SFC
Internet-Draft
Intended status: Standards Track
Expires: August 16, 2014

M. Boucadair
C. Jacquenet
France Telecom
R. Parker
Affirmed Networks
D. Lopez
Telefonica I+D
J. Guichard
C. Pignataro
Cisco Systems, Inc.
February 12, 2014

Service Function Chaining: Framework & Architecture
draft-boucadair-sfc-framework-02

Abstract

IP networks rely more and more on the combination of advanced functions (besides the basic routing and forwarding functions) for the delivery of added value services. This document defines a reference architecture and a framework to enforce Service Function Chaining (SFC) with minimum requirements on the physical topology of the network.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

Table of Contents

1. Introduction

1.1. On the Proliferation of Service Functions

IP networks rely more and more on the combination of advanced functions (besides the basic routing and forwarding functions) for the delivery of added value services. Typical examples of such functions include firewall (e.g., [RFC6092]), DPI (Deep Packet Inspection), LI (Lawful Intercept) module, NAT44 [RFC3022], NAT64 [RFC6146], DS-Lite AFTR [RFC6333], NPTv6 [RFC6296], HOST_ID injection, HTTP Header Enrichment function, TCP tweaking and optimization function, transparent caching, charging function, load-balancer, etc.

Such advanced functions are denoted SF (Service Function) in this document.

The dynamic enforcement of a SF-derived, adequate forwarding policy for packets entering a network that supports such advanced Service Functions has become a key challenge for operators and service providers. SF-inferred differentiated forwarding is ensured by tweaking the set of Service Functions to be invoked. How to bind a flow of packets that share at least one common characteristic to a forwarding plane is policy-based, and subject to the set of SF functions that need to be solicited for the processing of this specific flow.

Service Providers need to rationalize their service delivery logics and master its underlying complexity.

The overall problem space is described in [I-D.ietf-sfc-problem-statement]. A companion document that lists a set of requirements is available at [I-D.boucadair-sfc-requirements].

1.2. Scope

This document defines a framework to enforce Service Function Chaining (SFC) with minimum requirements on the physical topology of the network. The proposed solution allows for differentiated forwarding: packets are initially classified at the entry point of an SFC-enabled network, and are then forwarded according to the ordered set of SF functions that need to be activated to process these packets in the SFC-enabled domain.

This document does not make any assumption on the deployment context. The proposed framework covers both fixed and mobile networks (e.g., to rationalize the proliferation of advanced features at the Gi Interface [RFC6459]).

Considerations related to the chaining of Service Functions that span domains owned by multiple administrative entities is out of scope. Note, a single administrative entity may manage multiple domains.

1.3. Objectives

The main objectives of the proposed framework are listed below:

- Create service-inferred forwarding planes.
- Efficiently master the chained activation of Service functions, regardless of the network topology and routing policies.
- Allow packets to be forwarded to the required Service Functions without changing the network topology or overlay transports necessary for packet delivery to/from Service Functions.
- Allow for differentiated packet forwarding by selecting the set of Service functions to be invoked.
- Allow to easily change the sequentiality of the activation of Service functions to be invoked.
- Allow to easily change the set of Service functions to be invoked.
- Ease management (including withdrawal) of Service functions and minimize any subsequent topology update.
- Automate the overall process of generating and enforcing policies to accommodate a set of network connectivity service objectives.

1.4. Assumptions

The following assumptions are made:

- Not all SFs can be characterized with a standard definition in terms of technical description, detailed specification, configuration, etc.
- There is no global nor standard list of SFs enabled in a given administrative domain. The set of SFs varies as a function of the service to be provided and according to the networking environment.
- There is no global nor standard SF chaining logic. The ordered set of SFs that need to be activated to deliver a given connectivity service is specific to each administrative entity.
- The chaining of SFs and the criteria to invoke some of them are specific to each administrative entity that operates the SF-enabled network (also called administrative domain).
- SF chaining logic and related policies should not be exposed outside a given administrative domain.
- Several SF chaining logics can be simultaneously enforced within an administrative domain to meet various business requirements.
- No assumption is made on how FIBs and RIBs of involved nodes are populated.
- How to bind the traffic to a given SF chaining is policy-based.

1.5. Rationale

Given the assumptions listed in Section 1.4, the rationale of the framework is as follows:

- The framework separates the dynamic provisioning of required SF functions from packet handling operations (e.g., forwarding decisions).
- The technical characterization of each SF is not required to design the SFC architecture and SFC operations.
- No IANA registry is required to store the list of SFs. In particular, assignment of identifiers, header fields, or any other indication of the Service Function Chain, are all strictly local in scope. An identifier assigned in one administrative domain will not indicate the same set of SFs in another administrative domain.
- No IANA registry is required to store the SF chaining candidates. The set of SFCs are local to each administrative domain, and are as such not global.
- No specific SF chaining is assumed. The description of SF chains is an information that will be processed by the nodes that participate to the delivery of a network service. The set of listed/chained SF functions is generated by each administrative entity operating the network.
- SF handling is policy-based: SF chains can be updated or deleted, new SFs can be added without any impact on existing SFs, etc. In

particular, this design is compliant with the global framework discussed in [I-D.sin-sdnrg-sdn-approach].
- For the sake of efficiency, policy enforcement is automated (but policies can be statically enforced, for example).
- To minimize fragmentation, a minimal set of information needs to be signaled (possibly in data packets).
- Advanced features (e.g., load balancing) are also described and may be configured according to policies that can be service-specific. Policy decisions are made by a Policy Decision Point [RFC2753] and the solicited enforcement points are responsible for applying these decisions, whatever the objective to achieve.
- SFs can be embedded in nodes that intervene in the transport service or supported by dedicated nodes (e.g., dedicated servers). The decision to implement one of these two models (or a combination thereof) is deployment-specific and it is orthogonal to the overall procedure.
- Multiple SFC-enabled domains can be deployed within the same administrative domain. Nodes are provisioned with the policy table of the SFC-enabled domain they belong to.
- The overall consistency of the differentiated forwarding policy is ensured by the PDP.
- The PDP can be responsible to enforce other policies than those described in the SFC Policy Tables.

2. Terminology

This document makes use of the following terms:

- SF (Service Function): refers to a function which is enabled in the network operated by an administrative entity. One or many Service Functions can be involved in the delivery of added-value services. A non-exhaustive list of Service Functions include: firewall (e.g., [RFC6092]), DPI (Deep Packet Inspection), LI (Lawful Intercept) module, NAT44 [RFC3022], NAT64 [RFC6146], DS-Lite AFTR [RFC6333], NPTv6 [RFC6296], HOST_ID injection, HTTP Header Enrichment function, TCP optimizer, load-balancer, etc. This document does not make any assumption in the OSI Layer on which the Service Function acts on; the exact definition of each Service Function is deployment-specific.

- SFC-enabled domain: denotes a network (or a region thereof) that implements SFC.

- SF Identifier: is a unique identifier that unambiguously identifies a SF within a SFC-enabled domain. SF Identifiers are assigned, configured and managed by the administrative entity that operates the SFC-enabled domain. SF identifiers can be structured as strings; other formats can be used. SF Identifiers are not

required to be globally unique nor be exposed to or used by another SF-enabled domain.

- SF Map: refers to an ordered list of SF identifiers. Each SF Map is identified with a unique identifier called SF Map Index.

- SFC Policy Table: is a table containing a list of SF Maps, SFC classification rules and Locators for all SF Nodes. A SFC Policy Table may contain a default SF Map.

- SF Locator: A SF Node identifier used to reach the said SF node. A locator is typically an IP address or a FQDN.

- Legacy Node (Node for short): refers to any node that is not a SF Node nor a SFC Boundary Node. This node can be located within a SFC-enabled domain or outside a SFC-enabled domain.

- SF Proxy Node: a Network Element along the data path, to enforce SFC functions on behalf of legacy SF nodes.

3. Functional Elements

The following functional elements are defined in this document:

- SFC Boundary Node (or 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.

- SFC Egress Node (or Egress Node): denotes a SFC Boundary Node that handles traffic which leaves the SFC-enabled domain the Egress Node belongs to.

- SFC Ingress Node (or Ingress Node): denotes a SFC Boundary Node that handles traffic which enters the SFC-enabled domain the ingress Node belongs to.

- SF Node: denotes any node within an SFC-enabled domain that embeds one or multiple SFs.

- SFC Classifier (or Classifier): an entity that classifiers packets for service chaining according to classification rules defined in a SFC Policy Table. Packets are then marked with the corresponding SF Map Index. SFC Classifier is embedded in a SFC boundary (Ingress) Node. A SFC Classifier may be considered as a Service Function, and therefore be uniquely identified by a dedicated SF Identifier.

4. SFC Provisioning

It is out of scope of this document to discuss SF-specific policy enforcement; only SFC considerations are elaborated.

4.1. Assign Service Function Identifiers

The administrative entity that operates a SFC-enabled domain maintains a local repository that lists the enabled SFs. This administrative entity assigns a unique SF identifier for each SF type.

SF identifiers are structured as character strings. SF identifiers are case-sensitive.

The main constraint on the format is that two SFs MUST be assigned with different SF identifiers if they do not provide the exact same function, or do provide the same function but are unable to differentiation packets based on policies provisioned to the SF using an appropriate mechanism.

4.2. Service Function Locator

A SF may be embedded in one or several SF Nodes. The SF locator is typically the IP address or the FQDN to reach a given SF.

The use of an IP address is RECOMMENDED to avoid any extra complexity related to the support of name resolution capabilities in SF Nodes. Resolution capabilities are supported by the PDP (Policy Decision Point). In the rest of the document, we assume a SF locator is structured as an IP address (IPv4 or IPv6).

A SF can be reached by one or more locators. When multiple SF locators are in use, the locator to be used to reach a given SF can be driven by the PDP, a SF in a SFC, result of a load-balancing heuristic, etc.

4.3. Service Function Discovery

The local repository that lists the enabled SFs within an SFC-enabled domain may be built as a direct input from the administrative entity, or they may be discovered dynamically through appropriate protocol discovery means.

Whichever method is selected by the administrative entity is a local decision and is therefore outside the scope of this document. Any Service Function Discovery solution must comply with the requirements identified in [I-D.boucadair-sfc-requirements].

4.4. Building Service Function Maps

Added-value services delivered to the end-user rely on the invocation of several SFs. For each of these services, the administrative entity that operates an SFC-enabled domain builds one or several SF Maps. Each of these maps characterizes the list of SFs to be invoked with their exact invocation order.

Each SF Map is unambiguously identified with a unique identifier called the SF Map Index. The SF Map Index MUST be described as an unsigned integer.

Distinct chains can be applied for inbound and outbound traffic. The directionality of traffic is not included as an attribute of the SF Map, but it may be implicitly described by using two SF Maps installed and maintained in the SFC Policy Table. In such case, incoming packets would be marked with Index_1 for example, while outgoing packets would be forwarded according to a distinct SF Map identified with Index_2.

An example of SF Map to handle IPv6 traffic destined to an IPv4 remote server is defined as follows:

{15, {IPv6_Firewall, HOST_ID_Inject, NAT64}}.

To handle incoming packets destined to the same IPv6 host, the following SF Map can be defined:

{10, {IPv4_Firewall, NAT64}}.

4.5. Building Service Function Chaining (SFC) Policy Tables

A PDP (Policy Decision Point, [RFC2753]) is the central entity which is responsible for maintaining SFC Policy Tables (Figure 1), and enforcing appropriate policies in SF Nodes and SFC Boundary Nodes (Figure 1). PDP-made decisions can be forwarded to the participating nodes by using a variety of protocols (e.g., NETCONF [RFC6241]).

One or multiple SFC-enabled domains may be under the responsibility of the same PDP. Delimiting the scope of each SFC-enabled domain is under the responsibility of the administrative entity that operates the SF-enabled network.

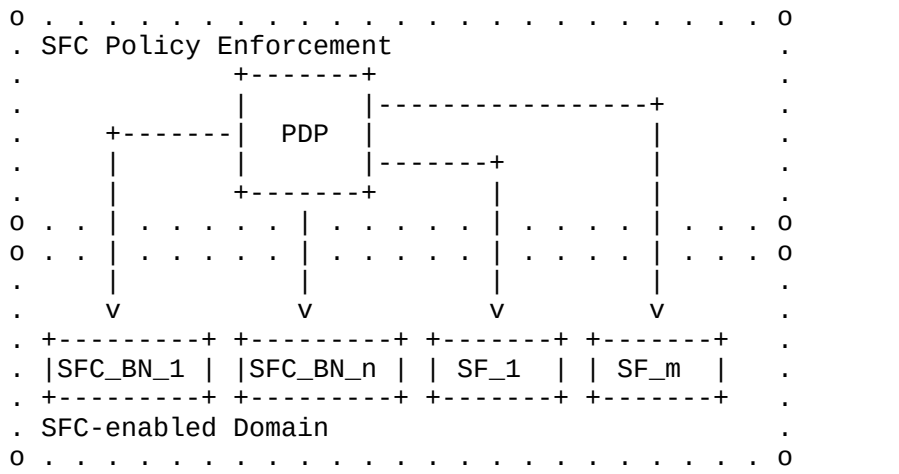


Figure 1: SFC Policy Enforcement Scheme.

The SF Node MUST be provisioned with the following information:

- Local SF Identifier(s): This information is required for an SF to identify itself within an SF Map.
- List of SF Maps: The PDP may configure the full list (default mode) or only as subset of SF Maps in which SF(s) supported by the SF Node is involved (see Section 10.7).
- List of SF Locators: The PDP may configure the full list of locators (default mode) or only the locators of next hop SFs of SF Maps in which SF(s) supported by the local SF node is involved (see Section 10.7).

  [DISCUSSION NOTE: Discuss if we maintain both forms of the SFC Policy table (full and lite) or select only one of them.]

Likewise, the SFC Boundary Node MUST be provisioned with the following information:

- List of SF Maps
- List of SF Locators
- List of SF Map Classification Rules (see Section 5.2).

In addition to the SFC Policy Table, other SF-specific policies can be installed by the PDP (e.g., configure distinct user profiles, activate specific traffic filters, configure traffic conditioners, etc.).

Policies managed by the PDP may be statically instantiated or dynamically triggered by external means (e.g., a AAA server).

In the event of any update (e.g., define a new SF Map, delete an SF Map, add a new SF Locator, update classification policy), the PDP MUST forward the updated policy configuration information in all relevant SF Nodes and SFC Boundary Nodes.

Distributing the load among several SF Nodes supporting the same SF can be driven by the PDP. Indeed, the PDP can generate multiple classification rules and SF Maps to meet some load-balancing objectives.

Load balancing may also be achieved locally by an SF Node. If the SF Node, SF Classifier, or SF Boundary Node has a table that provides the SF locator(s) of SF Nodes that provide a particular SF then it is possible to make that local load balancing decision.

The processing of packets by the nodes that belong to a SFC-enabled domain does not necessarily require any interaction with the PDP, depending on the nature of the SF supported by the nodes and the corresponding policies to be enforced. For example, traffic conditioning capabilities [RFC2475] are typical SF functions that may require additional solicitation of the PDP for the SF node to decide what to do with some out-of-profile traffic.

5. Theory Of Operation

The behavior of each node of a SFC-enabled domain is specified in the following sections. We assume that the provisioning operations discussed in Section 4 have been successful (i.e., SF functions have been adequately configured according to the SFC-specific policy to be enforced).

5.1. SFC Boundary Node

SFC Boundary Nodes act both as a SFC Ingress Node and as a SFC Egress Node for the respective directions of the traffic.

Traffic enters a SFC-enabled domain at a SFC Ingress Node (Section 5.3) and exits the domain at a SFC Egress Node (Section 5.4).

5.2. SFC Classifier

The SFC Classifier classifies packets based on (some of) the contents of the packet. Particularly, it classifies packets based on the possible combination of one or more header fields, such as source address, destination address, DS field, protocol ID, source port and destination port numbers, and any other information.

Each SF Map Classification Rule MUST be bound to one single SF Map (i.e., the classification rule must include only one SF Map Index).

5.3. SFC Ingress Node

When a packet is received through an interface of the SFC Ingress Node that connects to the outside of the SFC domain, the Ingress Node MUST:

- Inspect the received packet and check whether any existing SF Map Index is included in the packet.
  - The SFC Ingress Node SHOULD be configurable with a parameter to indicate whether received SF Map Index is to be preserved or striped. The default behavior is to strip any received SF Map Index.
  - Unless explicitly configured to trust SF Map index, The SFC Ingress Node MUST strip any existing SF Map Index if the packet is received from an SFC-enabled domain that has not explicitly been designated as "trusted".
- Check whether the received packet matches an existing classification rule (see Section 5.2).
- If no rule matches, forward the packet to the next hop according to legacy forwarding behavior (e.g., based upon the IP address conveyed in the DA field of the header).
- If a rule matches, proceed with the following operations:
  - Retrieve the locator of the first SF as indicated in the SF Map entry the rule matches. If multiple locators are available, the selection can be based on local criteria (e.g., the closest /best path).
  - Check whether the corresponding SF node is an immediate (L3) neighbor.
    - If so, update the packet with the SF Map Index of SF Map entry it matches and then forward the packet to the corresponding SF Node.
    - If not, (1) encapsulate the original packet into a new one that will be forwarded to the corresponding SF node, (2) update the encapsulated packet with the SF Map Index of SF Map entry it matches, and (3) forward the packet to the next hop to reach the first SF node.

As a result of this process, the packet will be sent to an SF Node or an Intermediate Node.

5.4. SFC Egress Node

When a packet is received through an interface that connects the SFC Egress Node to its SFC domain, the Egress Node MUST:

- Strip any existing SF Map Index.
- Forward the packet according to legacy forwarding policies.

5.5. SF Node

This section assumes the default behavior is each SF Node does not embed a Classifier as discussed in Section 10.4.

When a packet is received by a SF Node, the SF Node MUST:

- Check whether the packet conveys a SF Map Index.
- If no SF Map Index is included, forward the packet according to legacy forwarding policies.
- If the packet conveys a SF Map Index,
  - Retrieve the corresponding SF Map from the SFC Policy Table. If no entry is found in the table, forward the packet according to legacy forwarding policies.

    [DISCUSSION NOTE: Another design choice is to drop the packet and send a notification to the PDP. The justification for avoiding to drop the packet is that an SF can be part of the forwarding path of an SFC to which it does not belong to.]
  - If an entry is found in the SFC Policy Table, check whether the local SF Identifier is present in the SF Map:
    - If not, forward the packet according to legacy forwarding policies.

      [DISCUSSION NOTE: One would argue the packet should be dropped. The justification for avoiding to drop the packet is that an SF can be part of the forwarding path of an SFC to which it does not belong to + the SF node is provisioned with the full SFC Policy Table.]
    - If so, the packet is decapsulated (if needed) and then presented as an input to the local SF. In case several SFs are co-located in the same node, the packet is processed by all SFs indicated in the SF Map. Once the packet is successfully handled by local SF(s), the packet is forwarded to the next SF Node in the list or to an intermediate node (if the local SF Node is the last element in the SF Map). If the local SF node is not the last one in the SF Map, it

retrieves the next SF Node from the list, retrieve its locator for the SFC Policy Table, and forwards the packet to the next hop. If the local SF Node is the last element in the SF Map, it forwards the packet to the next hop according to legacy forwarding policies.

5.6. Intermediate Nodes

An Intermediate Node is any node that does not support any Service Function and which is located within a SFC-enabled domain.

No modification is required to intermediate nodes to handle incoming packets. In particular, routing and forwarding are achieved using legacy procedures.

6. Fragmentation Considerations

If adding the Service Chaining Header would result in a fragmented packet, the classifier should include a Service Chaining Header in each fragment. Doing so would prevent SF Nodes to dedicate resource to handle fragments.

7. Differentiated Services

When encapsulating an IP packet, the Ingress Node and each SF Node SHOULD use its Diffserv Codepoint (DSCP, [RFC2474]) to derive the DSCP (or MPLS Traffic-Class Field) of the encapsulated packet.

Generic considerations related to Differentiated Services and tunnels are further detailed in [RFC2983].

8. ECN (Explicit Congestion Notification) Considerations

When encapsulating an IP packet, the Ingress Node and each SF Node SHOULD follow [RFC6040] for ECN re-marking purposes.

9. Design Considerations

This section discusses two main protocol issues to be handled in order to deploy SFC.

A detailed design analysis is documented in [I-D.boucadair-sfc-design-analysis].

9.1. Transmit A SFC Map Index In A Packet

9.1.1. SFC Map Index

A SF Map Index is an integer that points to a SF Map.

In order to avoid all nodes of a SFC-enabled domain to be SF-aware, this specification recommends to undertake classifiers at boundary nodes while intermediate nodes forward the packets according to the SF Map Index conveyed in the packet (SF Node) or according to typical forwarding policies (any SF-unaware node).

An 8-bit field would be sufficient to accommodate deployment contexts that assume a reasonable set of SF Maps. A 16-bit (or 32-bit) field would be more flexible (e.g., to accommodate the requirement discussed in Section 10.3).

9.1.2. Where To Store SFC Map Indexes In A Packet?

SF Map Indexes can be conveyed in various locations of a packet:

- At L2 level
- Define a new IP option or a new IPv6 extension header
- Use IPv6 Flow Label
- Use MPLS Label
- Re-use an existing field (e.g., DS field)
- TCP option
- GRE Key
- Define a new shim
- Etc.

9.2. Steer Paths To Cross Specific SF Nodes

A SFC Ingress Node or a SF Node MUST be able to forward a packet that matches an existing SF Map to the relevant next hop SF Node. The locator of the next SF is retrieved from the SFC Policy Table. In case the next SF Node in the list is not an immediate (L3) neighbor, a solution to force the packet to cross that SF Node MUST be supported.

10. Deployment Considerations

10.1. Generic Requirements

The following deployment considerations should be taken into account:

- Avoid inducing severe path stretch compared to the path followed if no SF is involved.

- o Minimize path computation delays: due to the enforcement of classification rules in all participating nodes, misconception of Service function chaining, inappropriate choice of nodes elected to embed Service functions, etc., must be avoided.
- o Avoid SF invocation loops: the design of SF chainings should minimize as much as possible SF invocation loops. In any case, forwarding loops must be avoided.

10.2. Deployment Models

Below are listed some deployment model examples:

1. A full marking mechanism: Ingress nodes perform the classification and marking functions. Then, involved SF Nodes process received packets according to their marking.

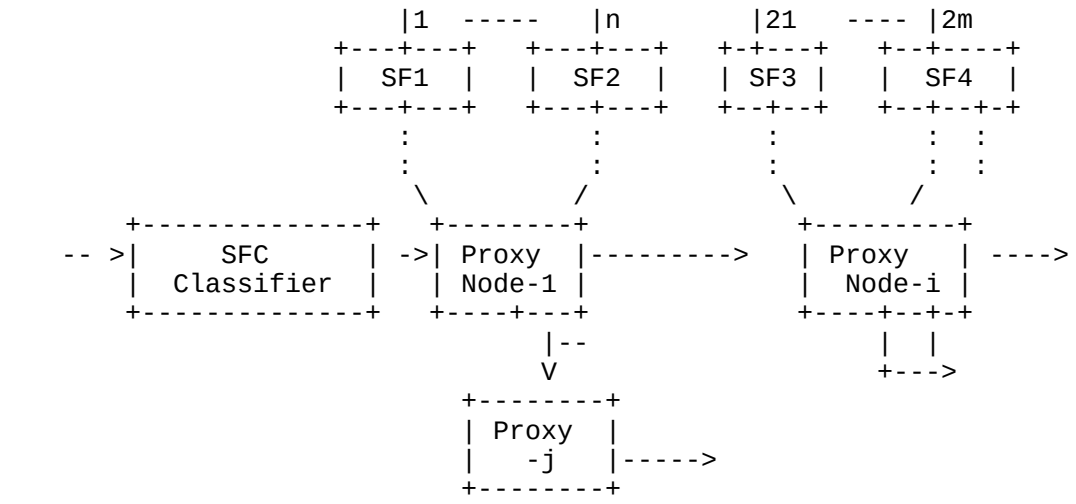
2. SF node mechanism, in which every SF Node embeds also a classifier, and the ingress node only decides the first node to forward to. Packets are forwarded at each node according to local policies. No marking is required when all SFs are co-located with a classifier. This model suffers from some limitations (see Section 10.4).

3. A router-based mechanism: All SF Nodes forward packets once processed to their default router. This default routes is responsible for deciding how the packet should be progressed at each step in the chain. One or multiple routers can be involved in the same Service Function Chain.

4. A combination thereof.

10.2.1. 1.1. Proxy Node for Legacy Service Functions

It is not uncommon to have multiple legacy service nodes located in close vicinity with a service chain proxy node, or one to two links away. The following figure depicts typical network architecture for chaining those service nodes that are not aware of service layer encapsulation.



Various deployment options can be envisaged:

1. Upgrade legacy service nodes to support required SFC functionalities.

2. Enable Proxy Service Nodes to involve these legacy nodes in instantiated SFCs.

3. Exclude legacy service nodes from a SFC domain.

It is up to the responsibility of each Service Provider to decide which option to deploy within its networks.

