption, and proper application of policy with needed segregation of application. This includes proper controls on the policies which drive the application of the SFC Encapsulation and associated metadata to packets. Similar issues need to be addressed if classification is performed within a service chaining domain, i.e., re-classification.

SFC Encapsulation: The SFC Encapsulation provides at a minimum SFP identification, and carries metadata. An operator may consider the SFC Metadata as sensitive. From a privacy perspective, a user may be concerned about the operator revealing data about (and not belonging to) the customer. Therefore, solutions should consider whether there is a risk of sensitive information slipping out of the operators control. Further, the SFC Encapsulation includes SFC OAM Functions, which need to not negatively affect the security considerations of an SFC-enabled domain.

Finally, all entities (software or hardware) interacting with the service chaining mechanisms need to provide means of security against malformed, poorly configured (deliberate or not) protocol constructs and loops. These considerations are largely the same as those in any network, particularly an overlay network.

7. Contributors and Acknowledgments

The editors would like to thank Sam Aldrin, Nicolas Bouthors, Stewart Bryant, Linda Dunbar, Alla Goldner, Ken Gray, Barry Greene, Anil Gunturu, David Harrington, Shunsuke Homma, Dave Hood, Nagendra Kumar, Hongyu Li, Andrew Malis, Guy Meador III, Kengo Naito, Thomas Narten, Ron Parker, Reinaldo Penno, Naiming Shen, Xiaohu Xu, and Lucy Yong for a thorough review and useful comments.

The initial version of this "Service Function Chaining (SFC) Architecture" document is the result of merging two previous documents, and this section lists the aggregate of authors, editors, contributors and acknowledged participants, all who provided important ideas and text that fed into this architecture.

[I-D.boucadair-sfc-framework]:

Authors:

Mohamed Boucadair
Christian Jacquenet
Ron Parker
Diego R. Lopez
Jim Guichard
Carlos Pignataro

Contributors:

Parviz Yegani
Paul Quinn
Linda Dunbar

Acknowledgements:

Many thanks to D. Abgrall, D. Minodier, Y. Le Goff, D. Cheng, R. White, and B. Chatras for their review and comments.

[I-D.quinn-sfc-arch]:

Authors:

Paul Quinn (editor)
Joel Halpern (editor)

Contributors:

Puneet Agarwal
Andre Beliveau
Kevin Glavin
Ken Gray
Jim Guichard
Surendra Kumar
Darrel Lewis
Nic Leymann
Rajeev Manur
Thomas Nadeau

Carlos Pignataro
Michael Smith
Navindra Yadav

Acknowledgements:

The authors would like to thank David Ward, Abhijit Patra, Nagaraj Bagepalli, Darrel Lewis, Ron Parker, Lucy Yong and Christian Jacquenet for their review and comments.

8. IANA Considerations

[RFC Editor: please remove this section prior to publication.]

This document has no IANA actions.

9. Informative References

[I-D.boucadair-sfc-framework]
Boucadair, M., Jacquenet, C., Parker, R., Lopez, D., Guichard, J., and C. Pignataro, "Service Function Chaining: Framework & Architecture", draft-boucadair-sfc-framework-02 (work in progress), February 2014.

[I-D.ietf-sfc-problem-statement]
Quinn, P. and T. Nadeau, "Service Function Chaining Problem Statement", draft-ietf-sfc-problem-statement-10 (work in progress), August 2014.

[I-D.quinn-sfc-arch]
Quinn, P. and J. Halpern, "Service Function Chaining (SFC) Architecture", draft-quinn-sfc-arch-05 (work in progress), May 2014.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, June 2011.

Authors' Addresses

Joel Halpern (editor)
Ericsson

Email: jmh@joelhalpern.com

Carlos Pignataro (editor)
Cisco Systems, Inc.

Email: cpignata@cisco.com