10.3. On Service Function Profiles (a.k.a., Contexts)

Service Functions may often enforce multiple differentiated policy sets. These policy sets may be coarsely-grained or fine-grained. An example of coarsely-grained policy sets would be an entity that performs HTTP content filtering where one policy set may be appropriate for child users whereas another is appropriate for adult users. An example of finely-grained policy sets would be PCEF (3GPP Policy Control Enforcement Function) that has a large number of differentiated QoS and charging profiles that are mapped on a per-subscriber basis.

The Service Function Chaining mechanism directly support coarsely-grained differentiated policy sets and indirectly support finely-grained differentiated policy sets.

From a Service Function Chaining perspective, each coarsely-grained policy set for a Service Function will be considered as a distinct logical instance of that Service Function. Consider the HTTP content filtering example where one physical or virtual entity provides both child and adult content filtering. The single entity is represented as two distinct logical Service Functions, each with their own Service Function Identifier from a chaining perspective. The two (logical) Service Functions may share the same IP address or may have distinct IP addresses.

Finely-grained policy sets, on the other hand, would unacceptably explode the number of distinct Service Chains that were required with an administrative domain. For this reason, Service Functions that utilize finely-grained policy sets are represented as a single Service Function that has its own internal classification mechanism in order to determine which of its differentiated policy sets to apply. Doing so avoids from increasing the size of the SFC Policy Table.

The threshold, in terms of number of policies, between choosing the coarsely-grained policy or finely-grained policy technique is left to the administrative entity managing a given domain.

[DISCUSSION NOTE: This section will be updated to reflect the conclusions of the discussions from the design analysis draft.]

10.4. SF Node is also a Classifier

If SF Nodes are also configured to behave as Classifiers, the SF Map Index is not required to be explicitly signalled in each packet. Concretely, the SFC Policy Table maintained by the SF Node includes classification rules. These classification rules are enforced to determine whether the local SF must be involved. If an incoming packet matches at least one classification rule pointing to an SF Map in which the SF Identifier is listed, the SF Node retrieves the next hop SF from the SF Map indicated in the classification rule.

The packet is then handled by the local SF, and the SF Node subsequently forwards the packet to the next hop SF. If not, the packet is forwarded to the next hop according to a typical IP forwarding policy.

Let us consider the example shown in Figure 2. The local SF Node embeds SFa. Once classification rules and the SF Maps are checked, the SF Node concludes SFa must be invoked only when a packet matches Rules 1 and 3. If a packet matches Rule 1, the next SF is SFC. If a packet matches Rule 3, the next SF is SFh.

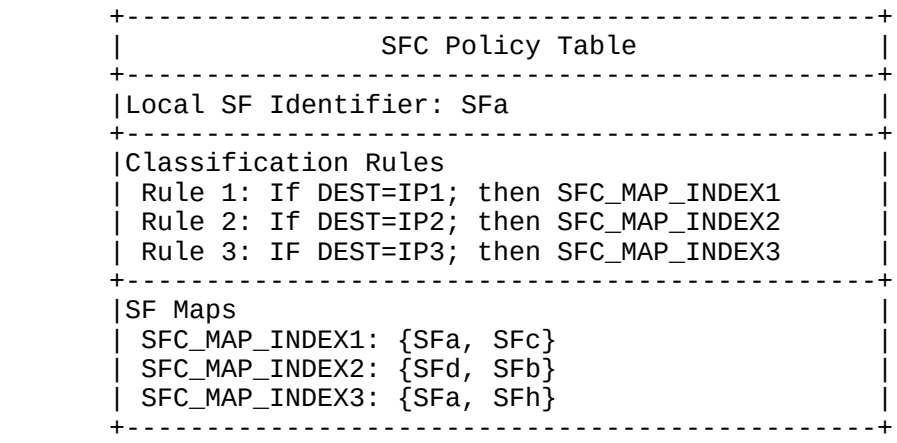


Figure 2: SFC Policy Table Example.

10.5. SFs within the Same Subnet

SF Nodes may be enabled in a SFC-enabled domain so that each of them has a direct L3 adjacency with other SF Nodes. In such configuration, no encapsulation scheme is required to exchange traffic between these nodes.

10.6. Service Function Loops

SF Nodes use the SFC Policy Table to detect whether the local SF was already applied to the received packet (i.e., detect SF Loop). The SF Node MUST invoke the local SF only if the packet is received from a SFC Boundary Node or a SF Node having an identifier listed before the local SF in the SF Map matched by the packet. SF Loop detection SHOULD be a configurable feature.

Figure 3 shows an example of a SFC Policy Table of a SF Node embedding SFa. Assume a packet received from Locb that matches Rule 2. SFa must not be invoked because SFb is listed after SFa (see the SF Map list). That packet will be forwarded without invoking SFa.

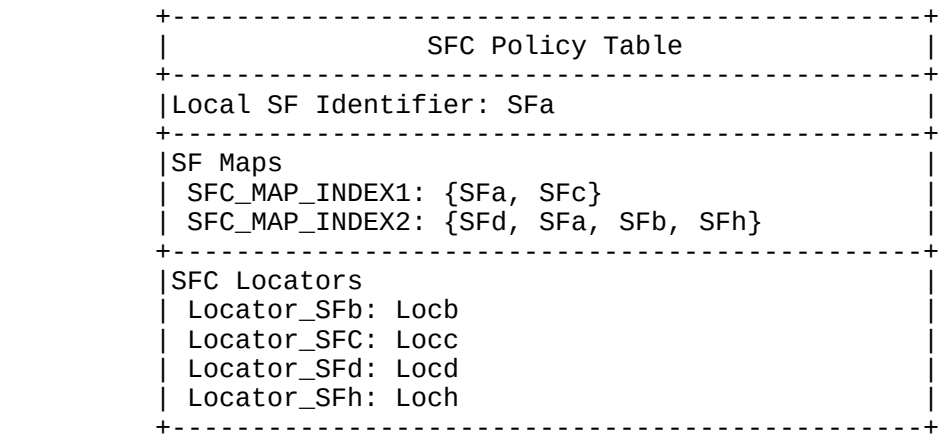


Figure 3: Dealing With SF Loops.

10.7. Lightweight SFC Policy Table

If SF loop detection is not activated in an SFC-enabled domain, the PDP may provision SF nodes with a "lightweight" SFC Policy Table. A lightweight SFC Policy Table is a subset of the full SFC Policy Table that includes:

- Only the SF Maps in which the local SF is involved.
- Only the next hop SF instead of the full SF chain.

An example of a lightweight SFC Policy Table is shown in Figure 4.

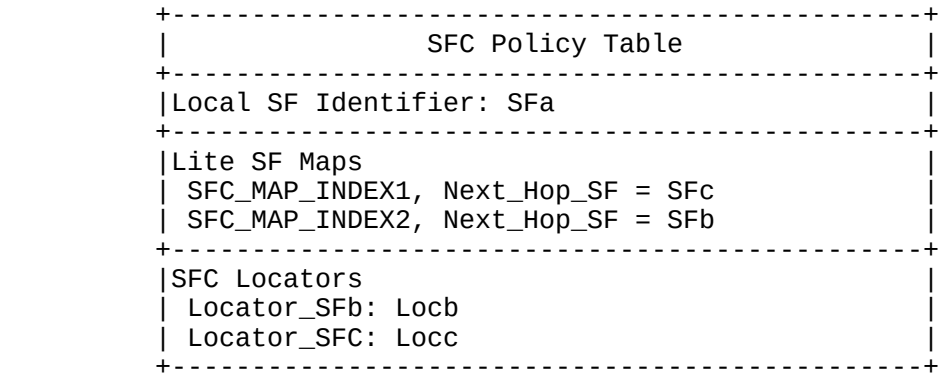


Figure 4: Lightweight SFC Policy Table.

10.8. Liveness Detection Of SFs By The PDP

The ability of the PDP to check the liveness of each SF invoked in a service chain has several advantages, including:

- Enhanced status reporting by the PDP (i.e., an operational status for any given service chain derived from liveness state of its SFs).
- Ability to support various resiliency policies (i.e., bypass SF Node, use alternate SF Node, use alternate chain, drop traffic, etc.) .
- Ability to support load balancing capabilities to solicit multiple SF instances that provide equivalent functions.

In order to determine the liveness of any particular SF Node, standard protocols such as ICMP or BFD (both single-hop [RFC5881] and multi-hop [RFC5883]) may be utilized between the PDP and the SF Nodes.

Because an SF Node can be responsive from a reachability standpoint (e.g., IP level) while the function its provides may be broken (e.g., a NAT module may be down), additional means to assess whether an SF is up and running are required. These means may be service-specific (e.g., [RFC6849], [I-D.tsou-softwire-bfd-ds-lite]).

For more sophisticated load-balancing support, protocols that allow for both liveness determination and the transfer of application-specific data, such as SNMP and NETCONF may be utilized between the PDP and the SF Nodes.

11. IANA Considerations

This document does not require any IANA actions.

12. Security Considerations

Means to protect SFC Boundary Nodes and SF Nodes against various forms of DDoS attacks MUST be supported. For example, mutual PDP and SF node authentication should be supported. Means to protect SF nodes against malformed, poorly configured (deliberately or not) SFC Policy Tables should be supported.

SFC Boundary Nodes MUST strip any existing SF Map Index when handling an incoming packet. A list of authorized SF Map Indexes are configured in the SFC elements.

NETCONF-related security considerations are discussed in [RFC6146].

Means to prevent SF loops should be supported.

Nodes involved in the same SFC-enabled domain MUST be provisioned with the same SFC Policy Table. Possible table inconsistencies may result in forwarding errors.

13. Contributors

The following individuals contributed to the document:

Parviz Yegani
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
USA
Email: pyegani@juniper.net

Paul Quinn
Cisco Systems, Inc.
USA
Email: paulq@cisco.com

Linda Dunbar
Huawei Technologies
1700 Alma Drive, Suite 500
Plano, TX 75075, USA
Phone: (469) 277 5840
Email: ldunbar@huawei.com

14. Acknowledgments

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

15. References

15.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

15.2. Informative References

[I-D.boucadair-sfc-design-analysis]
Boucadair, M., Jacquenet, C., Parker, R., and L. Dunbar, "Service Function Chaining: Design Considerations, Analysis & Recommendations", draft-boucadair-sfc-design-analysis-02 (work in progress), February 2014.

[I-D.boucadair-sfc-requirements]
Boucadair, M., Jacquenet, C., Jiang, Y., Parker, R., Pignataro, C., and K. Kengo, "Requirements for Service Function Chaining", draft-boucadair-sfc-requirements-03 (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-00 (work in progress), January 2014.

[I-D.sin-sdnrg-sdn-approach]
Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective From Within A Service Provider", draft-sin-sdnrg-sdn-approach-09 (work in progress), January 2014.

[I-D.tsou-softwire-bfd-ds-lite]
Tsou, T., Li, B., Zhou, C., Schoenwaelder, J., Penno, R., and M. Boucadair, "DS-Lite Failure Detection and Failover", draft-tsou-softwire-bfd-ds-lite-06 (work in progress), February 2014.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC2753] Yavatkar, R., Pendarakis, D., and R. Guerin, "A Framework for Policy-based Admission Control", RFC 2753, January 2000.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, October 2000.

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

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, June 2010.

[RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", RFC 5883, June 2010.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, November 2010.

[RFC6092] Woodyatt, J., "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service", RFC 6092, January 2011.

[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.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

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

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

[RFC6459] Korhonen, J., Soininen, J., Patil, B., Savolainen, T., Bajko, G., and K. Iisakkila, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)", RFC 6459, January 2012.

[RFC6849] Kaplan, H., Hedayat, K., Venna, N., Jones, P., and N. Stratton, "An Extension to the Session Description Protocol (SDP) and Real-time Transport Protocol (RTP) for Media Loopback", RFC 6849, February 2013.

Authors' Addresses

Mohamed Boucadair
France Telecom
Rennes 35000
France

EMail: mohamed.boucadair@orange.com

Christian Jacquenet
France Telecom
Rennes 35000
France

EMail: christian.jacquenet@orange.com

Ron Parker
Affirmed Networks
Acton, MA
USA

EMail: Ron_Parker@affirmednetworks.com

Diego R. Lopez
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid 28006
Spain

Phone: +34 913 129 041
EMail: diego@tid.es

Jim Guichard
Cisco Systems, Inc.
USA

EMail: jguichar@cisco.com

Carlos Pignataro
Cisco Systems, Inc.
USA

EMail: cpignata@cisco.com

SFC
Internet-Draft
Intended status: Informational
Expires: August 17, 2015

M. Boucadair
C. Jacquenet
France Telecom
Y. Jiang
Huawei Technologies Co., Ltd.
R. Parker
Affirmed Networks
K. Naito
NTT

February 13, 2015

Requirements for Service Function Chaining (SFC)
draft-boucadair-sfc-requirements-06

Abstract

This document identifies the requirements for the Service Function Chaining (SFC).

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 17, 2015.

Copyright Notice

Table of Contents

1. Introduction

This document identifies the requirements for the Service Function Chaining (SFC).

The overall problem space is described in [I-D.ietf-sfc-problem-statement].

2. Terminology

The reader should be familiar with the terms defined in [I-D.ietf-sfc-problem-statement].

The document makes use of the following terms:

- SFC-enabled domain: denotes a network (or a region thereof) that implements SFC.

- Service Function Loop: If a Service Function Chain is structured to not invoke Service Functions multiple times, a loop is the error that occurs when the same Service Function is invoked several times when handling data bound to that Service Function Chain. In other words, a loop denotes an error that occurs when a packet handled by a Service Function, forwarded onwards, and arrives once again at that Service Function while this is not allowed by the Service Function Chain it is bound to.

- Service Function Spiral: denotes a Service Function Chain in which data is handled by a Service Function, forwarded onwards, and arrives once again at that Service Function.

  * Note that some Service Functions support built-in functions to accommodate spirals; these service-specific functions may require that the data received in a spiral should differ in a way that will result in a different processing decision than the original data. This document does not make such assumption.

  * A Service Function Chain may involve one or more Service Function Spirals.

  * Unlike Service Function loop, spirals are not considered as errors.

3. Detailed Requirements List

The following set of functional requirements should be considered for the design of the Service Function Chaining solution.

3.1. Instantiating and Invoking Service Functions

SF_REQ#1: The solution MUST NOT make any assumption on whether Service Functions (SF) are deployed directly on physical hardware, as one or more Virtual Machines, or any combination thereof.

SF_REQ#2: The solution MUST NOT make any assumption on whether Service Functions each reside on a separate addressable Network Element, or as a horizontal scaling of Service Functions, or are co-resident in a single addressable Network Element, or any combination thereof.

Note: Communications between Service Functions having the same locator are considered implementation-specific. These considerations are therefore out of scope of the SFC specification effort.

SF_REQ#3: The solution MUST NOT require any IANA registry for Service Functions.

SF_REQ#4: The solution MUST allow multiple instances of a given Service Function (i.e., instances of a Service Function can be embedded in or attached to multiple Network Elements).

A. This is used for load-balancing, load-sharing, to minimize the impact of failures (e.g., by means of a hot or cold standby protection design), to accommodate planned maintenance operations, etc.

B. How these multiple devices are involved in the service delivery is deployment-specific.

SF_REQ#5: The solution MUST separate SF-specific policy provisioning-related aspects from the actual handling of packets (including forwarding decisions).

3.2. Chaining Service Functions

SFC_REQ#1: The solution MUST NOT assume any predefined order of Service Functions. In particular, the solution MUST NOT require any IANA registry to store typical Service Function Chains.

SFC_REQ#2: The identification of instantiated Service Function Chains is local to each administrative domain; it is policy-based and deployment-specific.

SFC_REQ#3: The solution MUST allow for multiple Service Chains to be simultaneously enforced within an administrative domain.

SFC_REQ#4: The solution MUST allow the same Service Function to belong to multiple Service Function Chains.

SFC_REQ#5: The solution MUST support the ability to deploy multiple SFC-enabled domains within the same administrative domain.

SFC_REQ#6: The solution MUST be able to associate the same or distinct Service Function Chains for each direction

(inbound/outbound) of the traffic pertaining to a specific service. In particular, unidirectional Service Function Chains, bi-directional Service Function Chains, or any combination thereof MUST be supported.

Note, the solution must allow to involve distinct SFC Boundary Nodes for upstream and downstream. Multiple SFC Boundary Nodes may be deployed within an administrative domain.

SFC_REQ#7: The solution MUST be able to dynamically enforce Service Function Chains. In particular, the solution MUST allow the update or the withdrawal of existing Service Function Chains, the definition of a new Service Function Chain, the addition of new Service Functions without having any impact on other existing Service Functions or other Service Function Chains.

SFC_REQ#8: The solution MUST provide means to control the SF-inferred information to be leaked outside an SFC-enabled domain. In particular, an administrative entity MUST be able to prevent the exposure of the Service Function Chaining logic and its related policies outside the administrative domain.

SFC_REQ#9: The solution MUST prevent infinite Service Function Loops.

A. Service Functions MAY be invoked multiple times in the same Service Function Chain (denoted as SF Spiral), but the solution MUST prevent infinite forwarding loops.

3.3. MTU Requirements

Packet fragmentation can be very expensive in SFC environment where fragmented packets have to be reassembled before sending to each SF on the chain. It is also worth noting that IPv6 traffic can only be fragmented by the end systems.

MTU_REQ#1: The solution SHOULD minimize fragmentation; in particular, a minimal set of SFC-specific information should be conveyed in the data packet.

MTU_REQ#2: Traffic forwarding on a SFC basis MUST be undertaken without relying on dedicated resources to treat fragments. In particular, Out of order fragments MUST be

forwarded on a per-SFC basis without relying on any state.

MTU_REQ#3: Some SFs (e.g., NAT) may require dedicated resources (e.g., resources to store fragmented packets) or they may adopt a specific behavior (e.g, limit the time interval to accept fragments). The solution MUST NOT interfere with such practices.

3.4. Independence from the Underlying Transport Infrastructure Requirements

UN_REQ#1: The solution MUST NOT make any assumption on how RIBs (Routing Information Bases) and FIBs (Forwarding Information Bases) are populated. Particularly, the solution does not make any assumption on protocols and mechanisms used to build these tables.

UN_REQ#2: The solution MUST be transport independent.

A. The Service Function Chaining should operate regardless of the network transport used by the administrative entity. In particular, the solution can be used whatever the switching technologies deployed in the underlying transport infrastructure.

B. Techniques such as MPLS are neither required nor excluded.

UN_REQ#3: The solution MUST allow for chaining logics where involved Service Functions are not within the same layer 3 subnet.

UN_REQ#4: The solution MUST NOT exclude Service Functions to be within the same IP subnet (because this is deployment-specific).

3.5. Traffic Classification Requirements

TC_REQ#1: The solution MUST NOT make any assumption on how the traffic is to be bound to a given chaining policy. In other words, classification rules are deployment-specific and policy-based. For instance, classification can rely on a subset of the information carried in a received packet such as 5-tuple classification, be subscriber-aware, be driven by traffic engineering considerations, or any combination thereof.

Because a large number (e.g., 1000s) of classification policy entries may be configured, means .Means to reduce classification look-up time such as optimizing the size of the classification table (e.g., aggregation) should be supported by the Classifier.

TC_REQ#2: The solution MUST NOT require every Service Function to be co-located with a SFC Classifier; this is a deployment-specific decision.

TC_REQ#3: The solution MAY allow traffic re-classification at the level of Service Functions (i.e., a Service Function can also be co-located with a Classifier). The configuration of classification rules in such context are the responsibility of the administrative entity that operates the SFC-enabled domain.

TC_REQ#4: The solution MUST allow Service Function Nodes to be configured (or pushed) with the detailed policies on which local Service Functions to invoke for packets associated with some Service Function Chains. The solution MUST allow those steering policies to be updated based on demand.

3.6. Data Plane Requirements

DP_REQ#1: The solution MUST be able to forward traffic between two Service Functions (involved in the same Service Function Chain) without relying upon the destination address field of the a data packet.

DP_REQ#2: The solution MUST allow for the association of a context with the data packets. In particular:

A. The solution MUST support the ability to invoke differentiated sets of policies for a Service Function (such sets of policies are called Profiles). A profile denotes a set of policies configured to a local Service Function (e.g., content-filter-child, content-filter-adult).

a. Few profiles should be assumed per Service Function to accommodate the need for scalable solutions.

b. A finer granularity of profiles may be configured directly to each Service Function; there is indeed

no need to overload the design of Service Function Chains with policies of low-level granularity.

DP_REQ#3: Service Functions may be reachable using IPv4 and/or IPv6. The administrative domain entity MUST be able to define and enforce policies with regards to the address family to be used when invoking a Service Function.

A. A Service Function Chain may be composed of IPv4 addresses, IPv6 addresses, or a mix of both IPv4 and IPv6 addresses.

B. Multiple Service Functions can be reachable using the same IP address. Each of these Service Functions is unambiguously identified with a Service Function Identifier.

DP_REQ#4:

3.7. OAM Requirements

OAM_REQ#1: The solution MUST allow for Operations, Administration, and Maintenance (OAM) features [RFC6291]. In particular, the solution MUST:

A. Support means to verify the completion of the forwarding actions until the SFC Border Node is reached (see Section 3.4.1 of [RFC5706]).

B. Support means to ensure coherent classification rules are installed in and enforced by all the Classifiers of the SFC domain.

C. Support means to correlate classification policies with observed forwarding actions.

D. Support in-band liveliness and functionality checking mechanisms for the instantiated Service Function Chains and the Service Functions that belong to these chains.

OAM_REQ#2: The solution MUST support means to detect the liveliness of Service Functions of an SFC-enabled domain. In particular, the solution MUST support means to (dynamically) detect that a Service Function instance is out of service and notify the relevant elements accordingly (PDP and Classifiers, for one).

OAM_REQ#3: Detailed diagnosis requirements are listed below:

A. The solution MUST allow to assess the status of the serviceability of a Service Function (i.e., the Service Function provides the service(s) it is configured for).

B. The solution MUST NOT rely only on IP reachability to assess whether a Service Function is up and running.

C. The solution MUST allow to diagnose the availability of a Service Function Chain (including the availability of a particular Service Function Path bound to a given Service Function Chain).

D. The solution MUST allow to retrieve the set of Service Function Chains that are enabled within a domain.

E. The solution MUST allow to retrieve the set of s Service Function Chains in which a given Service Function is involved.

F. The solution MUST allow to assess whether an SFC-enabled domain is appropriately configured (including the configured chains are matching what should be configured in that domain).

G. The solution MUST allow to assess the output of the classification rule applied on a packet presented to a Classifier of an SFC-enabled domain.

H. The solution MUST support the correlation between a Service Function Chain and the actual forwarding path followed by a packet matching that SFC.

I. The solution MUST allow to diagnose the availability of a segment of a Service Function Chain, i.e., a subset of Service Functions that belong to the said chain.

J. The solution MUST support means to notify the PDPs whenever some events occur (for example, a malfunctioning Service Function instance).

K. The solution MUST allow for local diagnostic procedures specific to each Service Function (i.e., SF built-in diagnostic procedures).

L. The solution MUST allow for customized service diagnostic.

OAM_REQ#4: Liveness status records for all Service Functions (including Service Function instances), Service Function Nodes, Service Function Chains (including the Service Function Paths bound to a given chain) MUST be maintained.

OAM_REQ#5: SFC-specific counters and statistics MUST be provided. These data include (but not limited to):

* Number of flows ever and currently assigned to a given Service Function Chain and a given Service Function Path.

* Number of flows, packets, bytes dropped due to policy.

* Number of packets and bytes in/out per Service Function Chain and per Service Function Path.

* Number of flows, packets, bytes dropped due to unknown Service Function Chain or Service Function Path (this is valid in particular for a Service Function Node).

3.8. Recovery and Load Balancing Requirements

LB_REQ#1: The solution MUST allow for load-balancing among multiple instances of the same Service Function.

A. Load-balancing may be provided by legacy technologies or protocols (e.g., make use of load-balancers)

B. Load-balancing may be part of the Service Function itself.

C. Load-balancer may be considered as a Service Function element.

D. Because of the possible complications, load balancing SHOULD NOT be driven by the SFC Classifier.

LB_REQ#2: The solution MUST separate SF-specific policy provisioning-related aspects from the actual handling of packets (including forwarding decisions).

LB_REQ#3: The solution SHOULD support protection of the failed or over-utilized Service Function instances. The protection

mechanism can rely on local decisions among the nodes that are connected to both active/standby Service Function instances.

3.9. Compatibility with Legacy Service Functions Requirements

LEG_REQ#1: The solution MUST allow for gradual deployment in legacy infrastructures, and therefore coexist with legacy technologies that cannot support SFC-specific capabilities, such as Service Function Chain interpretation and processing. The solution MUST be able to work in a domain that may be partly composed of opaque elements, i.e., elements that do not support SFC-specific capabilities.

3.10. QoS Requirements

QoS_REQ#1: The solution MUST be able to provide different SLAs (Service Level Agreements, [RFC7297]). In particular,

A. The solution MUST allow for different levels of service to be provided for different traffic streams (e.g., configure Classes of Service (CoSes)).

B. The solution MUST be able to work properly within a Diffserv domain [RFC2475].

C. The solution SHOULD support the two modes defined in [RFC2983].

QoS_REQ#2: ECN re-marking, when required, MUST be performed according to [RFC6040].

3.11. Security Requirements

SEC_REQ#1: The solution MUST provide means to prevent any information leaking that would be used as a hint to guess internal engineering practices (e.g., network topology, service infrastructure topology, hints on the enabled mechanisms to protect internal service infrastructures, etc.).

The solution MUST support means to protect the SFC domain as a whole against attacks that would lead to the discovery of Service Functions enabled in a SFC domain.
In particular, topology hiding means MUST be supported to avoid the exposure of the SFC-enabled domain

topology (including the set of the service function chains supported within the domain and the corresponding Service Functions that belong to these chains).

SEC_REQ#2: The solution MUST support means to protect the SFC-enabled domain against any kind of denial-of-service and theft of service (e.g., illegitimate access to the service) attack.

For example, a user should not be granted access to connectivity services he/she didn't subscribe to (including direct access to some SFs), at the risk of providing illegitimate access to network resources.

SEC_REQ#3: The solution MUST NOT interfere with IPsec [RFC4301] (in particular IPsec integrity checks).

4. IANA Considerations

This document does not require any action from IANA.

5. Security Considerations

Some security-related requirements to be taken into account when designing the Service Function Chaining solution are listed in Section 3.11. These requirements do not cover the provisioning interface used to enforce policies into the Classifier, Service Functions, and Service Function Nodes.

6. Contributors

The following individuals contributed text to the document:

Hongyu Li
Huawei Technologies Co., Ltd.
Bantian, Longgang district
Shenzhen 518129,
China

EMail: hongyu.lihongyu@huawei.com

Jim Guichard
Cisco Systems, Inc.
USA

EMail: jguichar@cisco.com

Paul Quinn
Cisco Systems, Inc.
USA

Email: paulq@cisco.com

Linda Dunbar
Huawei Technologies
5430 Legacy Drive, Suite #175
Plano TX
USA

EMail: linda.dunbar@huawei.com

7. Acknowledgements

Many thanks to K. Gray, N. Takaya, H. Kitada, H. Kojima, D. Dolson, B. Wright, and J. Halpern for their comments.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

[I-D.ietf-sfc-problem-statement]
Quinn, P. and T. Nadeau, "Service Function Chaining Problem Statement", draft-ietf-sfc-problem-statement-11 (work in progress), February 2015.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, October 2000.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC5706] Harrington, D., "Guidelines for Considering Operations and Management of New Protocols and Protocol Extensions", RFC 5706, November 2009.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, November 2010.

[RFC6291] Andersson, L., van Helvoort, H., Bonica, R., Romascanu, D., and S. Mansfield, "Guidelines for the Use of the "OAM" Acronym in the IETF", BCP 161, RFC 6291, June 2011.

[RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", July 2014.

Authors' Addresses

Mohamed Boucadair
France Telecom
Rennes 35000
France

EMail: mohamed.boucadair@orange.com

Christian Jacquenet
France Telecom
Rennes 35000
France

EMail: christian.jacquenet@orange.com

Yuanlong Jiang
Huawei Technologies Co., Ltd.
Bantian, Longgang district
Shenzhen 518129,
China

EMail: jiangyuanlong@huawei.com

Ron Parker
Affirmed Networks
Acton, MA
USA

EMail: Ron_Parker@affirmednetworks.com

Kengo Naito
NTT
Midori-Cho 3-9-11
Musashino-shi, Tokyo 180-8585
Japan

EMail: naito.kengo@lab.ntt.co.jp

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 23, 2015

P. Quinn, Ed.
Cisco Systems, Inc.
T. Nadeau, Ed.
Brocade
February 19, 2015

Service Function Chaining Problem Statement
draft-ietf-sfc-problem-statement-13.txt

Abstract

This document provides an overview of the issues associated with the deployment of service functions (such as firewalls, load balancers, etc.) in large-scale environments. The term service function chaining is used to describe the definition and instantiation of an ordered list of instances of such service functions, and the subsequent "steering" of traffic flows through those service functions.

The set of enabled service function chains reflect operator service offerings and is designed in conjunction with application delivery and service and network policy.

This document also identifies several key areas that the SFC working group will investigate to guide its architectural and protocol work and associated drafts.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 23, 2015.

Table of Contents

1. Introduction

The delivery of end-to-end services often require various service functions including traditional network service functions (for example firewalls and server load balancers), as well as application-specific features such as http header manipulation. Service functions may be delivered within the context of an isolated user (e.g. a tenant), or shared amongst many users/user groups.

Current service function deployment models are often tightly coupled to network topology and physical resources resulting in relatively rigid and static deployments. The static nature of such deployments greatly reduces, and in many cases, limits the ability of an operator to introduce new or modify existing services and/or service functions. Furthermore there is a cascading effect: changing one (or more) elements of a service function chain often affects other elements in the chain and/or the network elements used to construct the chain.

This issue is particular acute in elastic service environments that require relatively rapid creation, destruction or movement of physical or virtual service functions or network elements. Additionally, the transition to virtual platforms requires an agile service insertion model that supports elastic and very granular service delivery, post-facto modification and the movement of service functions and application workloads in the existing network. The service insertion model must also retain the network and service policies and the ability to easily bind service policy to granular information such as per-subscriber state.

This document outlines the problems encountered with existing service deployment models for Service Function Chaining (SFC) (often referred to simply as service chaining (in this document the terms will be used interchangeably), as well as the problems of service chain creation, deletion, modification/update, policy integration with service chains, and policy enforcement within the network infrastructure. The document highlights three key areas of WG focus for addressing the issues highlighted in this draft that will form the basis for the possible WG solutions that address the current problems.

1.1. Definition of Terms

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.

Network Overlay: A logical network built, via virtual links or packet encapsulation, over an existing network (the underlay).

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.

Service Function: 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.

A non-exhaustive list of service functions includes: firewalls, WAN and application acceleration, Deep Packet Inspection (DPI), server load balancers, NAT44 [RFC3022], NAT64 [RFC6146], HTTP Header Enrichment functions, TCP optimizer.

The generic term "L4-L7 services" is often used to describe many service functions.

Service Function Chain (SFC): A service function chain defines an ordered or partially 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 Overlay: An overlay network created for the purpose of forwarding data to required service functions.

Service Topology: The service overlay connectivity forms a service topology.

2. Problem Space

The following points describe aspects of existing service deployments that are problematic, and that the Service Function Chaining (SFC) working group aims to address.

2.1. Topological Dependencies

Network service deployments are often coupled to network topology, whether it be physical or virtualized, or a hybrid of the two. For example, use of a firewall requires that traffic flow through the firewall, which require means placing the firewall on the network path (often via creation of VLANs), or architecting the network topology to steer traffic through the firewall. Such dependency imposes constraints on service delivery, potentially inhibiting the network operator from optimally utilizing service resources, and reduces flexibility. This limits scale, capacity, and redundancy across network resources.

These topologies serve only to "insert" the service function (i.e., ensure that traffic traverses a service function); they are not required from a native packet delivery perspective. For example, firewalls often require an "in" and "out" layer-2 segment and adding a new firewall requires changing the topology (i.e., adding new layer-2 segments and/or IP subnets).

As more service functions are required - often with strict ordering - topology changes are needed in "front" and "behind" each service function resulting in complex network changes and device configuration. In such topologies, all traffic, whether a service function needs to be applied or not, often passes through the same strict order.

The topological coupling limits placement and selection of service functions: service functions are "fixed" in place by topology and therefore placement and service function selection taking into account network topology information such as load, new links, or traffic engineering is often not possible.

A common example is web servers using a server load balancer as the default gateway. When the web service responds to non-load balanced traffic (e.g., administrative or backup operations) all traffic from the server must traverse the load balancer forcing network administrators to create complex routing schemes or create additional interfaces to provide an alternate topology.

2.2. Configuration complexity

A direct consequence of topological dependencies is the complexity of the entire configuration, specifically in deploying service function chains. Simple actions such as changing the order of the service functions in a service function chain require changes to the logical and/or physical topology. However, network operators are hesitant to make changes to the network once services are installed, configured and deployed in production environments for fear of misconfiguration and consequent downtime. All of this leads to very static service delivery deployments. Furthermore, the speed at which these topological changes can be made is not rapid or dynamic enough as it often requires manual intervention, or use of slow provisioning systems.

2.3. Constrained High Availability

Since traffic reaches many service functions based on network topology, alternate, or redundant service functions must be placed in the same topology as the primary service.

An effect of topological dependency is constrained service function high availability. Worse, when modified, inadvertent non-high availability or downtime can result.

2.4. Consistent Ordering of Service Functions

Service functions are typically independent; service function_1 (SF1)...service function_n (SFn) are unrelated and there is no notion at the service layer that SF1 occurs before SF2. However, to an administrator many service functions have a strict ordering that must be in place, yet the administrator has no consistent way to impose and verify the ordering of the service functions that are used to deliver a given service. Furthermore, altering the order of a deployed chain is complex and cumbersome.

2.5. Application of Service Policy

Service functions rely on topology information such as VLANs or packet (re)classification to determine service policy selection, i.e., the service function specific action taken. Topology information is increasingly less viable due to scaling, tenancy and complexity reasons. Topology-centric information often does not convey adequate information to the service functions, forcing functions to individually perform more granular classification. In other words, the topology information is not granular enough, and its semantics often overloaded.

2.6. Transport Dependence

Service functions can and will be deployed in networks with a range of network transports, including network under and overlays, such as Ethernet, GRE, VXLAN, MPLS, etc. The coupling of service functions to topology may require service functions to support many transport encapsulations or for a transport gateway function to be present.

2.7. Elastic Service Delivery

Given that the current state of the art for adding/removing service functions largely centers around VLANs and routing changes, rapid changes to the deployed service capacity (increasing or decreasing) can be hard to realize due to the risk and complexity of VLANs and/or routing modifications.

2.8. Traffic Selection Criteria

Traffic selection is coarse, that is, all traffic on a particular segment traverses all service functions whether the traffic requires service enforcement or not. This lack of traffic selection is largely due to the topological nature of service deployment since the forwarding topology dictates how (and what) data traverses which service function(s). In some deployments, more granular traffic selection is achieved using policy routing or access control filtering. This results in operationally complex configurations and is still relatively coarse and inflexible.

2.9. Limited End-to-End Service Visibility

Troubleshooting service related issues is a complex process that involve both network-specific and service-specific expertise. This is especially the case when service function chains span multiple DCs, or across administrative boundaries. Furthermore, the physical and virtual environments (network and service), can be highly divergent in terms of topology and that topological variance adds to these challenges.

2.10. Per-Service Function (re)Classification

Classification occurs at each service function independent from previously applied service functions since there are limited mechanisms to share the detailed classification information between services. The classification functionality often differs between service functions, and service functions may not leverage the classification results from other service functions.

2.11. Symmetric Traffic Flows

Service function chains may be unidirectional or bidirectional depending on the state requirements of the service functions. In a unidirectional chain traffic is passed through a set of service functions in one forwarding direction only. Bidirectional chains require traffic to be passed through a set of service functions in both forwarding directions. Many common service functions such as DPI and firewall often require bidirectional chaining in order to ensure flow state is consistent.

Existing service deployment models provide a static approach to realizing forward and reverse service function chain association most often requiring complex configuration of each network device throughout the SFC. In other words, the same complex network configuration must be in place for both "directions" of the traffic, effectively doubling the configuration and associated testing. Further, if partial symmetry is required (i.e. only some of the services in the chain required symmetry), the network configuration complexity increases since the operator must ensure that the exceptions -- the services that do not need the symmetry flow -- are handled correctly via unique configuration to account for their requirements.

2.12. Multi-vendor Service Functions

Deploying service functions from multiple vendors often require per-vendor expertise: insertion models differ, there are limited common attributes and inter-vendor service functions do not share information, hence the need for standards to ensure interoperability.

3. Service Function Chaining

Service Function Chaining aims to address the aforementioned problems associated with service deployment. Concretely, the SFC working group will investigate solutions that address the following elements:

3.1. Service Overlay

Service function chaining utilizes a service specific overlay that creates the service topology. The service overlay provides service function connectivity, built "on top" of the existing network topology and allows operators to use whatever overlay or underlay they prefer to create a path between service functions, and to locate service functions in the network as needed.

Within the service topology, service functions can be viewed as resources for consumption and an arbitrary topology constructed to connect those resources in a required order. Adding new service functions to the topology is easily accomplished, and no underlying network changes are required.

Lastly, the service overlay can provide service specific information needed for troubleshooting service related issues.

3.2. Service Classification

Classification is used to select which traffic enters a service overlay. The granularity of the classification varies based on device capabilities, customer requirements, and services offered. Initial classification determines the service function chain required to process the traffic. Subsequent classification can be used within a given service function chain to alter the sequence of service functions applied. Symmetric classification ensures that forward and reverse chains are in place. Similarly, asymmetric -- relative to required service function -- chains can be achieved via service classification.

3.3. SFC Encapsulation

The SFC encapsulation enables the creation of a service chain in the data plane and can convey information about the chain such as chain identification and OAM status.

The SFC encapsulation also carries data plane metadata which provides the ability to exchange information between logical classification points and service functions (and vice versa) and between service functions. Metadata is not used as forwarding information to deliver packets along the service overlay.

Metadata can include the result of antecedent classification and/or information from external sources. Service functions utilize metadata, as required, for localized policy decisions.

In addition to sharing of information, the use of metadata addresses several of the issues raised in section 2, most notably by decoupling policy from the network topology, and by removing the need for per-service function classification (and re-classification) described in section 2.10.

A common approach to service metadata creates a common foundation for interoperability between service functions, regardless of vendor.

4. IANA Considerations

This document makes no request to IANA.

5. Security Considerations

Although this problem statement does not introduce any protocols, when considering service function chaining, the three main areas begin investigated (see section 3) by the WG have security aspects that warrant consideration.

Service Overlay: The service overlay will be constructed using existing transport protocols (e.g. MPLS, VXLAN) and as such is subject to the security specifics of the transport selected. If an operator requires authenticity and/or confidentiality in the service overlay, a transport (e.g. IPSec) that provides such functionally can be used.

Classification: Since classification is used to select the appropriate service overlay, and required service encapsulation details, classification policy must be both accurate and trusted. Conveying the policy to a SFC-edge device node may be done via a multitude of methods depending on an operator's existing provisioning practices and security posture.

Additionally, traffic entering the SFC domain and being classified may be encrypted thus limiting the granularity of classification. The use of pervasive encryption varies based on type of traffic, environment and level of operator control. For instance a large enterprise can mandate how encryption is used by its users, whereas a broadband provider likely does not have the ability to do so.

The use of encrypted traffic however does not obviate the need for SFC (nor the problems associated with current deployment models described herein), rather when encrypted traffic must be classified, the granularity of such classification must adapt. In such cases, service overlay selection might occur, for example, using outer (i.e. unencrypted) header information, on the presence of encryption, or via external information about the packets.

SFC Encapsulation: As described in section 3, the SFC encapsulation carries information about the SFC, and data plane metadata. Depending on environment and security posture, the SFC encapsulation might need to be authenticated and/or encrypted. The use of an appropriate overlay transport as described above can provide data plane confidentially and authenticity.

The exchange of SFC encapsulation data such as metadata must originate from trusted source(s) and, if needed, be subject to authenticity and confidentiality during the exchange to the various SFC nodes.

SFC and Multi-tenancy: If tenant isolation is required in an SFC deployment, an appropriate network transport overlay that provides adequate isolation and identification can be used. Additionally, tenancy might be used in the selection of the appropriate service chain, however, as stated, the network overlay is still required to provide transport isolation. SF deployment and how specific SFs might or might not be allocated per tenant is outside the scope of this document.

The SFC Architecture draft present a more complete review of the security implications of a complete SFC architecture.

6. Contributors

The following people are active contributors to this document and have provided review, content and concepts (listed alphabetically by surname):

Puneet Agarwal
Broadcom
Email: pagarwal@broadcom.com

Mohamed Boucadair
France Telecom
Email: mohamed.boucadair@orange.com

Abhishek Chauhan
Citrix
Email: Abhishek.Chauhan@citrix.com

Uri Elzur
Intel
Email: uri.elzur@intel.com

Kevin Glavin
Riverbed
Email: Kevin.Glavin@riverbed.com

Ken Gray
Cisco Systems
Email: kegray@cisco.com

Jim Guichard
Cisco Systems
Email:jguichar@cisco.com

Christian Jacquenet
France Telecom
Email: christian.jacquenet@orange.com

Surendra Kumar
Cisco Systems
Email: smkumar@cisco.com

Nic Leymann
Deutsche Telekom
Email: n.leymann@telekom.de

Darrel Lewis
Cisco Systems

Email: darlewis@cisco.com

Rajeev Manur
Broadcom
Email:rmanur@broadcom.com

Brad McConnell
Rackspace
Email: bmcconne@rackspace.com

Carlos Pignataro
Cisco Systems
Email: cpignata@cisco.com

Michael Smith
Cisco Systems
Email: michsmit@cisco.com

Navindra Yadav
Cisco Systems
Email: nyadav@cisco.com

7. Acknowledgments

The authors would like to thank David Ward, Rex Fernando, David McDysan, Jamal Hadi Salim, Charles Perkins, Andre Beliveau, Joel Halpern and Jim French for their reviews and comments.

Additionally, the authors would like to thank the IESG and Benjamin Kaduk for their detailed reviews and suggestions.

8. Informative References

[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.

Authors' Addresses

Paul Quinn (editor)
Cisco Systems, Inc.

Email: paulq@cisco.com

Thomas Nadeau (editor)
Brocade

Email: tnadeau@lucidvision.com

SFC Working Group
Internet Draft
Category: Informational

R. Krishnan
Brocade Communications
A. Ghanwani
Dell
J. Halpern
S. Kini
Ericsson
D. R. Lopez
Telefonica I+D

Expires: October 2014

April 21, 2014

SFC Long-lived Flow Use Cases

draft-krishnan-sfc-long-lived-flow-use-cases-02

Abstract

Long-lived flows such as file transfers, video streams are common in today's networks. In the context of service function chaining, this draft suggests use cases for dynamic bypass of certain service nodes for such flows. The benefit of this approach would be to avoid expensive Layer 7 service node processing for such flows based on dynamic decisions and improve overall performance.

This Internet-Draft will expire on April, 2014.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC 2119].

Table of Contents

1. Introduction

In the context of service function chaining, this draft suggests use cases for dynamic bypass of certain service nodes for long-lived flows such as file transfers, video streams. The benefit of this approach would be to avoid expensive Layer 4-7 service node processing for such flows and improve overall performance. The focus would be only on long-lived flows which are observable and controllable from a control plane perspective; attempting dynamic bypass for short-lived flows would cause excessive control plane chattiness without any significant performance benefit.

For long-lived flows, in order to dynamically bypass certain service nodes in the service function chain, the key is to make sure that the Layer 7 flow can be identified using Layer 2/3/4 fields in the packet. Examples of such flows are file transfers (FTP) and video streams (typically use HTTP) which can be mapped to a unique IP 5 tuple (IP source address, IP destination address, IP protocol, TCP/UDP source port, TCP/UDP destination port). We note that it may not always be possible to identify a Layer 7 flow based on L2/L3/L4 fields in the packet header. An example of this could be file transfers under persistent HTTP sessions where multiple files would be transferred using the same fields in the packet headers.

The definition of long-lived flow in this context can re-use the definition in [I2RS-large-flow] and [OPSAWG-large-flow], where flows are categorized into 4 types - short-lived small flows, short-lived large flows, long-lived small flows and long-lived large flows. In this draft we are concerned with the last 2 types -- long-lived small flows and long-lived large flows - and we refer to these as long-lived flows. This identification of long-lived flows is based on L2/L3/L4 fields in the packet header that is consistent with that the definition of a flow in IPFIX [RFC 7011].

The criteria used by the service node for identifying a long-lived Layer 4-7 flow can use similar criteria, with appropriate modification to account for long-lived small flows, as the techniques described in [OPSAWG-large-flow] for large flow identification. The mechanics of dynamic bypass are quite different for different service functions and are described in the following sections.

For the mechanisms in this draft, our focus is on the following SFC components:

- An SFC Control Plane Application which is responsible for implementing the control plane functionality and programming the data plane for SFC.

- An SFC edge, which is a switch/router responsible for adding/removing the service chain header to the packets.

1.1. Acronyms

COTS: Commercial Off-the-shelf

DOS: Denial of Service

DDoS: Distributed Denial of Service

ECMP: Equal Cost Multi-path

GRE: Generic Routing Encapsulation

LAG: Link Aggregation Group

LSR: Label Switch Router

MPLS: Multiprotocol Label Switching

NVGRE: Network Virtualization using Generic Routing Encapsulation

PBR: Policy Based Routing

QoS: Quality of Service

STT: Stateless Transport Tunneling

TCAM: Ternary Content Addressable Memory

VXLAN: Virtual Extensible LAN

2. Transparent Firewall Use Case

A transparent firewall determines that a long-lived flow (e.g. video stream, file transfer) has no security issues. This long-lived flow is made to dynamically bypass the firewall service function but continue to execute the other service functions in the chain (e.g. NAPT). The key benefit is overall performance improvement. The event sequence for this use case is detailed below. Another point to note is that the firewall is transparent and does not perform packet modification.

2.1. Event Sequence

1. The firewall examines packets of a flow and deems that it is benign. This can be based on many factors such as

    a. The packets are encrypted packets which cannot be decrypted and examined further

    b. The packets are from a trusted source

    c. The packets are from a trusted application

2. The firewall determines that the flow can be identified using a Layer 2/3/4 rule in the fast path. The firewall moves the flow from the internal slow path (which inspects every packet) to the fast path (which does only switching and skips the detailed inspection of every packet).

3. Based on the above criteria and also having identified the flow as a long-lived flow, the firewall determines that the flow is a benign one and does not need to be processed by the firewall any more.

4. The firewall signals this information to the SFC Control Plane Application.

5. The SFC Control Plane Application assigns the flow to a different service function chain that excludes the firewall.

6. The flow continues to be monitored by the SFC edge switch/router for activity.

7. Once the flow is detected as having become inactive, the flow is aged out by the SFC edge switch/router.

8. The SFC edge switch/router signals a flow age event to the SFC Control Plane Application.

9. The SFC Control Plane Application removes the dynamic service chain association created for the flow.

3. Long-tail content CDN Use Case

Most popular content is of interest to a number of users; typical examples are newly released movies, latest television episodes, etc. Such content is very amenable to caching. A single copy of the

content is delivered to the cache; the content is delivered to multiple users from the cache.

Long-tail personalized content is of interest to only a few users; typical examples are documentaries, older movies etc. Long-tail personalized content is typically not shared by many users and is not amenable to caching [CDNI-long-tail]. Caching of such content, could cause excessive thrashing of the cache.

The idea is to improve performance by identifying such long-tail content and bypassing the CDN cache in the service chain for such content. This would be dynamic in nature, since content which is not so popular can become popular and vice versa. The focus will be on long-lived content such as movies, catch up episodes which generate long-lived flows. The key benefit is overall performance improvement. The event sequence for this use case is detailed below.

For the purpose of this draft, our focus is on the following components in the CDN:

. CDN Monitoring System: The CDN Monitoring System monitors various aspects of the content such as

  o Dynamic Content Usage: Number of users simultaneously viewing the same content.

  o Content Life: If the content is long-lived or short-lived. Examples of long-lived content are movies, catch up episodes, etc., while examples of short-lived content are video clips, advertisements, etc.

. CDN Cache: This is the node in the network where the content is cached.

For a general overview of CDNs, see [CDN-overview].

3.1. Event Sequence

1. The CDN Monitoring System monitors the numbers of users and type of content being accessed. By default, we assume the CDN Cache is bypassed.

2. If the number of users viewing the same content exceeds a pre-programmed threshold and the content is long-lived, the CDN Monitoring System instructs the SFC Control Plane Application to dynamically add a CDN Cache to the service chain for that content.

This is done by installing a rule for that flow in the SFC edge switch/router.

3. If the number of users viewing the same content falls below a pre-programmed threshold and the content is long-lived, the monitoring server instructs the SFC Control Plane Application to a dynamically remove a CDN Cache from the service chain for the content. This is done by removing the rule for that flow from the SFC edge switch/router.

4. IPsec Management in Mobile Environments

Existing security procedures for flow protection in LTE are based on the use of IPsec tunnels between the radio base stations (eNodeBs) and some central node in the core, where a security gateway (SecGW) is deployed. The eNodeB device located on the cell site initiates the IPSec tunnel through the backhaul network to the SecGW, where the tunnel is terminated and the traffic is forwarded towards its final destination. IPsec ESP is the method that LTE standards use for achieving the required levels of security [TS33.401].

To avoid traffic bottlenecks and in order to guarantee a high level of service availability, a recommended practice is the concurrent use of several SecGW devices. The one that is to be used for a given traffic flow may be determined by several criteria such as the origin of the traffic (user traffic vs network control), flows with well-known characteristics, e.g. security properties (HTTPS, secure VPNs), etc. In this way, more critical traffic can be prioritized, and different levels of security can be applied depending of payload characteristics.

Such an optimization could be applied as well to long-lived flows in a dynamic way, relaxing security procedures for non-sensitive ones, e.g. it may not be necessary to secure a well-known video stream that is openly available, applying differentiated policies to avoid congestion, or even hardening the security procedures according to the user’s data profile.

4.1. Event Sequence

1. A monitoring element such as a DPI appliance analyzes the new flows arriving at the default SecGW device used by a given eNodeB device according to criteria such as:

   . Security payload protection;

   . Application and transport protocol(s) in use;

. Relevant parameters in those protocols (URL, content-transfer declarations, etc.).

2. If the monitoring element identifies a long-lived flow that matches its differentiating criteria, it signals the flow to the SFC Control Plane Application.

3. The SFC Control Plane Application assigns the flow to a different service function chain that makes the eNodeB device use a different SecGW device.

4. Once the flow is becomes inactive, it is aged out by the eNodeB device and signaled as such to the SFC Control Plane Application.

5. The SFC Control Plane Application removes the dynamic service chain association that was created for the flow.

5. Operational Considerations

Any modification to the SFC path (due to insertion or removal of a service function) could result in temporary mis-ordering in the delivery of packets.

6. IANA Considerations

None.

7. Security Considerations

This draft specifies a use case for SFC and does not introduce any new security requirements beyond those already under consideration for SFC.

8. Acknowledgements

9. References

9.1. Normative References

9.2. Informative References

[OPSAWG-large-flow] Krishnan, R. et al., "Mechanisms for Optimal LAG/ECMP Component Link Utilization in Networks," February 2014.

[I2RS-large-flow] Krishnan, R. et al., "I2RS Large Flow Use Case," November 2013.

[CDNI-long-tail] Krishnan, R. et al., "Best practices and Requirements for delivering Long Tail personalized content delivery over CDN Interconnections," work in progress, May 2013.

[CDN-overview] Dilley, J. et al., "Globally distributed content delivery," IEEE Internet Computing, September-October 2002.

[RFC 7011] Claise, B., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," September 2013.

[RFC 2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.

[TS33.401] 3GPP Technical Specification 33.401, "Security Architecture," December 2013.

Authors' Addresses

Ram Krishnan
Brocade Communications
ramk@brocade.com

Anoop Ghanwani
Dell
anoop@alumni.duke.edu

Joel Halpern
Ericsson
joel.halpern@ericsson.com

Sriganesh Kini
Ericsson
Sriganesh.kini@ericsson.com

Diego Lopez
Telefonica I+D
Don Ramon de la Cruz, 82 Street
Madrid, 28006, Spain
+34 913 129 041
diego@tid.es

Service Function Chaining
Internet-Draft
Intended status: Informational
Expires: November 4, 2014

S. Kumar
C. Obediente
Cisco Systems, Inc.
M. Tufail
Citi
S. Majee
F5 Networks
C. Captari
Telstra Corporation

May 3, 2014

Service Function Chaining Use Cases In Data Centers
draft-kumar-sfc-dc-use-cases-02

Abstract

Data center operators deploy a variety of layer 4 through layer 7 service functions in both physical and virtual form factors. Most traffic originating, transiting, or terminating in the data center is subject to treatment by multiple service functions.

This document describes use cases that demonstrate the applicability of Service Function Chaining (SFC) within a data center environment and provides SFC requirements for data center centric use cases, with primary focus on Enterprise data centers.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 4, 2014.

Table of Contents

1. Introduction

Data centers -- enterprise, cloud or service provider -- deploy service nodes at various points in the network topology. These nodes provide a range of service functions and the set of service functions hosted at a given service node may overlap with service functions hosted at other service nodes.

Often, data center topologies follow a hierarchical design with core, aggregation, access and virtual access layers of network devices. In such topologies service nodes are deployed either in the aggregation or access layers. More recent data center designs utilize a folded CLOS topology to improve scale, performance and resilience while ensuring deterministic hop count between end points. In such spine-leaf topologies, service nodes are often deployed at compute or virtual access layers as well as physical access layers.

The primary purpose of deploying service functions at different points in the network is to apply service functions to different types of traffic:

a. Traffic originating at physical or virtual workloads in the data center and destined to physical or virtual workloads in the data center; for example three-tiered deployment of applications: web, application, and database tiers, with traffic flowing between the adjacent tiers.

b. Traffic originating at a location remote to the data center and destined to physical or virtual workloads in the data center; for example traffic originating at a branch or regional office, destined to one of the primary data centers in an Enterprise, or traffic originating at one of the tenants of a Service Provider destined to that tenants applications in the Service Provider data center. Yet another variant of this type of traffic includes third party vendors and partners of the data center operator remotely accessing their applications in the data center over secure connections.

c. Traffic that is originating at a location remote to the data center and destined to a location remote to the data center but transiting through the data center; for example traffic originating at a mobile device destined to servers in the Internet routed through the data center to in order to service it.

Servicing of traffic involves directing the traffic through a series of service functions that may be located at different places in the network or within a single device connected to the network or any

combination in between. Delivery of multiple service functions in a sequence, in a datacenter, thus creates many requirements on the overall service delivery architecture. Such architectures may be termed service function chaining architectures while the list of service functions applied to the traffic is a Service Function Chain (SFC).

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Definition Of Terms

Additional terms are defined in [I-D.ietf-sfc-problem-statement], which the reader may find helpful.

End Point (EP): A device or an application that is the ultimate origination or destination entity of specific traffic. Mobile devices, desktop or server computers and applications running on them are some examples.

Workload (WL): A physical or virtual machine performing a dedicated task that consumes compute, storage, network and other resources. This may include web servers, database servers, storage servers and a variety of application servers.

Service Function (SF): A function that is responsible for specific treatment of received packets. A Service Function can act at the network layer or other OSI layers. A Service Function can be a virtual instance or be embedded in a physical network element. One of multiple Service Functions can be embedded in the same network element. Multiple instances of the Service Function can be enabled in the same administrative domain. A non-exhaustive list of Service Functions includes: firewalls, WAN and application acceleration, Deep Packet Inspection (DPI), server load balancers, NAT44 [RFC3022], NAT64 [RFC6146], HOST_ID injection, HTTP Header Enrichment functions, TCP optimizer, etc.

Service Node (SN): A virtual or physical device that hosts one or more service functions, which can be accessed via the network location associated with it.

Deep Packet Inspection (DPI): service function that performs stateful inspection of traffic, identification of applications and policy enforcement, among others.

Intrusion Detection and/or Prevention System (IDS/IPS): Is a DPI SN with additional capabilities to recognize malware and other threats and take corrective action.

Edge Firewall (EdgeFW): SN hosting service functions such as VPN, DHCP, NAT, IP-Audit, Protocol Inspection, DPI etc. with policies primarily focusing on threats external to the data center.

Segment Firewall (SegFW): SN hosting a subset of the functions in the EdgeFW not including VPN and is deployed to protect traffic crossing segments, such as VLANs.

Application Firewall (AppFW): service function that isolates traffic within a segment or protects from application specific threats. This falls into the same class as DPI but deployed much closer to the applications. It is an intra-segment firewall.

Application Delivery Controller (ADC): service function that distributes traffic across a pool of servers (applications) for efficient resource utilization, application scaling as well as to provide high availability among others.

Web Optimization Control (WOC): SN hosting service functions to optimize the use of WAN link bandwidth, improve effective user throughput and latencies leading to overall improved user experience. WOC includes various optimizers such as compression, de-duplication, congestion control, application specific optimizers, etc. WOC requires peers at either end of the WAN link to perform optimizations. The scope of this document is limited to the DC side of the WAN link.

Monitoring (MON): SN hosting service functions to obtain operational visibility into the network to characterize network and application performance, troubleshoot performance issues, optimize resource utilization, etc.

Note: The above definitions are generalized. Actual implementations may vary in scope and in a lot of cases the actual service functions hosted on SNs overlap. For instance, DPI function is not only implemented as a standalone service function but is also implemented in EdgeFWs. Likewise EdgeFw functions, such as VPN, are implemented in routers. The terms used are representative of common usage and not absolute deployment.

3. Use Cases

The following sections highlight some of the most common data center use case scenarios and are in no way exhaustive.

3.1. Traffic Types

IT assets in an enterprise are consolidated into a few data centers located centrally. This consolidation stems from regulatory compliance regarding security, control on the enterprise assets, operational cost savings, disaster recovery strategies, etc. The data center resources are accessible from any geographic location whether inside or outside the enterprise network. Further, enterprise data centers may be organized along lines of businesses, with each business treated as a tenant, thereby supporting multi-tenancy.

Service provider data centers have similar requirements as the enterprise. Data centers may be distributed regionally and globally to support the needs of their tenants. Multi-tenancy underlines every consideration in such data centers: resources and assets are organized & managed on tenant boundaries, policies are organized along tenant boundaries, traffic is segregated and policies enforced on tenant boundaries, etc. This is true in all "as a service" models: IaaS, PaaS and SaaS.

This leads to two primary types of traffic: North-South and East-West, both with different service requirements.

3.2. North-South Traffic

North-South traffic originates from outside the data center and is typically associated with users - onsite, remote and VPN - conducting their jobs. The traffic may also be associated with consumers accessing news, email, social media and other websites. This traffic is typically destined to applications or resources hosted in the data centers. Increasing adoption of BYOD and social networking applications requires traffic be analyzed, application and users be identified, transactions be authorized, and at the same time security threats be mitigated or eliminated. To this end, various service functions, as illustrated in Figure 1, are deployed in different SNs and in many instances of those SNs, at various topological locations in the network. The SNs are selected based on the policy required for the specific use case.

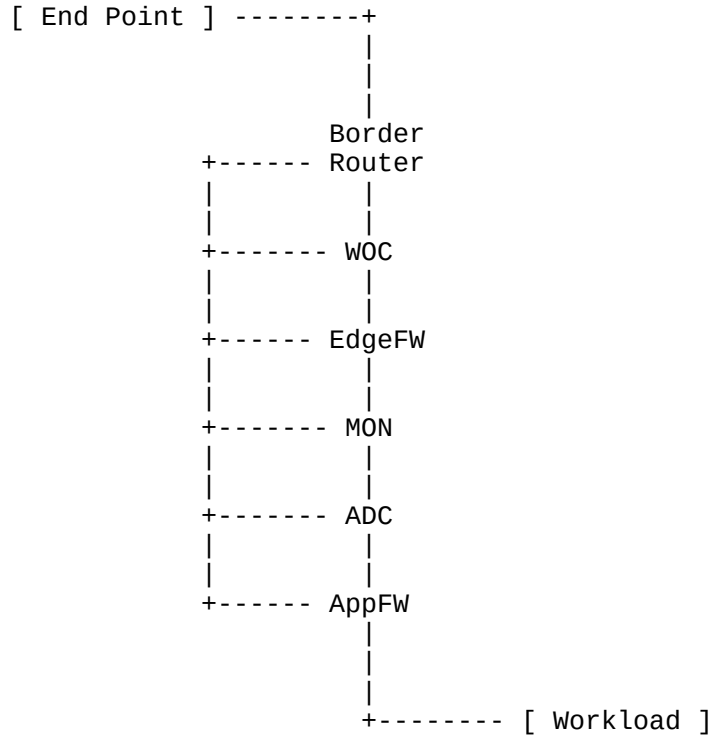


Figure 1: Service functions applied to North-South traffic

Figure 1 shows the ordered list of SNs, from top to bottom, representing the flow of traffic from End Point to Workload and vice versa. Traffic does not always strictly flow through all the SNs in that order. Traffic flows through various permutations, of the subsets, of the SNs. The connections from each of the service nodes to every other service node (as depicted by the vertical line to the left) represents the network topology required to achieve such traffic flows. Each permutation represents a service function chain.

Certain ordering of the SNs naturally exists due to the nature of the functions applied. For instance, WOC is not effective on VPN traffic - requires VPN termination prior to WOC. Likewise EdgeFW may not be effective on WOC traffic. Vendor implementations of SNs enable choices for various deployments and ordering. For instance EdgeFW detects the presence of WOC through TCP options or explicit configuration and hence WOC may even be deployed on the traffic that has passed through the EdgeFW. Constructing service function chains in the underlay network thus requires complex topology.

3.2.1. Sample north-south service function chains

SFC-1. EdgeFW

SFC-2. EdgeFW : ADC

SFC-3. EdgeFW : ADC : AppFW

SFC-4. WOC : EdgeFW : ADC : AppFW

SFC-5. WOC : EdgeFW : MON : ADC : AppFW

3.2.2. Sample north-south SFC description

Sample service chains numbered SFC-1 through SFC-5 capture the essence of services required on the north-south traffic.

SFC-1: This represents the simplest of use cases where a remote or mobile worker accesses a specific data center server. Traffic comes into the data center on VPN and is terminated on the EdgeFW. EdgeFW subjects the traffic to its policies, which may in turn select other service functions such as DPI, IPS/IDS, hosted on the EdgeFW. As an alternative deployment, some of these service functions may be hosted outside the EdgeFW and reachable via VLAN stitching. EdgeFW policy permitting, traffic is allowed to its destination.

SFC-2: This is an extension of SFC-1. Traffic instead of destined to a specific server is destined to a data center application that is front-ended by an ADC. The EdgeFW performs its function as before and the traffic is allowed, policy permitting. This traffic reaches its virtual destination, the ADC. ADC, based on local policy, which includes among other things predictors to select the real destination, determines the appropriate application instance. ADCs are stateful and ensure the return traffic pass through them by performing source NAT. Since many applications require the original source address, ADC preserves the original address in extension headers of the HTTP protocol. Traffic is then forwarded on to the ultimate destination - the real application workload.

SFC-3: This extends SFC-2. The segment where the application server resides may be shared with other applications and resources. To segregate these applications and resources further fine grain policies may be required and are enforced via a security appliance such as the AppFW. As a consequence AppFW first services the traffic from the load balancer before it is forwarded to its ultimate destination, the application server.

SFC-4: This is a variant of SFC-3 with WOC being part of the chain. This represents the use case where users at a branch office access the data center resources. The WOC SNs located at either end of the WAN optimize the traffic first. The WOC located in the datacenter requires a mechanism to steer traffic to it while not deployed inline with the traffic. This is achieved either with PBR or VLAN stitching. WOC treated traffic is subject to firewall policies, which may lead to the application of SFs such as protocol inspection, DPI, IDS/IPS and then forwarded to its virtual destination, the ADC.

SFC-5: This is similar to SFC-4. An additional service - MON, is used to collect and analyze traffic entering and leaving the data center. This monitoring and analysis of traffic helps maintain performance levels of the infrastructure to achieve service level agreements, particularly in SP data centers.

3.3. East-West Traffic

This is the predominant traffic in data centers today. Server virtualization has led to the new paradigm where virtual machines can migrate from one server to another across the datacenter. This explosion in east-west traffic is leading to newer data center network fabric architectures that provide consistent latencies from one point in the fabric to another.

The key difference with east-west from the north-south traffic is in the kind of threats and the security needs thereof. Unlike north-south traffic where security threats may come from outside the data center, any threat to this traffic comes from within the data center.

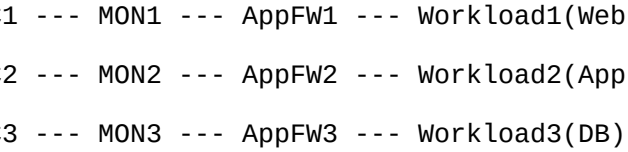


Figure 2: Service functions applied to East-West traffic

Service functions applied on the east-west traffic is captured in a generalized fashion in Figure 2. ADCs, although shown as isolated SNs in each of the tiers, is often consolidated into a smaller number of ADC SNs shared among the different tiers. Virtual IPs in such ADCs represent the individual ADC instances. Flows are terminated at

the VIPs and re-initiated towards the load balanced workloads.

As an example, HTTP GET request arriving at ADC1 is load balanced on to a webserver pool represented as Workload1. In order to respond to the GET request, Workload1 generates traffic to an application server in a pool represented as Workload2 through ADC2, which load balances the webserver initiated traffic. Likewise, the application server, as part of processing the webserver's request generates traffic to a DB server pool represented as Workload3 through ADC3, which load balances the application server initiated traffic. The traffic arriving at different ADCs, in this example, can be arriving at different VIPs, instead, each corresponding to its tier but belonging to the same ADC. In this sense, traffic flow across the tiers is VIP centric as opposed to device instance.

Traffic traversing between the ADC and the selected server in each tier, is subject to monitoring and one or more application firewalls specializing in different kinds and aspects of threats. These again can be shared just as the ADC due to steering mechanisms although it adds complexity in network configuration.

3.3.1. Sample east-west service function chains

SFC-6. SegFW : ADC : MON : AppFW

3.3.2. Sample east-west SFC description

SFC-6: In a typical three tiered architecture, requests coming to a webserver trigger interaction with application servers, which in turn trigger interaction with the database servers. It has to be noted that each of these tiers are deployed in their own segments or zones for isolation, optimization and security. SegFW enforces the security policies between the tiers and facilitates isolation at the segment level or address space re-use via NAT deployment. ADC provides the distribution, scale and resiliency to the applications while the AppFW protects and isolates traffic within the segment in addition to enforcing application specific security policies. Finally, monitoring service enables visibility into application traffic, which in turn is used to maintain application performance levels.

3.4. Multi-tenancy

Multi-tenancy is relevant in both enterprise as well as service provider data centers although it is the primary differentiator between service provider (SP) and enterprise datacenter. Enterprises treat organizations or business units within the enterprise as tenants and thus require tenant aware service models.

Multi-tenant service delivery is achieved in two primary ways: a) SNs themselves are tenant aware - every SN is built to support multiple tenants. b) SN instances are dedicated for each tenant. In both the cases, the SP manages the SNs.

To support multi-tenant aware service functions or SNs, traffic being serviced by a service function chain has to be identified by a tenant identifier. A tenant identifier has to be carried along with the traffic to be serviced. It is typical of tenant assets to be deployed in an isolated layer2 or layer3 domain such as VLAN, VXLAN or VRF. It has to be noted that the SNs themselves maybe deployed in different domains to suit the deployment needs of the SP and hence using the domain in which the SN is deployed is not an option. Although such a model is feasible it removes the deployment flexibility for the service providers.

3.5. SFCs in data centers

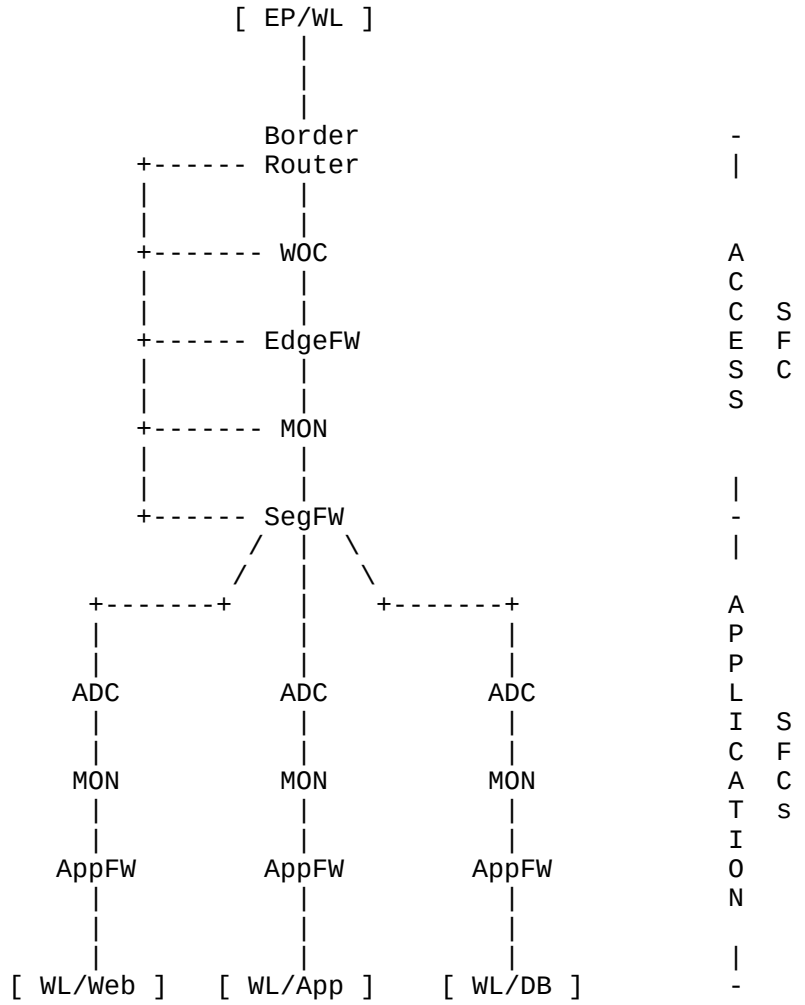


Figure 3: Service function chains in data center

Figure 3 shows the global view of SFCs applied in an enterprise or service provider data center. At a high level the SFCs can be broadly categorized into two types:

1. Access SFCs

2. Application SFCs

Access SFCs are focused on servicing traffic entering and leaving the data center while Application SFCs are focused on servicing traffic destined to applications.

Service providers deploy a single "Access SFC" and multiple "Application SFCs" for each tenant. Enterprise data center operators on the other hand may not have a need for Access SFCs depending on the size and requirements of the enterprise. Where such Access SFCs are indeed needed, such as large enterprises, the operator may deploy a bare minimum Access SFC instead. Such simple Access SFCs include WOC and VPN SFs to support the branch and mobile user traffic while at the same time utilizing the security policies in the application SFCs. The latter is the case in de-perimetrized network architectures where security policies are enforced close to the resources and applications as opposed to the WAN edge.

4. Drawbacks Of Existing Service Chaining Methods

The above use cases are realized in a traditional fashion and are not viable in the evolving hybrid data centers with virtual and physical assets. The following are some of the obvious short comings of existing SFC methods exposed by the above use cases.

DB-1. Policy based purely on VLANs is no longer sufficient. Connecting SNs to each other to construct a service chain thus makes it very static and removes deployment flexibility. As can be seen from the sample north-south service chains, a large number of VLANs not only have to be stitched in a certain fashion to achieve a basic SFC, it is simply not flexible to share the SNs among different SFCs as even simple sharing among a few SNs becomes intractable from basic configuration perspective let alone future changes or manageability aspects.

DB-2. Traffic does not always have to be steered through all the SNs of a traditional VLAN stitched service chain. In Figure 1, traffic from the border router is not always necessary to flow through the WOC as remote or mobile worker may not have a WOC peer deployed. Connecting multiple VLANs among service nodes to overcome to achieve this only aggravates the problem of deployment and manageability. Truly, there exists a need for dynamically determining the next sub SFC at such branching points to avoid forcing all

traffic through the same SFC.

DB-3. Virtual environments require the virtual SNs be migration capable just like the compute workloads. As a consequence it is simply not feasible to continue VLAN stitching in the hybrid data centers. Every time a virtual SN migrates, such as the AppFW in Figure 1 and Figure 2, the operator has to ensure the VLANs are provisioned in the destination. Further, stretching the VLANs across the network may not be an option for the operator or even worse the virtual SN may be L3 hop away from the previous SN.

DB-4. Policy Based Routing (PBR) can be used to move traffic to SNs. Although it provides a much better granularity than VLAN stitching it suffers from the requirement to configure such policies all along the path to the SNs. In Figure 1, if WOC is multiple hops away from the border router, all network elements in between border router and WOC need to be configured with consistent policies.

DB-5. Source NAT (SNAT) is required by some SNs, such as ADC in Figure 1, in order to ensure traffic sent to the load balanced servers pass through the ADC in reverse direction. However, SNAT removes the ability to detect the originator of the traffic. Using HTTP extension header to pass originator information is not only an overhead but addresses only one specific protocol.

DB-6. Static service chains do not allow for modifying the SFCs as they require the ability to add SNs or remove SNs to scale up and down the service capacity. Likewise the ability to dynamically pick one among the many SN instance is not available. For instance, WOC must scale to support the high data rate of traffic flowing to the data center. Likewise, AppFWs must scale up to not impact the workload throughput. Further they may be required to scale within tenant boundaries.

DB-7. Static SFCs constructed over the under lay network cannot pass metadata to the SNs. Border Router in Figure 1 cannot pass policy based tags derived locally at the start of the SFC all the way through the SFC. Such metadata is necessary to enforce consistent security policies across the network, as one example.

DB-8. In multi-tenant deployments, the segment on which the SN is deployed may not correspond to the segment assigned to the tenant in which the workloads are hosted. In Figure 2, AppFW may be deployed on a different segment than the Workload. As a consequence, it is not viable to derive the tenant segment simply based on the tag associated with the incoming traffic at the AppFW. This ultimately prevents the ability to have the same SN serve multiple tenants. Forcing the SN to be on the same segment as the tenants' workload limits deployment flexibility.

DB-9. Traffic may originate in a physical or virtual network or transit these networks before being delivered to the SNs for servicing. The following is very complex to achieve with the existing SFC mechanism, primarily due to very conflicting nature of their environments: physical and static vs. virtual and dynamic.

A. Physical SN servicing traffic originating in the virtual access network.

B. Virtual SN servicing traffic originating in the physical network.

DB-10. Although SNs are purpose built service appliances, it is neither a requirement nor an indication of how service functions are implemented in emerging data centers with commodity compute and storage capabilities. AppFW in Figure 1, for instance, may be built and deployed as a virtual SN. Further, SFCs are limited to exclusively physical or virtual SNs and not a mix. This excludes the ability to combine the benefits offered by physical SNs with the flexibility and agility of the virtual SNs. The EdgeFW in Figure 1, for instance, may be a purpose built SN to take advantage of SFs implemented in hardware while the AppFW may be a virtual SN deployed to be close to the virtual workload and may even move with the workload in the virtual environment.

DB-11. Troubleshooting is one of the predominant issues plaguing SFC deployments. The reasons range from misconfiguration at different elements in the network that are responsible for directing traffic to the service nodes, to, tracking the traffic paths starting from the point of entry into the DC to the point of exit to the application through various SNs. When desired services are not effective on certain traffic, determining the reason is simply not viable in a large scale deployment. Figure 3 provides a view of the complexity in

terms of the permutations of the SFCs, their paths in the network and the configuration in the network elements required and managed for proper operation.

5. General Requirements

The above use cases and the drawbacks thereof lead to the following general requirements in today’s evolving hybrid datacenters to apply SFCs to traffic.

GR1. SFC polices MUST be applicable at the edges - network elements as well as the workloads.

GR2. SFC policies MUST be applicable to either Ingress or Egress traffic.

GR3. SFC MUST support virtual as well as physical SNs.

GR4. SFC SHOULD support the ability to mix virtual and physical SNs in the same SFC.

GR5. SFC SNs MUST be deployable L2 or L3 hop away from each other or from the SFC starting entity.

GR6. SFC traffic MUST be allowed to follow paths not constrained by the underlying static network topology.

GR7. SFC SNs MUST be able to derive the tenant identification without being tied to the underlying topology

GR8. SFCs MUST support the ability to pass metadata among the SNs or between the SNs and the network elements.

GR9. A composite SFC SHOULD be achievable by way of joining sub SFCs, branching to sub SFCs where necessary.

GR10. SFCs SHOULD NOT require SNAT inside the SFs to attract traffic back to them

GR11. SFCs SHOULD have the ability to choose SN instances dynamically, at the time of forwarding traffic to them.

GR12. An OAM mechanism to easily troublshoot as well as validate the paths traversed by the SFCs SHOULD be supported.

6. Acknowledgements

The authors would like to thank Paul Quinn, Jim Guichard, Jim French, Larry Kreegar and Nagaraj Bagepalli for their review and comments.

A special thanks to Abhijit Patra for his guidance.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

Security of traffic being serviced is very important in the use cases described in this document. The SNs deployed as part of the SFC are expected to include SFs specifically addressing the security aspect either individually or in concert with other SFs. In this regard organizational security policies are expected to drive the security posture adapted in the SFCs. However, securing the traffic moving between the SFs or SNs is not a consideration beyond the methods used for moving such traffic.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[I-D.ietf-sfc-problem-statement] Quinn, P. and T. Nadeau, "Service Function Chaining Problem Statement", draft-ietf-sfc-problem-statement-05 (work in progress), April 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.

Authors' Addresses

Surendra Kumar
Cisco Systems, Inc.
170 W. Tasman Dr.
San Jose, CA 95134

Email: smkumar@cisco.com

Cesar Obediente
Cisco Systems, Inc.
7200-10 Kit Creek Rd.
Resarch Triangle Park, NC 27709-4987

Email: cobedien@cisco.com

Mudassir Tufail
Citi
238 King George Rd
Warren, NJ 07059-5153

Email: mudassir.tufail@citi.com

Sumandra Majee
F5 Networks
90 Rio Robles
San Jose, CA 95134

Email: S.Majee@f5.com

Claudiu Captari
Telstra Corporation
Level 15, 525 Collins Street
Melbourne, 3000

Email: Claudiu.Captari@team.telstra.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 21, 2015

W. Liu, Ed.
H. Li
O. Huang
Huawei Technologies
M. Boucadair, Ed.
France Telecom
N. Leymann
Deutsche Telekom AG
Q. Fu
China Mobile
Q. Sun
China Telecom
C. Pham
Telstra Corporation
C. Huang
Carleton University
J. Zhu
Huawei Technologies
P. He
Ciena Corp

September 17, 2014

Service Function Chaining (SFC) General Use Cases
draft-liu-sfc-use-cases-08

Abstract

The delivery of value-added services relies on the invocation of advanced Service Functions in a sequential order. This mechanism is called Service Function Chaining (SFC). The set of involved Service Functions and their order depends on the service context and other deployment-specific considerations.

Having a single use case document eases the effort of deriving requirements that are to be met by SFC solution(s). Moreover, it allows to identify commonalities between the various use cases that are of interest. This document presents a set of general use cases of Service Function Chaining (SFC).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute

working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 21, 2015.

Table of Contents

1. Introduction

The delivery of Value-Added Services (VAS) relies on the invocation of various Service Functions (SFs). Typically, the traffic is forwarded through a set of Network Elements embedding Service Functions, e.g.:

a. Direct a portion of the traffic to a Network Element for monitoring and charging purposes.

b. Before sending traffic to DC servers, steer the traffic to cross a load balancer to distribute the traffic load among several links, Network Elements, etc.

c. Mobile network operators split mobile broadband traffic and steer them along an offloading path.

d. Use a firewall to filter the traffic for IDS (Intrusion Detection System)/IPS (Intrusion Protection System).

e. Use a security gateway to encrypt/decrypt the traffic. SSL offloading function can also be enabled.

f. If the traffic has to traverse different networks supporting distinct address families, for example IPv4/IPv6, direct the traffic to a CGN (Carrier Grade NAT, [RFC6888][RFC6674]) or NAT64 [RFC6146].

g. Some internal service platforms rely on implicit service identification. Dedicated Service Functions are enabled to enrich packets (e.g., HTTP header enrichment) with the identity of the subscriber or the UE (User Equipment).

h. Operators offer VAS on a per subscription basis. It is desirable to steer traffic only from the subscribers, who have subscribed to VAS, to the relevant service platforms.

Having a single use case document eases the effort of deriving requirements that are to be met by SFC solution(s). Moreover, it allows to identify commonalities between the various use cases that are of interest. This document describes some use cases of Service Function Chaining (SFC). It is not the purpose of this document to be exhaustive, but instead, we try to draw the set of deployments context that are likely to see SFC solutions deployed.

For most of the use cases presented in this document:

- Instantiated SFC are driven by business and engineering needs.

- The amount of instantiated SFCs can vary in time, service engineering objectives and service engineering choices.

- The amount of instantiated SFCs are policy-driven and are local to each administrative entity.

- The technical characterization of each Service Function is not frozen in time. A Service Function can be upgraded to support new features or disable an existing feature, etc.

- Some stateful SFs (e.g., NAT or firewall) may need to treat both outgoing and incoming packets. The design of SF Maps must take into account such constraints, otherwise, the service may be disturbed. The set of SFs that need to be invoked for both direction is up to the responsibility of each administrative entity operating an SFC-enabled domain.

- For subscription-based traffic steering, subscriber-awareness capability is required. A UE is allocated a dynamic IPv4 address and/or IPv6 prefix when attaching to a network. This IPv4 address and/or IPv6 prefix can change from time to time. The requirement is to be able to correlate an IPv4 address and/or IPv6 prefix to a subscriber identity from that will be used to trigger the invocation of some Service Functions.

- Some Service Functions may be in the same subnet; while others may not. Service Functions are deployed directly on physical hardware, as one or more Virtual Machines, or any combination thereof.

2. Terminology

This document makes use of the terms defined in [I-D.ietf-sfc-architecture].

Service Flow: packets/frames with specific service characteristics (e.g., packets matching a specific tuple of fields in the packet header and/or data) or determined by some service-inferred policies (such as access port and etc.).

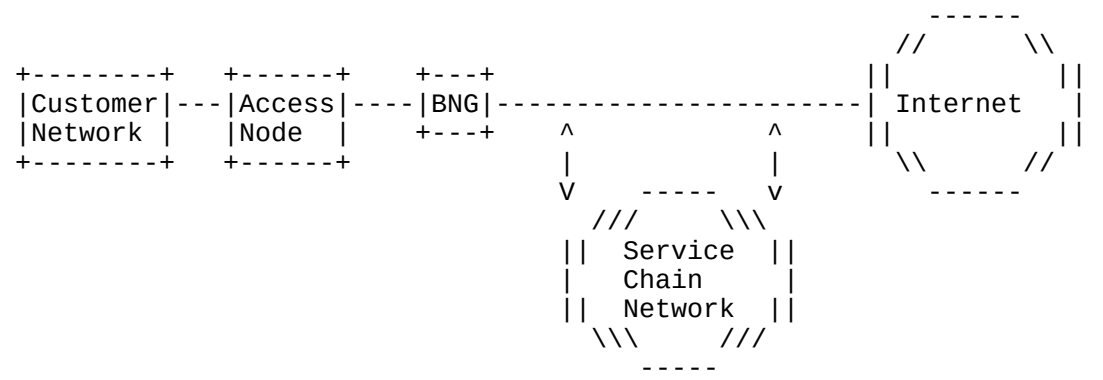
Gi interface: 3GPP defines the Gi interface as the reference point between the GGSN (Gateway GPRS Support Node) and an external PDN (Packet Domain Network). This interface reference point is called SGi in 4G networks (i.e., between the PDN Gateway (PGW) and an external PDN)[RFC6459].

3. Service Function Chaining Use Cases

Service Function Chains can be deployed in a diversity of scenarios such as broadband networks, mobile networks, and DC center. This section describes a set of scenarios for Service Function Chaining deployment. Please note that for each SFC there is a corresponding symmetrical reverse chain, so we added bidirectional links to each SFC use case figure below. For those links that do not have a directional mark, they are bidirectional by default.

3.1. Service Function Chain in Fixed Broadband Networks

In fixed broadband networks, users may be accessed into the network via different technologies, which typically includes DSL, Ethernet and PON. Whatever the access technology is, the architecture for access and metro network is similarly comprised of Access Nodes (ANs) and Broadband Network Gateways (BNGs), where the AN is usually a device providing access to network for customers with variant access technologies and the BNG is the first IP node and providing subscriber authorization, authentication and accounting.



Service Function Chain in Fixed Broadband Networks

Figure 1: An example of Service Function Chain in Broadband Networks

Figure 1 illustrates a fixed broadband network with Service Chaining. Service Chain Network is deployed behind BNG and before Internet. The Service Function Chain in Figure 1 may include several Service Functions to perform services such as DPI, NAT44, DS-Lite, NPTv6, Parental control, Firewall, load balancer, Cache, etc.

The Broadband Forum (BBF) is developing a study document (SD-326) about Flexible Service Chaining, whose scope includes identifying and

documenting use cases relevant to fixed broadband networks. Though SD-326 is an internal project within BBF, the content related to use cases has been communicated to IETF per the following Liaison Statement: Broadband Forum Work on Flexible Service Chaining (SD-326) (http://datatracker.ietf.org/liaison/1304/). As BBF is the leading organization on fixed broadband network architectures, this liaison will serve as reference for service chaining use cases applicable in such fixed broadband context. Future liaison statements from BBF may provide additional use cases, and will be referenced here as appropriate.

3.2. Service Function Chain in Mobile Networks: Brief Overview

3GPP defines the Gi interface as the reference point between the GGSN (Gateway GPRS Support Node) and an external PDN (Packet Domain Network) [RFC6459]. This interface reference point is called SGi in 4G networks (i.e., between the PDN Gateway (PGW) and an external PDN) [RFC6459]. Note, there is no standard specification of such reference points (i.e., Gi and SGi) in terms of functions to be located in that segment.

> Note: The use cases do not include 3GPP release details. For more information on the 3GPP releases detail, the reader may refer to Section 6.2 of [RFC6459].

Traffic is directed to/from Internet traversing one or more Service Functions. Note, these Service Functions are called "enablers" by some operators. One example of enabler function is a HTTP Header Enrichment Function. There are also other VAS function such as Parental Control or network-based Firewall. Subscribers can opt-in and opt-out to these services anytime using a self-served portal or by calling the Operator’s customer service.

In light of current deployments, plenty of Service Functions are enabled in the Gi Interface (e.g., DPI, billing and charging, TCP optimization, web optimization, video optimization, header enrichment, etc.). Some of these Service Functions are co-located on the same device while others are enabled in standalone boxes. In order to fulfill new business needs, especially to offer innovative added-value services, the number of enabled Service Functions in the Gi Interface is still growing. Some of these functions are not needed to be invoked for all services/UEs, e.g.,: TCP optimization function only for TCP flows, HTTP header enrichment only of HTTP traffic, Video optimization function for video flows, IPv6 firewall + NAT64 function for outgoing IPv6 packets, IPv4 firewall + NAT64 function for incoming IPv4 packets, etc..

3GPP has defined Traffic Detection Function (TDF) which implements DPI (detection) functionality along with enforcement and charging of the corresponding detected applications [TS.23203]. TDF resides on Gi/SGi interface.

> Note: It was tempting to use TDF and DPI terms interchangeably, but given the diversity of deployments involving DPI modules the text uses DPI to refer to legacy deployments. The behavior of such DPI modules is deployment-specific.

Several (S)Gi Interfaces can be deployed within the same PLMN (Public Land Mobile Network). This depends mainly on the number of PDNs and other factors. Each of these interfaces may involve a differentiated set of Service Functions to be involved.

More details about SFC usage within Mobile Networks can be found in [I-D.ietf-sfc-use-case-mobility].

3.3. Common Service Chain Network: A Convergence Tool

From previous two use cases, we can see commonalities in service chaining. Even though fixed and mobile broadband networks are deployed separately, for integrated operators that running both networks it is obviously beneficial to provide service chaining to both networks from a common service chain network.

In addition to resource optimisation, a common service chain network can also enable seamless service switchover from one network to the other. For example, a customer is watching football game on his mobile phone via 3G network. After he arrives home, he can switch over to the WLAN on his home gateway, which is backhauled to the network by Fiber To The Home (FTTH, a typical PON service), 100 Mbps broadband access. In the case, it is easier to provide seamless service from a common service chain network.

SFC can be used as a tool to better address convergence needs (including adjust the service delivery to the access network constraints or to the capabilities of connected devices).

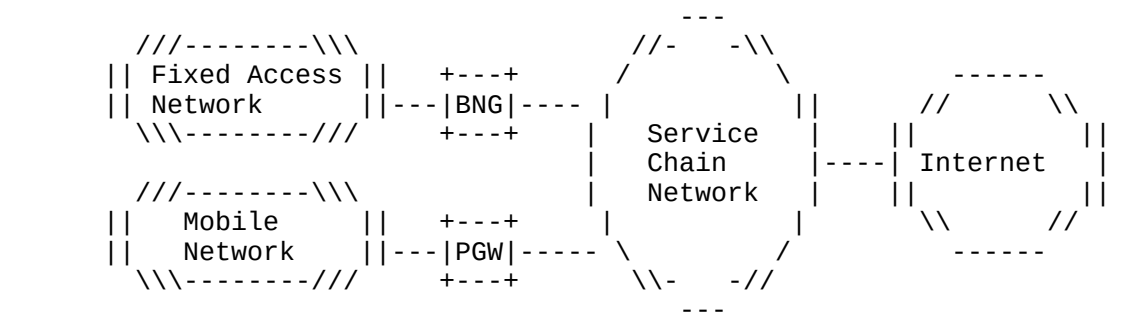


Figure 2: Common Service Chain Network

Figure 2 illustrates a common Service Chain Network is shared by both fixed and mobile broadband networks. Such a common service chain network can be deployed as consisting of only network nodes with specific functions, or in a data center. In both cases, service nodes, whether physical or virtual, are shared by both wireline and wireless networks. Operators manage service chains universally for both networks and traffic from both networks may go through the same service chain.

3.4. Distributed Service Function Chain

Besides the deployment use cases listed above, a Service Function Chain is not necessarily implemented in a single location but can also be distributed crossing several portions of the network (e.g., data centers) or even using a Service Function that is located at an network element close to the customer (e.g. certain security functions).

Multiple SFC-enabled domains can be enabled in the same administrative domain.

For steering traffic to subscription-based Service Functions, the SFC Classifier needs to understand which subscriber a flow belong to in order to retrieve the service profile to apply to this flow. In some contexts, it is not possible to identify in a permanent manner the subscriber by the source IP address because that IP address may be assigned dynamically. Out-of-band methods to correlate the source IP address and a subscriber identifier may be needed in a given administrative domain. The SFC Classifier can rely on pull or push methods to correlate an IP address and/or IPv6 prefix to a subscriber identity. Examples are querying the PCRF or receiving RADIUS Accounting messages respectively.

For steering traffic to traffic management Service Functions such as video optimisation platform, in mobile network, it is desirable to perform optimisation on when required. That is when there is congestion in the Radio cells. One option for the SFC Classifier to have this congestion-awareness is for the network to provide this information to the SFC Classifier, directly, or via an intermediate actionable-intelligence function, which can combine other inputs or policies. How those policies and feedback data are configured to the SFC Classifier may be specific to each administrative domain.

3.5. Service Function Chain in Data Center

In DC (Data Center), like in broadband and mobile networks, Service Function Chains may also be deployed to provide added-value services.

Figure 3 illustrates a possible scenario for Service Function Chain in Data Center: SFs are located between the DC Router (access router) and the Servers. From Servers to Internet, there are multiple Service Functions such as IDS/IPS, FW, NAT lined up and a monolithic SFC created for all incoming traffic.

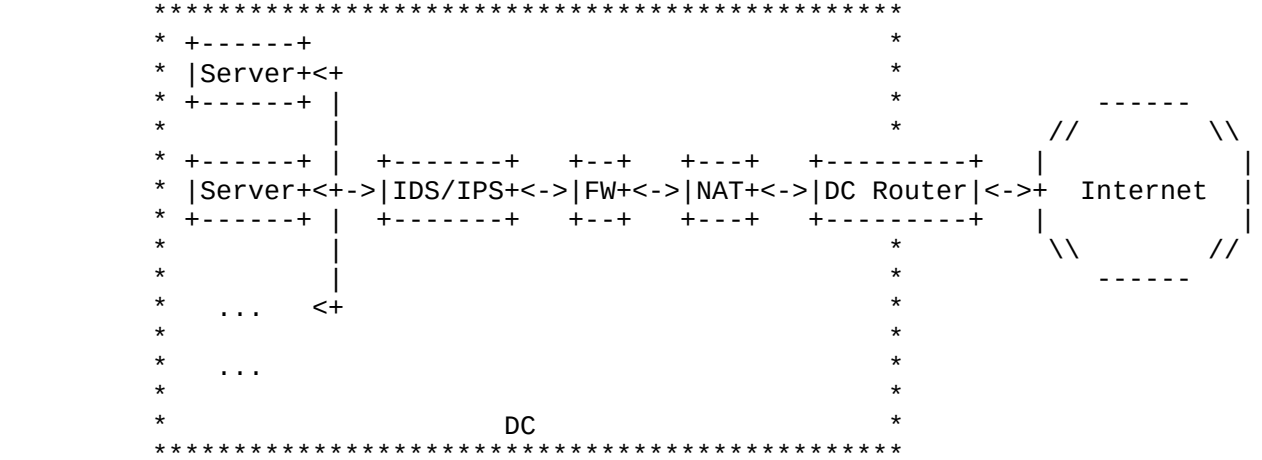


Figure 3: Service Function Chain in Data Center

More details about Service Function Chain in Data Center can be found in [I-D.ietf-sfc-dc-use-cases].

3.6. Service Function Chain in Cloud CPE

Cloud CPE is one deployment scenario where the value-added service functions are centralized (e.g., hosted in the network or cloud side), leaving the subscriber side box with basic L2/L3

functionalities. In this scenario, all the value added services are configured by subscribers and enabled in the network side.

Subscribers can define their own added value services. The Cloud CPE will translate those services requests into chains of Service Functions. Such architecture must support means to differentiate subscribers and their traffic.

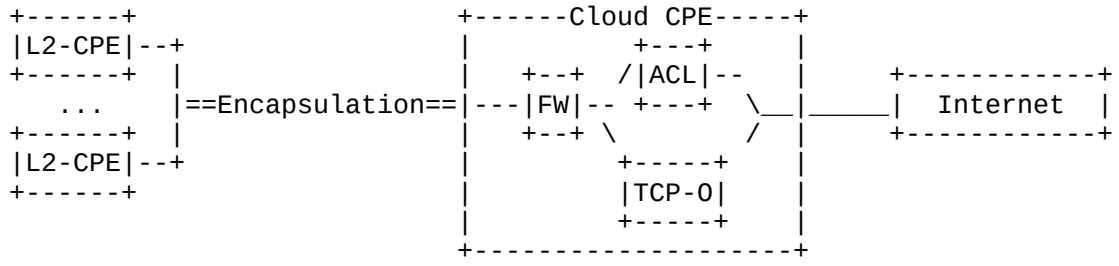


Figure 4: SFC in Cloud CPE

4. Abstraction of SFC in Different Deployment Scenarios

This section presents the SFC scenarios from a different angle, i.e., the abstraction of SFC use cases in different deployment scenarios. Each of the use case may belong to one or many of the categories listed below:

Category	Description
Per-service Characteristic SFC	Chain different Service Functions based on service/application characteristics
Per-user/subscription SFC	Chain different Service Functions based on user requirements or subscription information. Note, this does not mean that millions of SFCs will be instantiated but SF classification is subscriber-aware.
TE-Oriented SF	Chain different Service Functions for Traffic Engineering purposes. This may includes load, utilisation, planned maintenance, etc.
Bi-directional Flow SFC	Function path that contain bi-directional Service Functions
SFC over Multi-Underlay Networks	Service Functions distributed over different underlay networks
SFC over Service Functions Forking	SFC that contains the paths for different service or applications

4.1. Per-service characteristic SFC

The traffic in a network is usually forwarded based on destination IP or MAC addresses. In an operator's network, some Service Functions are implemented, where traffic is steered through these Service Functions in a certain sequence according to service characteristics and objectives.

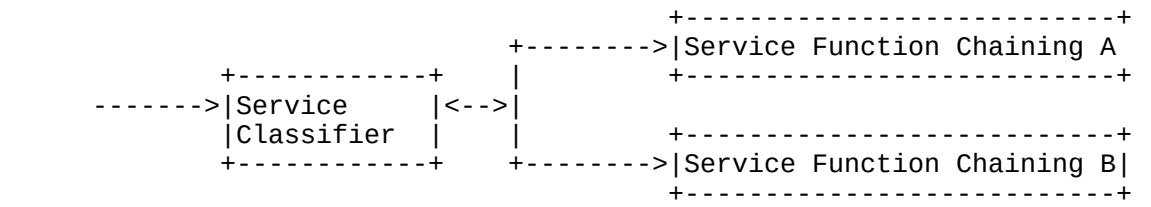


Figure 5: General Service Function Chain

Traffic enters a SFC-enabled domain in a service classifier, which identifies traffic and classifies it into service flows. Service flows are forwarded on a per SF Map basis.

4.2. Per-user/subscription requirement SFC

In operator networks with user subscription information, it is considered as a value added service to provide different subscribers with differentiated services. Subscribers may subscribe different services and the order handling at the operator side will translate those subscription request into configuration operations so that the service will be appropriately delivered to the subscribers. Configuration operations include in particular the provisioning of classification rules.

4.3. TE-Oriented SFC

TE-oriented SFC is required by operators in achieving flexible service operating. For example, if certain paths are congested or certain Service Functions are overloaded, SFC forwarding should be inferred accordingly.

4.4. SFC for Bi-directional Flow

Some Service Functions, for example, NAT or TCP optimization, need to handle bi-directional flows, while others SFs such as video optimization don’t need to handle bi-directional flows.

Due to IPv4 address exhaustion, more and more operators have deployed or are about to deploy IPv6 transition technologies such as NAT64 [RFC6146]. The traffic traversing a NAT64 function may go through different types of IP address domains. One key feature of this scenario is that characteristics of packets before and after processed by the service processing function are different, e.g., from IPv6 to IPv4. The unpredictability of processed packets, due to the algorithm in the Service Function, brings difficulties in steering the traffic.

A variety of hosts can be connected to the same network: IPv4-only, dual-stack, and IPv6-only. A differentiated forwarding path can be envisaged for each of these hosts. In particular, DS hosts should not be provided with a DNS64, and as such there traffic should not be delivered to a NAT64 device. Means to guide such differentiated path can be implemented at the host side; but may also be enabled in the network side as well.

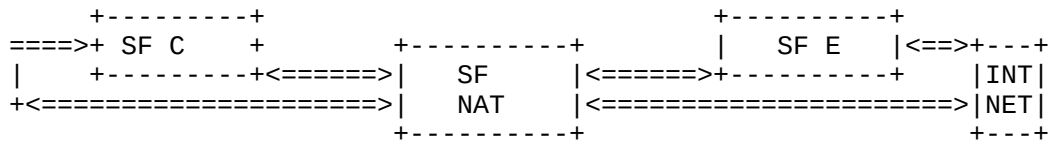


Figure 6: Service Function Chain with NAT64 function

Figure 6 shows a specific example of Service Function Chain with NAT function. Service flow1 is processed by SF(Service Function) C, NAT and E sequentially. In this example, the SF NAT performs NAT64. As a result, packets after processing by the SF NAT are in IPv4, which is a different version of IP header from the packets before processing. Service Function Chaining in this scenario should be able to identify the flow even if it is changed after processed by Service Functions.

4.5. SFC over Multiple Underlay Networks

Operators may need to deploy their networks with various types of underlay technologies. Therefore, Service Function Chaining needs to support different types of underlay networks.

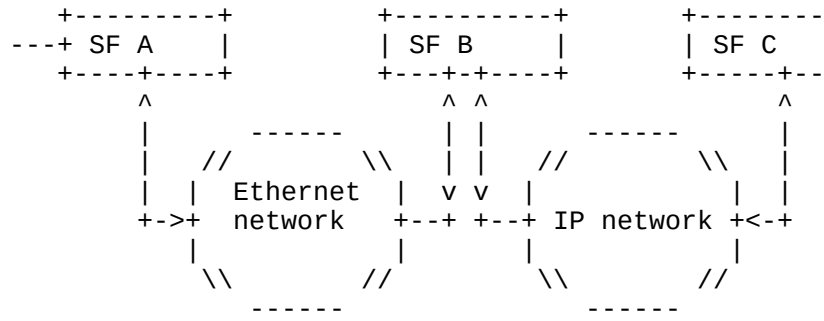


Figure 7: multiple underlay networks: Ethernet and IP

Figure 7 illustrates an example of Ethernet and IP network, very common and easy for deployment based on existing network status, as underlay networks. SF(Service Functions) A, B and C locate in Ethernet and IP networks respectively. To build a Service Function Chain of SF A, B and C, Service Function Chaining needs to support steering traffic across Ethernet and IP underlay networks.

4.6. SFC over Service Path Forking

To enable service or content awareness, operators need DPI functions to look into packets. When a DPI function is part of a Service Function Chain, packets processed by the DPI function may be directed to different paths according to result of DPI processing. That means a forking service path.

In this use case, the switching SF is another classifier which need to classify flow and shepherd them to different paths.

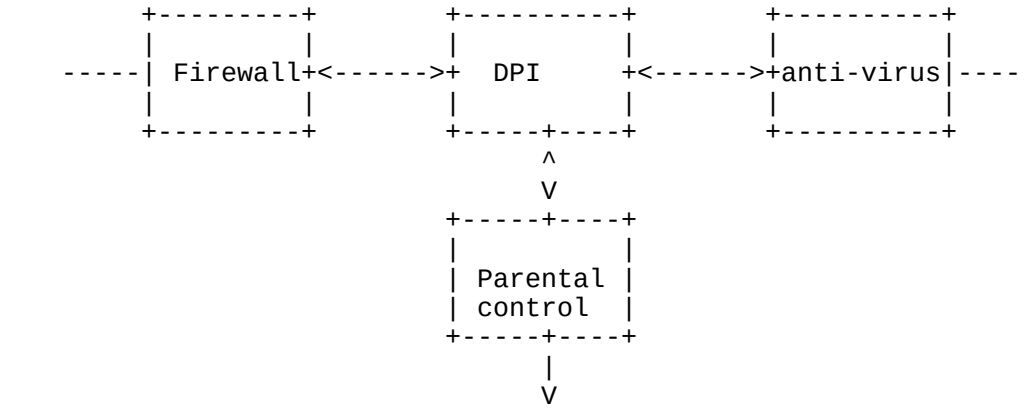


Figure 8: a forking service path

Figure 8 shows the use case of a forking service path. Traffic first goes through a firewall and then arrives at DPI function which discerns virus risk. If a certain pre-configured pattern is matched, the traffic is directed to an anti-virus function.

Such DPI function may fork out more than one path.

Service function sharing is sub-category of the service function forking. Some carrier grade hardware box or Service Functions running on high performance servers can be shared to support multiple Service Function Chains. Following is an example.

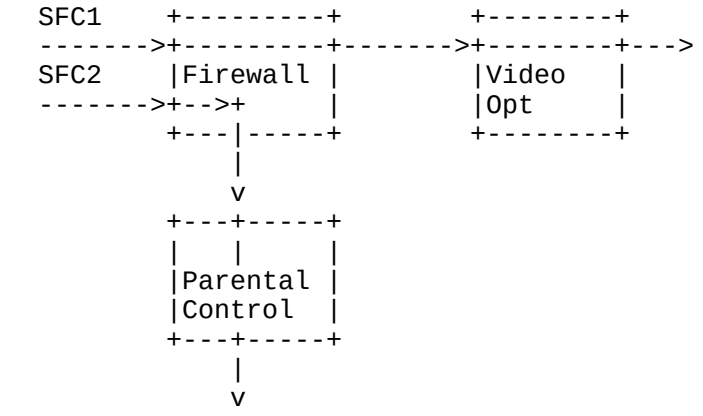


Figure 9: Two Service Function Chains share one Service Function

In Figure 9, there are three Service Functions, firewall, VideoOpt and Parental Control, and two Service Functions Chains SFC1 and SFC2. SFC1 serves broadband user group1 which subscribes to secure web surfing and Internet video optimization, while SFC2 serves broadband user group2 which subscribes to secure web surfing with parental control. SF Firewall is shared by both Service Function Chains.

4.7. Recursive SFC

SFCs can be provided in a recursive manner. A Level 1 service provider can sell SFC services to multiple clients. Each client can further partition its SFC and serve as a Level 2 service provider to sell differentiated SFCs to different clients. This process can continue several iterations making recursive service a new business model which is becoming popular today.

Consider a use case where an enterprise leases a virtual service network from a data canter provider. The enterprise then creates two service chains out of the virtual service network. The first service chain, designed for its employees, will force traffic flows to go through NAT, DPI, firewall, LB, and various servers. The second one, designed for its customers, will only go through NAT and web servers. Its customers can create specific websites for different departments such as purchase department, service department, etc.

An important characteristic of recursive service is that each service provider is a separate entity who owns the SFC it purchased from lower level provider and who also decides the SFCs it creates for its clients.

5. Security Considerations

This document does not define an architecture nor a protocol. It focuses on listing use cases and typical Service Function examples. Some of these functions are security-related.

SFC-related security considerations are discussed in [I-D.ietf-sfc-architecture].

6. Acknowledgements

Jie Hu and Zhen Cao contributed to an earlier version of this document.

This document has benefited from reviews, suggestions, comments and proposed text provided by the following members of the Service Function Chaining Working Group(sfc WG), listed in alphabetical order: David Binet, Hui Deng, Alla Goldner, Yuanlong Jiang, Jerome Moisand, Lehong Niu, Ron Parker, and Lucy Yong.

7. Informative References

[I-D.ietf-sfc-architecture]
Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", draft-ietf-sfc-architecture-01 (work in progress), September 2014.

[I-D.ietf-sfc-dc-use-cases]
Surendra, S., Tufail, M., Majee, S., Captari, C., and S. Homma, "Service Function Chaining Use Cases In Data Centers", draft-ietf-sfc-dc-use-cases-01 (work in progress), July 2014.

[I-D.ietf-sfc-use-case-mobility]
Haeffner, W., Napper, J., Stiemerling, M., Lopez, D., and J. Uttaro, "Service Function Chaining Use Cases in Mobile Networks", draft-ietf-sfc-use-case-mobility-01 (work in progress), July 2014.

[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.

[RFC6459] Korhonen, J., Soininen, J., Patil, B., Savolainen, T., Bajko, G., and K. Iisakkila, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)", RFC 6459, January 2012.

[RFC6674] Brockners, F., Gundavelli, S., Speicher, S., and D. Ward, "Gateway-Initiated Dual-Stack Lite Deployment", RFC 6674, July 2012.

[RFC6888] Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", BCP 127, RFC 6888, April 2013.

[TS.23203] 3GPP, "Policy and charging control architecture", December 2013.

Authors' Addresses

Will(Shucheng) Liu (editor)
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

Email: liushucheng@huawei.com

Hongyu Li
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

Email: hongyu.li@huawei.com

Oliver Huang
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

Email: oliver.huang@huawei.com

Mohamed Boucadair (editor)
France Telecom
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Nicolai Leymann
Deutsche Telekom AG

Email: n.leymann@telekom.de

Qiao Fu
China Mobile

Email: fuqiao@chinamobile.com

Qiong Sun
China Telecom
No.118 Xizhimennei street, Xicheng District
Beijing 100035
P.R. China

Email: sunqiong@ctbri.com.cn

Chuong Pham
Telstra Corporation
Level 8, 18 Smith Street
Parramatta 2150
Australia

Email: Pham, Chuong D <Chuong.D.Pham@team.telstra.com>

Changcheng Huang
Carleton University
1125 Colonel By Drive
Ottawa ON K1S 5B6
Canada

Email: huang@sce.carleton.ca

Jiafeng Zhu
Huawei Technologies
Santa Clara,CA
US

Email: Jiafeng.zhu@huawei.com

Peng He
Ciena Corp

Email: phe@ciena.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 6, 2014

P. Quinn, Ed.
Cisco Systems, Inc.
J. Halpern, Ed.
Ericsson
May 5, 2014

Service Function Chaining (SFC) Architecture
draft-quinn-sfc-arch-05.txt

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. This document does not propose solutions, protocols, or extensions to existing protocols.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 6, 2014.

Table of Contents

1. Introduction

This document describes an architecture used for the creation of Service Function Chains (SFC) in a network. It includes architectural concepts, principles, and components.

Service Function Chaining enables the creation of composite services that consist of an ordered set of Service Functions (SF) that must be applied to packets and/or frames selected as a result of classification. Each SF is referenced using an identifier that is unique within an administrative domain. No IANA registry is required to store the identity of SFs.

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.

1.1. Scope

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. Definition of Terms

Classification: Locally instantiated policy and customer/network/service profile matching of traffic flows for identification of appropriate outbound forwarding actions.

SFC Network Forwarder (NF): SFC network forwarders provide network connectivity for service function forwarders (SFF) and service functions (SF).

Service Function Forwarder (SFF): A service function forwarder is responsible for delivering traffic received from the SFC network forwarder to one or more connected service functions via information carried in the SFC encapsulation.

Service Function (SF): A function that is responsible for specific treatment of received packets. A Service Function can act at the network layer or other OSI layers. A Service Function can be a virtual instance or be embedded in a physical network element. One of multiple Service Functions can be embedded in the same network element. Multiple instances of the Service Function can be enabled in the same administrative domain.

A non-exhaustive list of Service Functions includes: firewalls, WAN and application acceleration, Deep Packet Inspection (DPI), server load balancers, NAT44 [RFC3022], NAT64 [RFC6146], HOST_ID injection, HTTP Header Enrichment functions, TCP optimizer, etc.

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 service does not contain the SFC encapsulation.

Service Function Identity (SFID): A unique identifier that represents a service function. SFIDs are unique for each SF within an SFC domain.

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.

Note: The term "service" is overloaded with varying definitions. For example, to some a service is an offering composed of several elements within the operators network whereas for others a service, or more specifically a network service, is a discrete element such as a firewall. Traditionally, these network services host a set of service functions and have a network locator where the service is hosted.

Service Node (SN): Physical or virtual element that hosts one or more service functions and has one or more network locators associated with it for reachability and service delivery.

Service Function Chain (SFC): A service Function chain defines an ordered set of service functions that must be applied to packets and/or frames selected as a result of classification. The implied order may not be a linear progression as the architecture allows for nodes that copy to more than one branch. The term service chain is often used as shorthand for service function chain.

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

Service Function Path (SFP): The instantiation of a SFC in the network. Packets follow a service function path from a classifier through the requisite service functions

2. Architectural Concepts

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

2.1. Service Function Chains

In most networks services are constructed as a sequence of SFs that represent an SFC. At a high level, an SFC creates an abstracted view of a service and specifies the set of required SFs as well as the order in which they must be executed. Graphs, as illustrated in Figure 1, define each SFC. SFs can be part of zero, one, or many SFCs. A given 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 SF 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. SFCs can have more than one terminus.

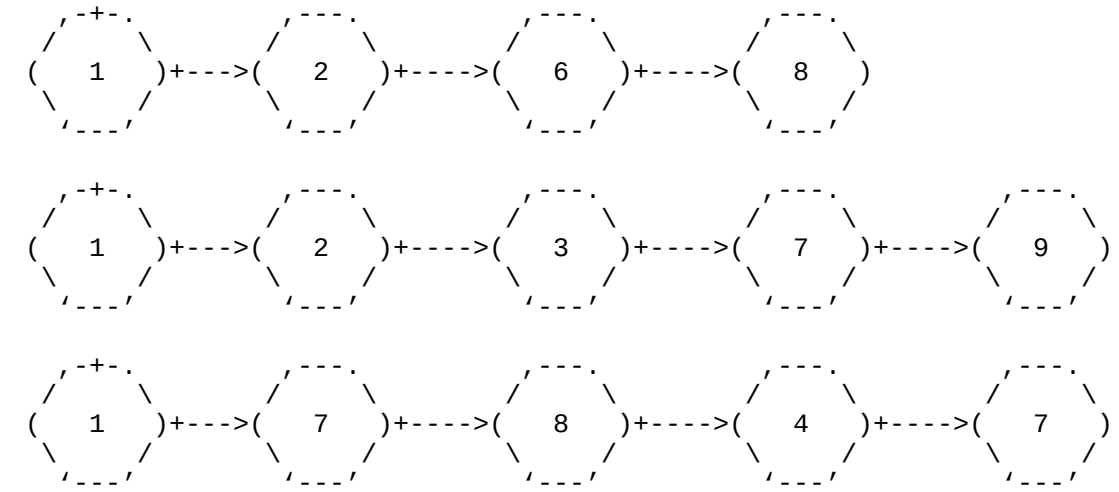


Figure 1: Service Function Chain Graphs

The architecture allows for two or more SFs to be co-resident on the

same service node. In these cases, some implementations 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. Implementation details of such mechanisms are considered out-of-scope for this document.

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). 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.

SFCs may contain cycles; that is traffic may need to traverse more than once one or more SFs within an SFC.

2.3. Service Function Paths

When an SFC is instantiated into the network it is necessary to select the specific instances of SFs that will be used, and to create the service topology for that SFC using SF’s network locator. Thus, instantiation of the SFC results in the creation of a Service Function Path (SFP) and is used for forwarding packets through the SFC. In other words, an SFP is the instantiation of the defined SFC.

This abstraction enables the binding of SFCs to specific instances, or set of like instances of SFs based on a range of policy attributes defined by the operator. For example, an SFC definition might specify that one of the SF elements is a firewall. However, on the network, there might exist a number of instances of the same firewall (that is to say they enforce the same policy) and only when the SFP is created is one of those firewall instances selected. The selection can be based on a range of policy attributes, ranging from simple to more elaborate criteria.

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. Consistent policy identifiers: a common identifier is used for SF policy selection.

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

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

   Generally speaking, the metadata can be thought of as providing, and sharing the result of classification (that occurs with the SFC 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 in turn imposes that information in the SFC encapsulation for delivery to the requisite SFs. The SFs in turn utilize the user/subscriber information for local policy decisions.

5. Heterogeneous control/policy points: allowing SFs to use independent mechanisms (out of scope for this document) like IF-MAP or Diameter to populate and resolve local policy and (if needed) local classification criteria.

4. Core SFC Architecture Components

The following sub-sections provide details on each logical component that form the basis of the SFC architecture. An overview of how each of these components interact is provided in the following figure.

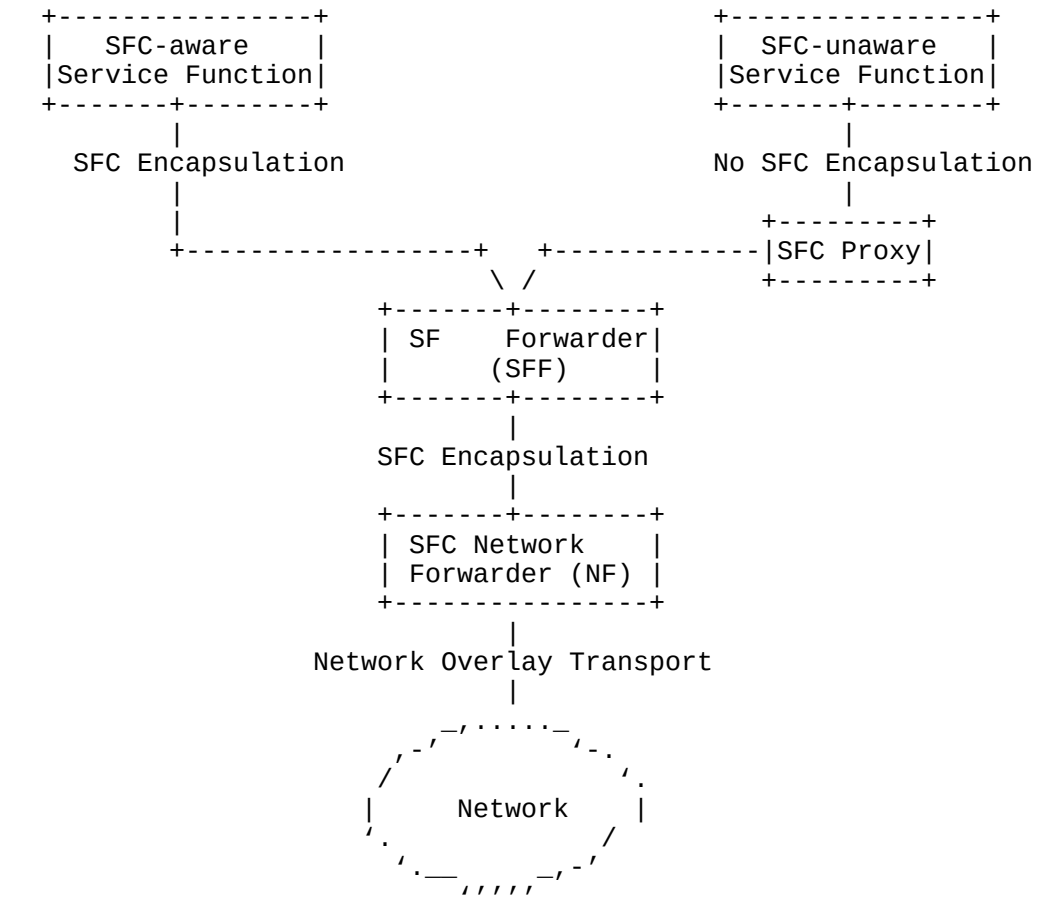


Figure 2: Service Function Chain Architecture Components

4.1. SFC Encapsulation

The SFC encapsulation enables service function path selection and the sharing of metadata/context information.

The SFC encapsulation provides 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. The SFC encapsulation therefore, relies on an outer network transport. Transit nodes -- such as router and switches -- simply 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 and as such any network transport protocol may be used to carry the SFC encapsulation.

4.2. Service Function (SF)

The concept of a SF evolves; rather than being viewed as a bump in the wire, a SF becomes a resource within a specified administrative domain that is available for consumption as part of a composite service. As such, SFs have one or more network locators through which they are reachable, and a variable set of attributes that describe the function offered. The combination of network locator and attributes are used to construct an SFP. SFs send/receive SFC encapsulated data from one or more SFFs.

While the SFC architecture defines 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 a logical SFC proxy function is defined. The SFC proxy acts a gateway between the SFC encapsulation and SFC unaware SFs. The integration of SFC-unaware service function is discussed in more detail in the SFC proxy section.

4.3. Service Function Forwarder (SFF)

The SFF is responsible for forwarding packets and/or frames received from an NF to one or more SFs associated with a given SFF using information conveyed in the SFC encapsulation.

The collection of SFFs creates a service plane using an 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 SF of a given SFC should be applied as traffic flows through the associated SFP. Each SFF need only maintain SFC forwarding information that is relevant locally. The SFC forwarding state at all SFFs collectively represents the SFPs associated with each SFC in the SFC domain.

SFP	Ordered Service Functions			
ID	order1	order2	order3	...
SFP1	SFID1	SFID5	SFID20	
SFP4	SFID100	SFID3	SFID4	SFID9
...				

Figure 3: SFF Table

Figure 3 depicts a view of the service forwarding state for two SFPs - SFP1 and SFP4. The SF columns of this table may come from different SFFs.

The SFF component has the following primary responsibilities:

1. SFP forwarding : Traffic arrives at an SFF from one or more NFs. 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 hop in the SFP, the SFF, encapsulates the traffic in the appropriate network transport and delivers it to the NF for delivery to the next SFF along the path.

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 servicing it, SFF fails to find the next SF or knows from the service forwarding state that the SFC is complete. SFF removes the SFC encapsulation and delivers the packet to an NF for forwarding.

3. Maintaining flow state: In some cases, the SFF may be stateful. It creates flows and stores flow-centric information. When traffic arrives after being steered through an SFC-unaware SF, the SFF must perform re-classification of traffic to determine the SFP. A state-full SFF simplifies such classification to a flow lookup.

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. Existing implementations may not be able to act on the SFC encapsulation. These platforms MAY opt to use a transport mechanism which carries the service path information from the SFC encapsulation, and information derived from the SFC encapsulation, to build transport 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. Network Forwarder (NF)

This component is responsible for performing the overlay encapsulation/de-capsulation and forwarding of packets on the overlay network. NF forwarding may consult the SFC encapsulation or the inner payload of an incoming packet only in the necessary cases to achieve optimal forwarding in the network.

4.5. Classification/Re-classification

Traffic that satisfies classification criteria is directed into an SFP and forwarded to the requisite service function(s). Classification is handled by a logical service classification function, and initial classification occurs at the edge of 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 directed into SFP A, or quite granular: all packets matching this 5-tuple are subject to SFP B.

As a consequence of the classification decision, the appropriate SFC encapsulation is imposed on the data prior to forwarding along the SFP.

The SFC architecture supports reclassification (or non-initial

classification) as well. As packets traverse an SFP, reclassification 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.

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 reclassifies 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. In this simple example, the DPI service function reclassified the traffic based on local application layer classification capabilities (that were not available during the initial classification step).

4.6. SFC Control Plane

The SFC control plane is responsible for constructing the SFPs; translating the SFCs to the forwarding paths and propagating path information to participating nodes - network and service - to achieve requisite forwarding behavior to construct the service overlay. For instance, a SFC construction may be static - using specific SF instances, or dynamic - choosing service explicit SF instances at the time of delivering traffic to the SF. In SFC, SFs are resources; the control plane advertises their capabilities, availability and location. 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 administrative domain wide view of all available service function resources as well as the network locator through which they are reachable.

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

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

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

5. Allocation of metadata associated with a given SFP and propagation of metadata syntax to relevant SF instances and/or SFC encapsulation-proxies or their respective policy planes.

4.7. 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. This component is optional. 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

4.8. 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 the network path(s) between service nodes, characteristics of the nodes themselves such as memory, number of virtual interfaces, routes, etc..., and configuration of the SFs running on the service nodes.

5. The Role of Policy

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

Specifically, it is assumed that 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 which direct the control logic to change the service chain placement or the 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. Depending upon operator preferences, these policies may also determine which sequences of functions are valid and 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.

6. Load Balancing Considerations

Supporting function elasticity and high-availability shouldn’t overly complicate SFC or lead to unnecessary scalability problems.

In the simplest case, where there is only a single function in the chain (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 case where the classifier is separate from the single function or a function at the terminal address may need sub-prefix or per subscriber metadata, we would have a single chain (the metadata changes but the SFC chain does not), regardless of the number of potential terminal addresses for the flow. This is the case of the simple load balancer.

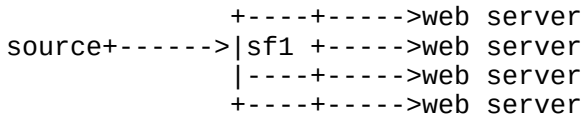


Figure 4: Simple Load Balancing

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 the following figure, we have a chain of five service functions between the traffic source and it’s destination.

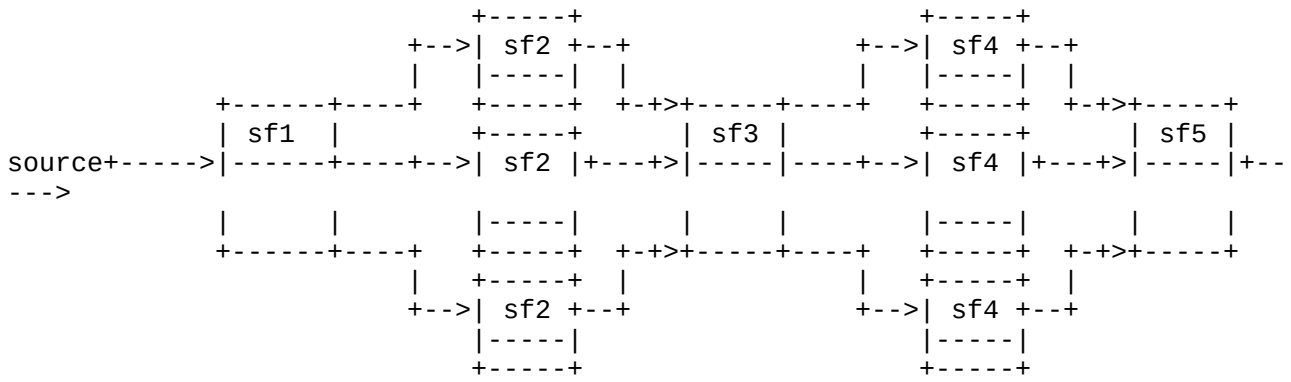


Figure 5: Load Balancing

Either through an imbedded action in sf1 and sf3, or through external control, the service functions sf2 and sf4 are elastically expanded and contracted dynamically. This would be represented as one chain: s1->s2->s3->s4->s5, but with multiple paths (not as a number of chains equal to the factorial combination of potential end-to-end paths). The load distribution decision will be localized (in general, although there might be macro policy controlling that - which is out of scope for the sake of a simple example). In this case, the control entity will push to the sf1 nodes, a table of sorts: sf2 with a series of next hops, and if needed some weighted or other metrics (these could also be decided locally by some policy, but sf1 would need to be aware of expand/contract triggers and actions). sf1 would use local logic -- hash, state table, etc. -- to distribute the chained packets to sf2.

The addition of high availability should likewise not require a multitude of new chains.

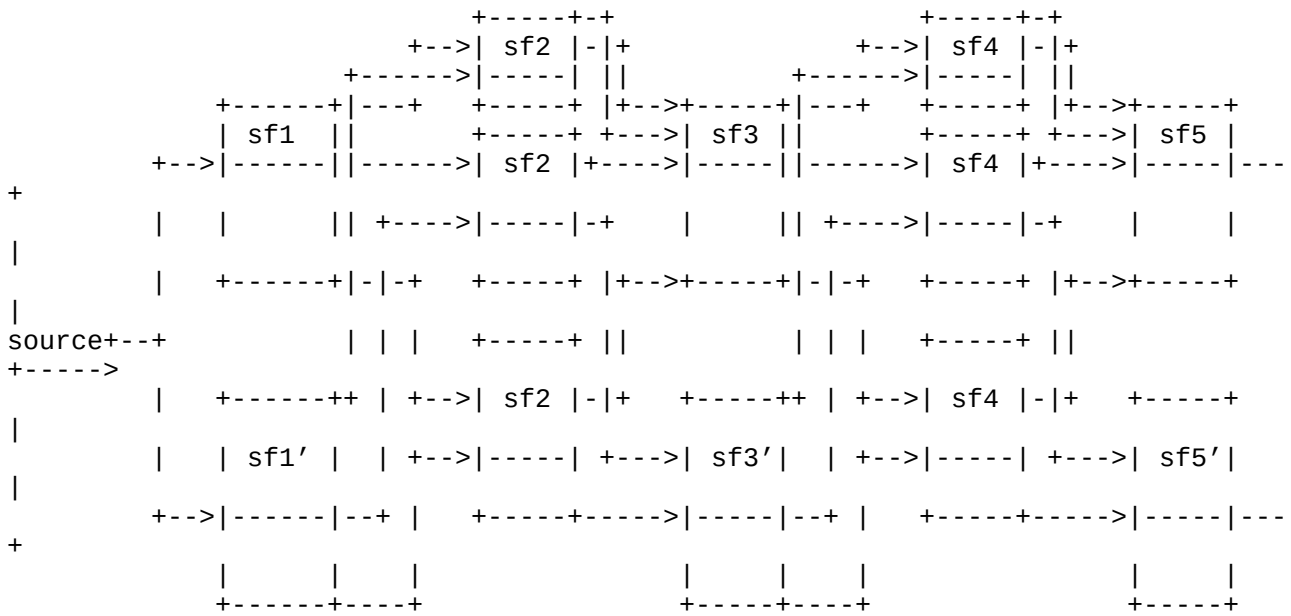


Figure 6: Load Balancing and HA

In the figure, sf1, sf3 and sf5 have a redundant counterpart for high availability purposes (typical of stateful appliance/function redundancy strategies, these entities may have private connections for transferring state not shown). Note that the elasticity of sf2 and sf4 provide a separate high availability strategy for those functions. In the case where sf1', sf3' and sf5' provide transparent dynamic replacement (they assert the addressing characteristics of their counterparts via an internal or external trigger), there is still a single chain (again, not a factorial explosion).

7. SFC Proxy

In order for the SFC architecture to support SFC-unaware SF's, an optional, logical SFC proxy function may be used. This proxy removes the SFC encapsulation and then uses a local attachment circuit to deliver packets to SFC unaware SFs. More specifically:

For traffic received from a NF or SFF, destined to an SF, the SFC proxy:

- Removes the SFC encapsulation from SFC encapsulated packets and/or frames.
- Identifies the required SF to be applied based on information carried in the SFC encapsulation.
- Selects the appropriate outbound local attachment circuit through which the next SF for this SFP is reachable. This information is derived from the SFC encapsulation or from local configuration. Examples of a local attachment circuit include, but are not limited to, VLANs, IP-in-IP, GRE, VXLAN.
- Forwards the original payload via a 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.
- Imposes the appropriate SFC encapsulation based on the identification of the SFC to be applied.

8. MTU Considerations

Modern systems are expected to be able to cope gracefully with MTU issues that may arise from the application of additional headers to a packet. Adopting the recommendations of other WG's who have recently tackled this issue (e.g. [RFC6830]), there are several mechanisms for dealing with packets that are too large to transit the path from the point of service classification to the last function (SFn) in the SFC.

In the "stateful" approach, the classifier keeps a per-path record of the maximum size allowed, and sends an ICMP Too Big message to the original source when a packet which is too large is seen (where "too large" implies after the imposition of the appropriate SFC encapsulation).

In the "stateless" approach, for IPv4, packets without the 'DF' bit set, too-large packets are fragmented, and then the fragments are forwarded; all other packets are discarded and an ICMP Too Big message returned.

A recommendation of a specific mechanism and/or its implementation is beyond the scope of this document.

9. 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. 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 run in-band fate-sharing with 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.

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

Some of the detailed functions performed by SFC OAM include fault detection, continuity checks, connectivity verification, service path tracing, diagnostic and fault isolation, alarm reporting, performance measurement, locking and testing of service functions, and also allow for vendor-specific as well as experimental functions. SFC should leverage, and if needed extend relevant existing OAM mechanisms.

10. Summary

Service function chains enable composite services that are constructed from one or more service functions. This document provides a standard architecture, including architectural concepts, principles, and components, for the creation of Service function chains.

11. Security Considerations

This document does not define a new protocol and therefore creates no new security issues.

12. Contributors

The following people are active contributors to this document and have provided review, content and concepts (listed alphabetically by surname):

Puneet Agarwal
Broadcom
Email: pagarwal@broadcom.com

Andre Beliveau
Ericsson
Email: andre.beliveau@ericsson.com

Kevin Glavin
Riverbed
Email: Kevin.Glavin@riverbed.com

Ken Gray
Cisco Systems, Inc.
Email: kegray@cisco.com

Jim Guichard
Cisco Systems, Inc.
Email: jguichar@cisco.com

Surendra Kumar
Cisco Systems, Inc.
Email: smkumar@cisco.com

Darrel Lewis
Cisco Systems, Inc.
Email: darlewis@cisco.com

Nic Leymann
Deutsche Telekom
Email: n.leymann@telekom.de

Rajeev Manur
Broadcom
Email: rmanur@broadcom.com

Thomas Nadeau
Brocade
Email: tnadeau@lucidvision.com

Carlos Pignataro
Cisco Systems, Inc.

Email: cpignata@cisco.com

Michael Smith
Cisco Systems, Inc.
Email: michsmit@cisco.com

Navindra Yadav
Cisco Systems, Inc.
Email: nyadav@cisco.com

13. Acknowledgments

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.

14. IANA Considerations

This document creates no new requirements on IANA namespaces [RFC5226].

15. References

15.1. Normative References

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

15.2. Informative References

[NSCprob] "Network Service Chaining Problem Statement", <http://datatracker.ietf.org/doc/draft-ietf-sfc-problem-statement/>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[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.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.

Appendix A. Existing Service Deployments

Existing service insertion and deployment techniques fail to address new challenging requirements raised by modern network architectures and evolving technologies such as multi-tenancy, virtualization, elasticity, and orchestration. Networks, servers, storage technologies, and applications, have all undergone significant change in recent years: virtualization, network overlays, and orchestration have increasingly become adopted techniques. All of these have profound effects on network and services design.

As network service functions evolve, operators are faced with an array of form factors - virtual and physical - as well as with a range of insertion methods that often vary by vendor and type of service.

Such existing services are deployed using a range of techniques, most often associated with topology or forwarding modifications. For example, firewalls often rely on layer-2 network changes for deployment: a VLAN is created for the "inside" interface, and another for the "outside" interface. In other words, a new L2 segment was created simply to add a service function. In the case of server load balancers, policy routing is often used to ensure traffic from server’s returns to the load balancer. As with the firewall example, the policy routing serves only to ensure that the network traffic ultimately flows to the service function(s).

The network-centric information (e.g. VLAN) is not limited to insertion; this information is often used as a policy identifier on the service itself. So, on a firewall, the layer-2 segment identifies the local policy to be selected. If more granular policy discrimination is required, more network identifiers must be created either per-hop, or communicated consistently to all services.

Appendix B. Issues with Existing Deployments

Due to the tight coupling of network and service function resources in existing networks, adding or removing service functions is a complex task that is fraught with risk and is tied to operationalizing topological changes leading to massively static configuration procedures for network service delivery or update purposes. The inflexibility of such deployments limits (and in many cases precludes) dynamic service scaling (both horizontal and vertical) and requires hop-by-hop configuration to ensure that the correct service functions, and sequence of service functions are traversed.

A non-exhaustive list of existing service deployment and insertion techniques as well as the issues associated with each may be found in [NSCprob].

Appendix C. SFC Encapsulation Requirements

TBD

Authors' Addresses

Paul Quinn (editor)
Cisco Systems, Inc.

Email: paulq@cisco.com

Joel Halpern (editor)
Ericsson

Email: jmh@joelhalpern.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 28, 2015

P. Quinn
J. Guichard
S. Kumar
M. Smith
Cisco Systems, Inc.
W. Henderickx
Alcatel-Lucent
T. Nadeau
Brocade
P. Agarwal

R. Manur
Broadcom
A. Chauhan
Citrix
J. Halpern
Ericsson
S. Majee
F5
U. Elzur
Intel
D. Melman
Marvell
P. Garg
Microsoft
B. McConnell
Rackspace
C. Wright
Red Hat Inc.
K. Glavin
Riverbed
C. Zhang
L. Fourie
Huawei US R&D
R. Parker
Affirmed Networks
M. Zarny
Goldman Sachs
February 24, 2015

Network Service Header
draft-quinn-sfc-nsh-07.txt

Abstract

This draft describes a Network Service Header (NSH) inserted onto encapsulated packets or frames to realize service function paths.

NSH also provides a mechanism for metadata exchange along the instantiated service path.

1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of this Memo

Copyright Notice

Table of Contents

2. Introduction

Service functions are widely deployed and essential in many networks. These service functions provide a range of features such as security, WAN acceleration, and server load balancing. Service functions may be instantiated at different points in the network infrastructure such as the wide area network, data center, campus, and so forth.

The current service function deployment models are relatively static, and bound to topology for insertion and policy selection. Furthermore, they do not adapt well to elastic service environments enabled by virtualization.

New data center network and cloud architectures require more flexible service function deployment models. Additionally, the transition to virtual platforms requires an agile service insertion model that supports elastic service delivery; the movement of service functions and application workloads in the network and the ability to easily bind service policy to granular information such as per-subscriber state are necessary.

The approach taken by NSH is composed of the following elements:

1. Service path identification

2. Transport independent per-packet/frame service metadata.

3. Optional variable TLV metadata.

NSH is designed to be easy to implement across a range of devices, both physical and virtual, including hardware platforms.

An NSH aware control plane is outside the scope of this document.

The SFC Architecture document [SFC-arch] provides an overview of a service chaining architecture that clearly defines the roles of the various elements and the scope of a service function chaining encapsulation.

2.1. Definition of Terms

Classification: Locally instantiated policy and customer/network/service profile matching of traffic flows for identification of appropriate outbound forwarding actions.

SFC Network Forwarder (NF): SFC network forwarders provide network connectivity for service functions forwarders and service functions. SFC network forwarders participate in the network overlay used for service function chaining as well as in the SFC encapsulation.

Service Function Forwarder (SFF): A service function forwarder is responsible for delivering traffic received from the NF to one or more connected service functions, and from service functions to the NF.

Service Function (SF): A function that is responsible for specific treatment of received packets. A service function can act at the network layer or other OSI layers. A service function can be a virtual instance or be embedded in a physical network element. One of multiple service functions can be embedded in the same network element. Multiple instances of the service function can be enabled in the same administrative domain.

Service Node (SN): Physical or virtual element that hosts one or more service functions and has one or more network locators associated with it for reachability and service delivery.

Service Function Chain (SFC): A service function chain defines an ordered set of service functions that must be applied to packets and/or frames selected as a result of classification. The implied order may not be a linear progression as the architecture allows for nodes that copy to more than one branch. The term service chain is often used as shorthand for service function chain.

Service Function Path (SFP): The instantiation of a SFC in the network. Packets follow a service function path from a classifier through the requisite service functions

Network Node/Element: Device that forwards packets or frames based on outer header information. In most cases is not aware of the presence of NSH.

Network Overlay: Logical network built on top of existing network (the underlay). Packets are encapsulated or tunneled to create the overlay network topology.

Network Service Header: Data plane header added to frames/packets. The header contains information required for service chaining, as well as metadata added and consumed by network nodes and service elements.

Service Classifier: Function that performs classification and imposes an NSH. Creates a service path. Non-initial (i.e. subsequent) classification can occur as needed and can alter, or create a new service path.

Service Hop: NSH aware node, akin to an IP hop but in the service overlay.

Service Path Segment: A segment of a service path overlay.

NSH Proxy: Acts as a gateway: removes and inserts NSH on behalf of a service function that is not NSH aware.

2.2. Problem Space

Network Service Header (NSH) addresses several limitations associated with service function deployments today.

1. Topological Dependencies: Network service deployments are often coupled to network topology. Such dependency imposes constraints on the service delivery, potentially inhibiting the network operator from optimally utilizing service resources, and reduces the flexibility. This limits scale, capacity, and redundancy across network resources.

2. Service Chain Construction: Service function chains today are most typically built through manual configuration processes. These are slow and error prone. With the advent of newer service deployment models the control/management planes provide not only connectivity state, but will also be increasingly utilized for the creation of network services. Such a control/management planes could be centralized, or be distributed.

3. Application of Service Policy: Service functions rely on topology information such as VLANs or packet (re) classification to determine service policy selection, i.e. the service function specific action taken. Topology information is increasingly less viable due to scaling, tenancy and complexity reasons. The topological information is often stale, providing the operator with inaccurate placement that can result in suboptimal resource utilization. Furthermore topology-centric information often does not convey adequate information to the service functions, forcing functions to individually perform more granular classification.

4. Per-Service (re)Classification: Classification occurs at each service function independent from previously applied service functions. More importantly, the classification functionality often differs per service function and service functions may not

leverage the results from other service functions.

5. Common Header Format: Various proprietary methods are used to share metadata and create service paths. An open header provides a common format for all network and service devices.

6. Limited End-to-End Service Visibility: Troubleshooting service related issues is a complex process that involve both network-specific and service-specific expertise. This is especially the case when service function chains span multiple DCs, or across administrative boundaries. Furthermore, the physical and virtual environments (network and service) can be highly divergent in terms of topology and that topological variance adds to these challenges.

7. Transport Dependence: Service functions can and will be deployed in networks with a range of transports requiring service functions to support and participate in many transports (and associated control planes) or for a transport gateway function to be present.

Please see the Service Function Chaining Problem Statement [SFC-PS] for a more detailed analysis of service function deployment problem areas.

3. Network Service Header

A Network Service Header (NSH) contains metadata and service path information that are added to a packet or frame and used to create a service plane. The packets and the NSH are then encapsulated in an outer header for transport.

The service header is added by a service classification function - a device or application - that determines which packets require servicing, and correspondingly which service path to follow to apply the appropriate service.

3.1. Network Service Header Format

An NSH is composed of a 4-byte base header, a 4-byte service path header and context headers, as shown in Figure 1 below.

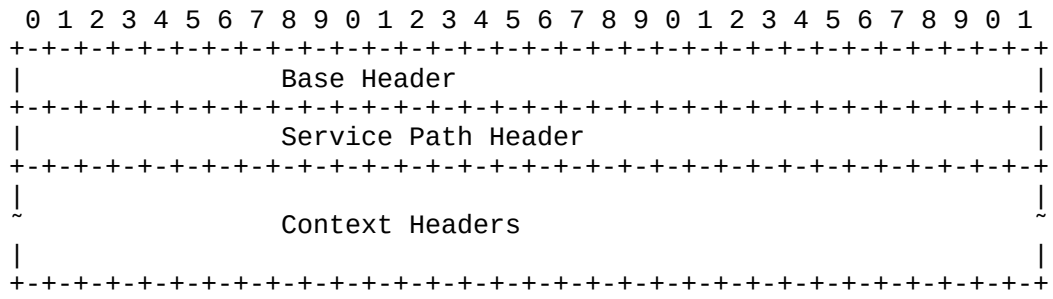


Figure 1: Network Service Header

Base header: provides information about the service header and the payload protocol.

Service Path Header: provide path identification and location within a path.

Context headers: carry opaque metadata and variable length encoded information.

3.2. NSH Base Header

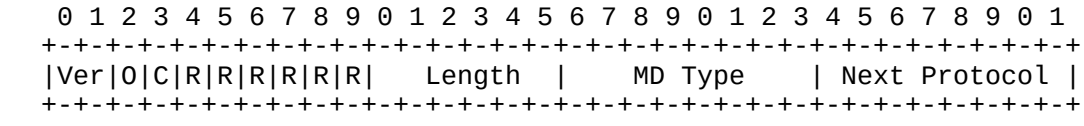


Figure 2: NSH Base Header

Base Header Field Descriptions

Version: The version field is used to ensure backward compatibility going forward with future NSH updates.

O bit: Indicates that this packet is an operations and management (OAM) packet. SFF and SFs nodes MUST examine the payload and take appropriate action (e.g. return status information).

OAM message specifics and handling details are outside the scope of this document.

C bit: Indicates that a critical metadata TLV is present (see Section 3.4.2). This bit acts as an indication for hardware implementers to decide how to handle the presence of a critical TLV without necessarily needing to parse all TLVs present. The C bit MUST be set to 1 if one or more critical TLVs are present.

All other flag fields are reserved.

Length: total length, in 4-byte words, of the NSH header, including optional variable TLVs.

MD Type: indicates the format of NSH beyond the base header and the type of metadata being carried. This typing is used to describe the use for the metadata. A new registry will be requested from IANA for the MD Type.

NSH defines two MD types:

0x1 which indicates that the format of the header includes fixed length context headers.

0x2 which does not mandate any headers beyond the base header and service path header, and may contain optional variable length context information.

The format of the base header is invariant, and not described by MD Type.

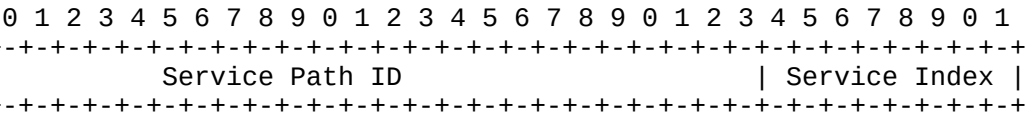
NSH implementations MUST support MD-Type 0x1, and SHOULD support MD-Type 0x2.

Next Protocol: indicates the protocol type of the original packet. A new IANA registry will be created for protocol type.

This draft defines the following Next Protocol values:

0x1 : IPv4
0x2 : IPv6
0x3 : Ethernet

3.3. Service Path Header



Service path ID (SPI): 24 bits
Service index (SI): 8 bits

Figure 3: NSH Service Path Header

Service Path Identifier (SPI): identifies a service path. Participating nodes MUST use this identifier for path selection. An administrator can use the service path value for reporting and troubleshooting packets along a specific path.

Service Index (SI): provides location within the service path. Service index MUST be decremented by service functions or proxy nodes after performing required services. MAY be used in conjunction with service path for path selection. Service Index is also valuable when troubleshooting/reporting service paths. In addition to location within a path, SI can be used for loop detection.

3.4. NSH MD-type 1

When the base header specifies MD Type 1, NSH defines four 4-byte mandatory context headers, as per Figure 4. These headers must be present and the format is opaque as depicted in Figure 5.

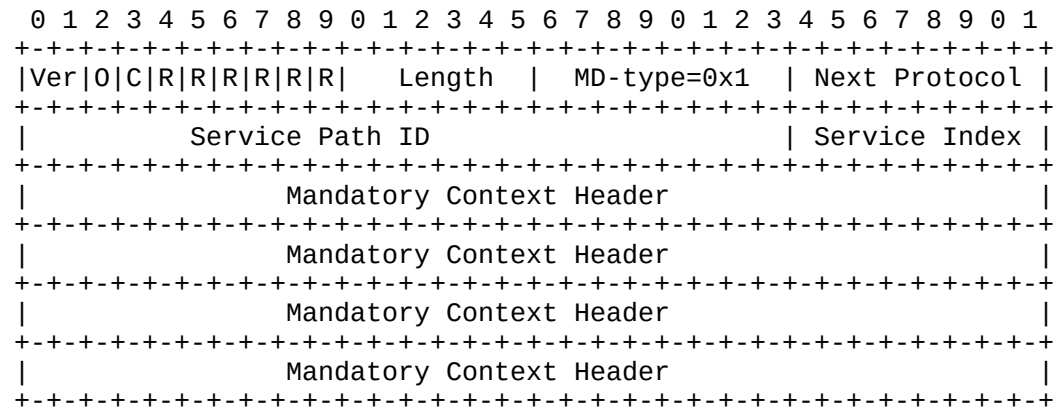


Figure 4: NSH MD-type=0x1

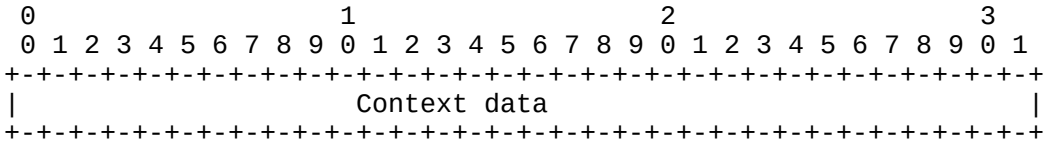


Figure 5: Context Header

3.4.1. Mandatory Context Header Allocation Guidelines

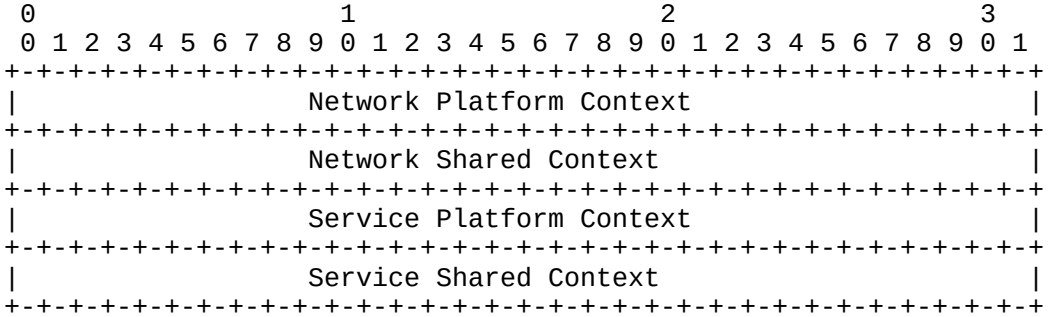


Figure 6: Context Data Significance

Figure 6, above, and the following examples of context header allocation are guidelines that illustrate how various forms of information can be carried and exchanged via NSH.

Network platform context: provides platform-specific metadata shared between network nodes. Examples include (but are not limited to) ingress port information, forwarding context and encapsulation type.

Network shared context: metadata relevant to any network node such as the result of edge classification. For example, application information, identity information or tenancy information can be shared using this context header.

Service platform context: provides service platform specific metadata shared between service functions. This context header is analogous to the network platform context, enabling service platforms to exchange platform-centric information such as an identifier used for load balancing decisions.

Service shared context: metadata relevant to, and shared, between service functions. As with the shared network context, classification information such as application type can be conveyed using this context.

The data center[dcalloc] and mobility[moballoc] context header allocation drafts provide guidelines for the semantics of NSH fixed context headers in each respective environment.

3.5. NSH MD-type 2

When the base header specifies MD Type 2, NSH defines variable length only context headers. There may be zero or more of these headers as per the length field.

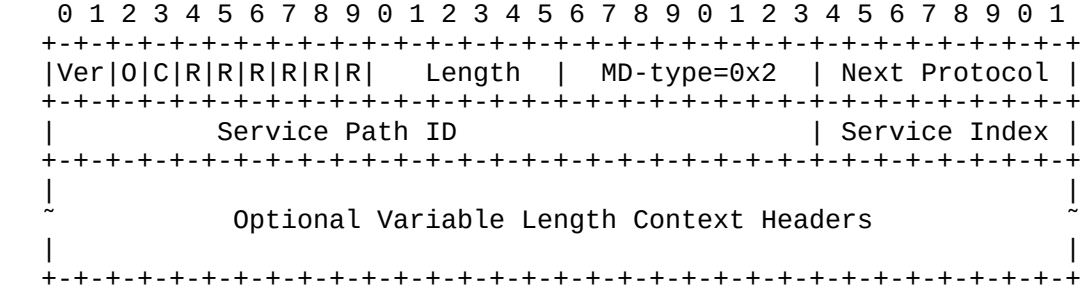


Figure 7: NSH MD-type=0x2

3.5.1. Optional Variable Length Metadata

NSH MD Type 2 MAY contain optional variable length context headers. The format of these headers is as described below.

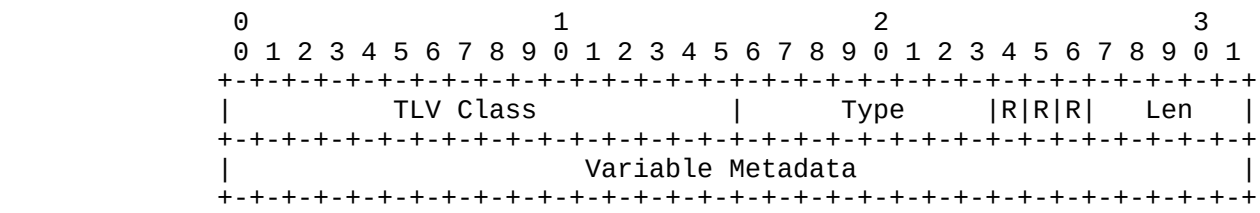


Figure 8: Variable Context Headers

TLV Class: describes the scope of the "Type" field. In some cases, the TLV Class will identify a specific vendor, in others, the TLV Class will identify specific standards body allocated types.

Type: the specific type of information being carried, within the scope of a given TLV Class. Value allocation is the responsibility of the TLV Class owner.

The most significant bit of the Type field indicates whether the TLV is mandatory for the receiver to understand/process. This effectively allocates Type values 0 to 127 for non-critical options and Type values 128 to 255 for critical options. Figure 7 below illustrates the placement of the Critical bit within the Type field.

Figure 9: Critical Bit Placement Within the TLV Type Field

Encoding the criticality of the TLV within the Type field is consistent with IPv6 option types.

If a receiver receives an encapsulated packet containing a TLV with the Critical bit set in the Type field and it does not understand how to process the Type, it MUST drop the packet. Transit devices MUST NOT drop packets based on the setting of this bit.

Reserved bits: three reserved bit are present for future use. The

reserved bits MUST be zero.

Length: Length of the variable metadata, in 4-byte words.

4. NSH Actions

Service header aware nodes - service classifiers, SFF, SF and NSH proxies, have several possible header related actions:

1. Insert or remove service header: These actions can occur at the start and end respectively of a service path. Packets are classified, and if determined to require servicing, a service header imposed. The last node in a service path, an SFF, removes the NSH. A service classifier MUST insert an NSH. At the end of a service function chain, the last node operating on the service header MUST remove it.

   A service function can re-classify data as required and that re-classification might result in a new service path. In this case, the SF acts as a logical classifier as well. When the logical classifier performs re-classification that results in a change of service path, it MUST remove the existing NSH and MUST impose a new NSH with the base header reflecting the new path.

2. Select service path: The base header provides service chain information and is used by SFFs to determine correct service path selection. SFFs MUST use the base header for selecting the next service in the service path.

3. Update a service header: NSH aware service functions MUST decrement the service index. A service index = 0 indicates that a packet MUST be dropped by the SFF performing NSH-based forwarding.

   Service functions MAY update context headers if new/updated context is available.

   If an NSH proxy (see Section 7) is in use (acting on behalf of a non-NSH-aware service function for NSH actions), then the proxy MUST update service index and MAY update contexts. When an NSH proxy receives an NSH-encapsulated packet, it removes the NSH before forwarding it to an NSH unaware SF. When it receives a packet back from an NSH unaware SF, it re-encapsulates it with the NSH, decrementing the service index.

4. Service policy selection: Service function instances derive policy selection from the service header. Context shared in the service header can provide a range of service-relevant information such as traffic classification. Service functions SHOULD use NSH to select local service policy.

Figure 10 maps each of the four actions above to the components in the SFC architecture that can perform it.

Component	Insert or remove service header: Insert	Remove	Remove and Insert	Select service path	Update a service header: Dec. Service Index	Update Context Header	Service Policy Selection
Service Classification Function	+					+	
Service Function Forwarder(SFF)		+		+		+	
Service Function (SF)					+	+	+
NSH Proxy	+	+			+	+	

Figure 10: NSH Action and Role Mapping

5. NSH Encapsulation

Once NSH is added to a packet, an outer encapsulation is used to forward the original packet and the associated metadata to the start of a service chain. The encapsulation serves two purposes:

1. Creates a topologically independent services plane. Packets are forwarded to the required services without changing the underlying network topology.

2. Transit network nodes simply forward the encapsulated packets as is.

The service header is independent of the encapsulation used and is encapsulated in existing transports. The presence of NSH is indicated via protocol type or other indicator in the outer encapsulation.

See Section 11 for NSH encapsulation examples.

6. NSH Usage

The NSH creates a dedicated service plane, that addresses many of the limitations highlighted in Section 2.2. More specifically, NSH enables:

1. Topological Independence: Service forwarding occurs within the service plane, via a network overlay, the underlying network topology does not require modification. Service functions have one or more network locators (e.g. IP address) to receive/send data within the service plane, the NSH contains an identifier that is used to uniquely identify a service path and the services within that path.

2. Service Chaining: NSH contains path identification information needed to realize a service path. Furthermore, NSH provides the ability to monitor and troubleshoot a service chain, end-to-end via service-specific OAM messages. The NSH fields can be used by administrators (via, for example a traffic analyzer) to verify (account, ensure correct chaining, provide reports, etc.) the path specifics of packets being forwarded along a service path.

3. Metadata Sharing: NSH provides a mechanism to carry shared metadata between network devices and service function, and between service functions. The semantics of the shared metadata is communicated via a control plane to participating nodes. Examples of metadata include classification information used for policy enforcement and network context for forwarding post service delivery.

4. Transport Agnostic: NSH is transport independent and is carried in an overlay, over existing underlays. If an existing overlay topology provides the required service path connectivity, that existing overlay may be used.

7. NSH Proxy Nodes

In order to support NSH-unaware service functions, an NSH proxy is used. The proxy node removes the NSH header and delivers the original packet/frame via a local attachment circuit to the service function. Examples of a local attachment circuit include, but are not limited to: VLANs, IP in IP, GRE, VXLAN. When complete, the service function returns the packet to the NSH proxy via the same or different attachment circuit.

NSH is re-imposed on packets returned to the proxy from the non-NSH-aware service.

Typically, an SFF will act as an NSH-proxy when required.

An NSH proxy MUST perform NSH actions as described in Section 4.

8. Fragmentation Considerations

Work in progress

9. Service Path Forwarding with NSH

9.1. SFFs and Overlay Selection

As described above, NSH contains a service path identifier (SPI) and a service index (SI). The SPI is, as per its name, an identifier. The SPI alone cannot be used to forward packets along a service path. Rather the SPI provide a level of indirection between the service path/topology and the network transport. Furthermore, there is no requirement, or expectation of an SPI being bound to a pre-determined or static network path.

The service index provides an indication of location within a service path. The combination of SPI and SI provides the identification and location of a logical SF (locator and order). The logical SF may be a single SF, or a set of SFs that are equivalent. In the latter case, the SFF provides load distribution amongst the collection of SFs as needed. SI may also serve as a mechanism for loop detection with in a service path since each SF in the path decrements the index; an index of 0 indicates that a loop occurred and packet must be discarded.

This indirection -- path ID to overlay -- creates a true service plane. That is the SFF/SF topology is constructed without impacting the network topology but more importantly service plane only participants (i.e. most SFs) need not be part of the network overlay topology and its associated infrastructure (e.g. control plane, routing tables, etc.). As mentioned above, an existing overlay topology may be used provided it offers the requisite connectivity.

The mapping of SPI to transport occurs on an SFF. The SFF consults the SPI/ID values to determine the appropriate overlay transport protocol (several may be used within a given network) and next hop for the requisite SF. Figure 10 below depicts an SPI/SI to network overlay mapping.

SPI	SI	NH	Transport
10	3	1.1.1.1	VXLAN-gpe
10	2	2.2.2.2	nvGRE
245	12	192.168.45.3	VXLAN-gpe
10	9	10.1.2.3	GRE
40	9	10.1.2.3	GRE
50	7	01:23:45:67:89:ab	Ethernet
15	1	Null (end of path)	None

Figure 11: SFF NSH Mapping Example

Additionally, further indirection is possible: the resolution of the required SF function locator may be a localized resolution on an SFF,rather than a service function chain control plane responsibility, as per figures 11 and 12 below.

SPI	SI	NH
10	3	SF2
245	12	SF34
40	9	SF9

Figure 12: NSH to SF Mapping Example

SF	NH	Transport
SF2	10.1.1.1	VXLAN-gpe
SF34	192.168.1.1	UDP
SF9	1.1.1.1	GRE

Figure 13: SF Locator Mapping Example

Since the SPI is a representation of the service path, the lookup may return more than one possible next-hop within a service path for a given SF, essentially a series of weighted (equally or otherwise) overlay links to be used (for load distribution, redundancy or policy), see Figure 13. The metric depicted in Figure 13 is an example to help illustrated weighing SFs. In a real network, the metric will range from a simple preference (similar to routing next-hop), to a true dynamic composite metric based on some service function-centric state (including load, sessions sate, capacity, etc.)

```
+----------------------------------+
| SPI | SI |   NH        |  Metric |
+----------------------------------+
| 10  |  3 | 10.1.1.1    |  1      |
|     |    | 10.1.1.2    |  1      |
|     |    |             |         |
| 20  | 12 | 192.168.1.1 |  1      |
|     |    | 10.2.2.2    |  1      |
|     |    |             |         |
| 30  |  7 | 10.2.2.3    |  10     |
|     |    | 10.3.3.3    |  5      |
+----------------------------------+
(encap type omitted for formatting)
```

Figure 14: NSH Weighted Service Path

9.2. Mapping NSH to Network Overlay

As described above, the mapping of SPI to network topology may result in a single overlay path, or it might result in a more complex topology. Furthermore, the SPIx to overlay mapping occurs at each SFF independently. Any combination of topology selection is possible.

Examples of mapping for a topology:

1. Next SF is located at SFFb with locator 10.1.1.1
   SFFa mapping: SPI=10 --> VXLAN-gpe, dst-ip: 10.1.1.1

2. Next SF is located at SFFc with multiple locator for load distribution purposes:
   SFFb mapping: SPI=10 --> VXLAN-gpe, dst_ip:10.2.2.1, 10.2.2.2, 10.2.2.3, equal cost

3. Next SF is located at SFFd with two path to SFFc, one for redundancy:
   SFFc mapping: SPI=10 --> VXLAN-gpe, dst_ip:10.1.1.1 cost=10, 10.1.1.2, cost=20

In the above example, each SFF makes an independent decision about the network overlay path and policy for that path. In other words, there is no a priori mandate about how to forward packets in the network (only the order of services that must be traversed).

The network operator retains the ability to engineer the overlay paths as required. For example, the overlay path between service functions forwarders may utilize traffic engineering, QoS marking, or

ECMP, without requiring complex configuration and network protocol support to be extended to the service path explicitly. In other words, the network operates as expected, and evolves as required, as does the service function plane.

9.3. Service Plane Visibility

The SPI and SI serve an important function for visibility into the service topology. An operator can determine what service path a packet is "on", and its location within that path simply by viewing the NSH information (packet capture, IPFIX, etc.). The information can be used for service scheduling and placement decisions, troubleshooting and compliance verification.

9.4. Service Graphs

In some cases, a service path is exactly that -- a linear list of service functions that must be traversed. However, increasingly, the "path" is actually a true directed graph. Furthermore, within a given service topology several directed graphs may exist with packets moving between graphs based on non-initial classification (usually performed by a service function). Note: strictly speaking a path is a form of graph; the intent is to distinguish between a directed graph and a path.

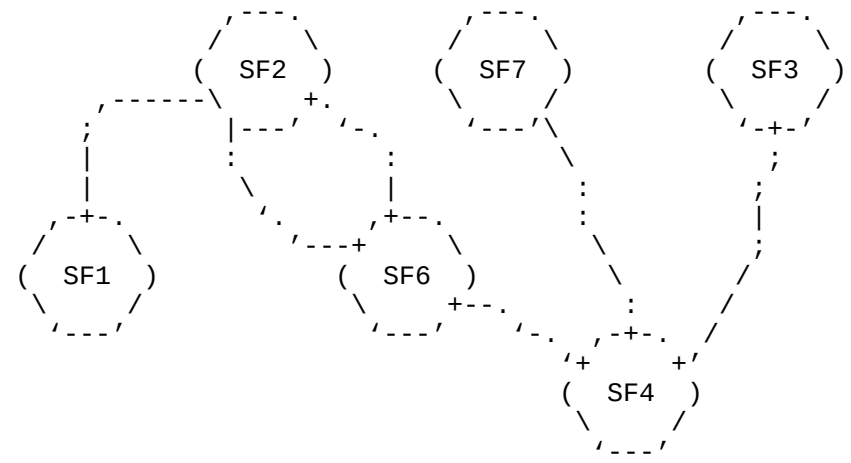


Figure 15: Service Graph Example

The SPI/SI combination provides a simple representation of a directed graph, the SPI represents a graph ID; and the SI a node ID. The

service topology formed by SPI/SI support cycles, weighting, and alternate topology selection, all within the service plane. The realization of the network topology occurs as described above: SPI/ID mapping to an appropriate transport and associated next network hops.

NSH-aware services receive the entire header, including the SPI/SI. An SF can now, based on local policy, alter the SPI, which in turn effects both the service graph, and in turn the selection of overlay at the SFF. The figure below depicts the policy associated with the graph in Figure 14 above. Note: this illustrates multiple graphs and their representation; it does not depict the use of metadata within a single service function graph.

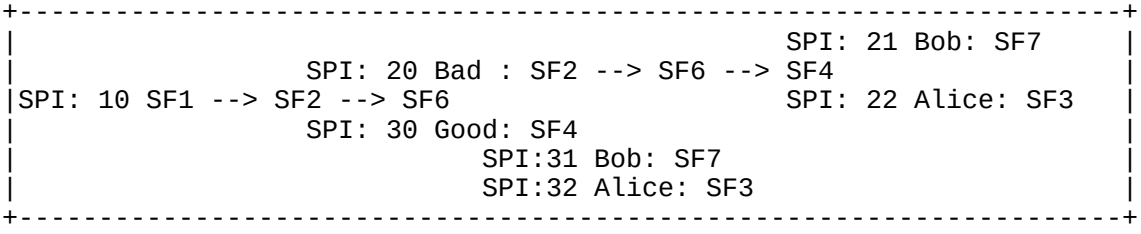


Figure 16: Service Graphs Using SPI

This example above does not show the mapping of the service topology to the network overlay topology. As discussed in the sections above, the overlay selection occurs as per network policy.

10. Policy Enforcement with NSH

10.1. NSH Metadata and Policy Enforcement

As described in Section 3, NSH provides the ability to carry metadata along a service path. This metadata may be derived from several sources, common examples include:

Network nodes: Information provided by network nodes can indicate network-centric information (such as VRF or tenant) that may be used by service functions, or conveyed to another network node post-service pathing.

External (to the network) systems: External systems, such as orchestration systems, often contain information that is valuable for service function policy decisions. In most cases, this information cannot be deduced by network nodes. For example, a cloud orchestration platform placing workloads "knows" what application is being instantiated and can communicate this information to all NSH nodes via metadata.

Service functions: Service functions often perform very detailed and valuable classification. In some cases they may terminate, and be able to inspect encrypted traffic. SFs may update, alter or impose metadata information.

Regardless of the source, metadata reflects the "result" of classification. The granularity of classification may vary. For example, a network switch might only be able to classify based on a 5-tuple, whereas, a service function may be able to inspect application information. Regardless of granularity, the classification information can be represented in NSH.

Once the data is added to NSH, it is carried along the service path, NSH-aware SFs receive the metadata, and can use that metadata for local decisions and policy enforcement. The following two examples highlight the relationship between metadata and policy:

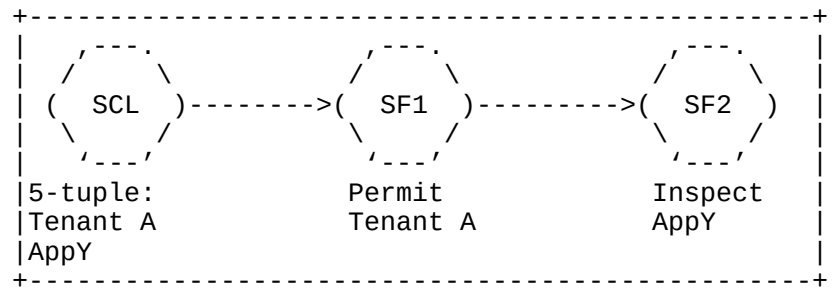


Figure 17: Metadata and Policy

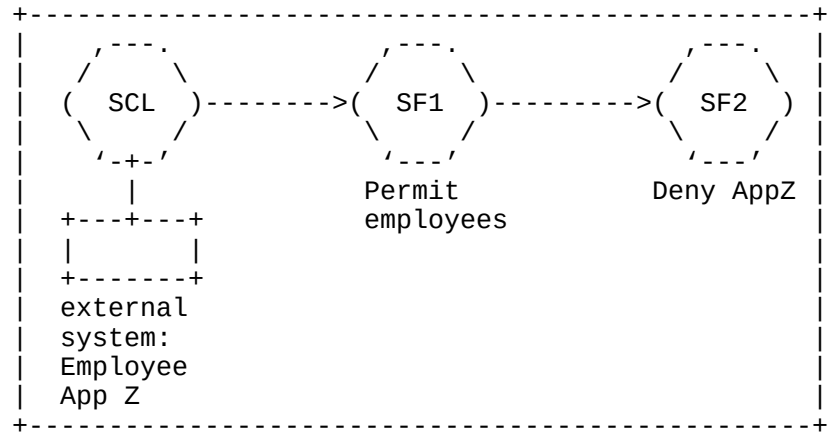


Figure 18: External Metadata and Policy

In both of the examples above, the service functions perform policy decisions based on the result of the initial classification: the SFs did not need to perform re-classification, rather they relied on a antecedent classification for local policy enforcement.

10.2. Updating/Augmenting Metadata

Post-initial metadata imposition (typically performed during initial service path determination), metadata may be augmented or updated:

1. Metadata Augmentation: Information may be added to NSH's existing metadata, as depicted in Figure 18. For example, if the initial classification returns the tenant information, a secondary classification (perhaps a DPI or SLB) may augment the tenant classification with application information. The tenant

classification is still valid and present, but additional information has been added to it.

2. Metadata Update: Subsequent classifiers may update the initial classification if it is determined to be incorrect or not descriptive enough. For example, the initial classifier adds metadata that describes the trafic as "internet" but a security service function determines that the traffic is really "attack". Figure 19 illustrates an example of updating metadata.

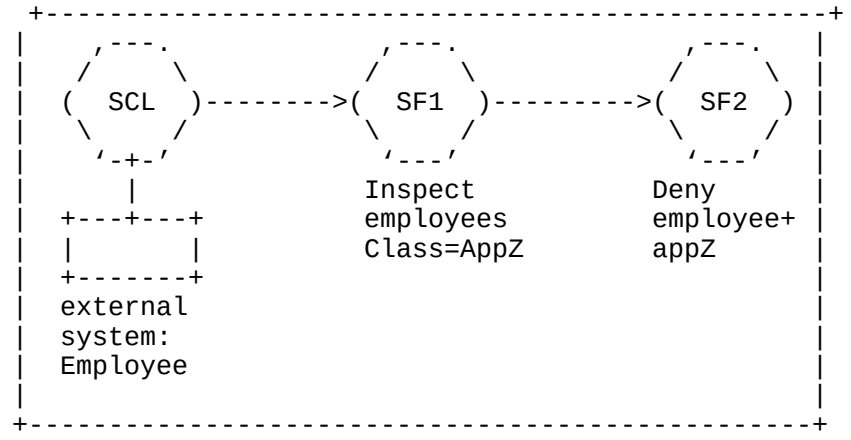


Figure 19: Metadata Augmentation

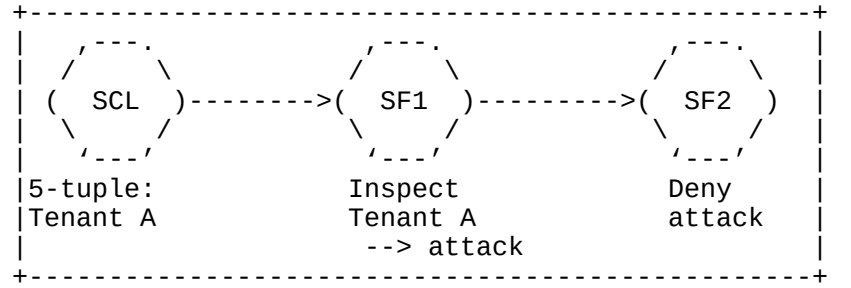


Figure 20: Metadata Update

10.3. Service Path ID and Metadata

Metadata information may influence the service path selection since the service path identifier can represent the result of classification. A given SPI can represent all or some of the metadata, and be updated based on metadata classification results. This relationship provides the ability to create a dynamic services plane based on complex classification without requiring each node to be capable of such classification, or requiring a coupling to the network topology. This yields service graph functionality as described in Section 9.4. Figure 20 illustrates an example of this behavior.

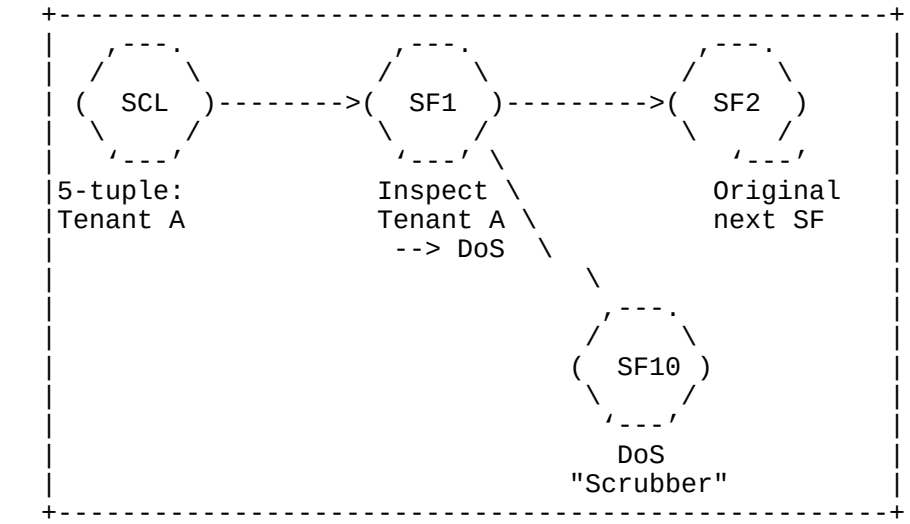


Figure 21: Path ID and Metadata

Specific algorithms for mapping metadata to an SPI are outside the scope of this draft.

11. NSH Encapsulation Examples

11.1. GRE + NSH

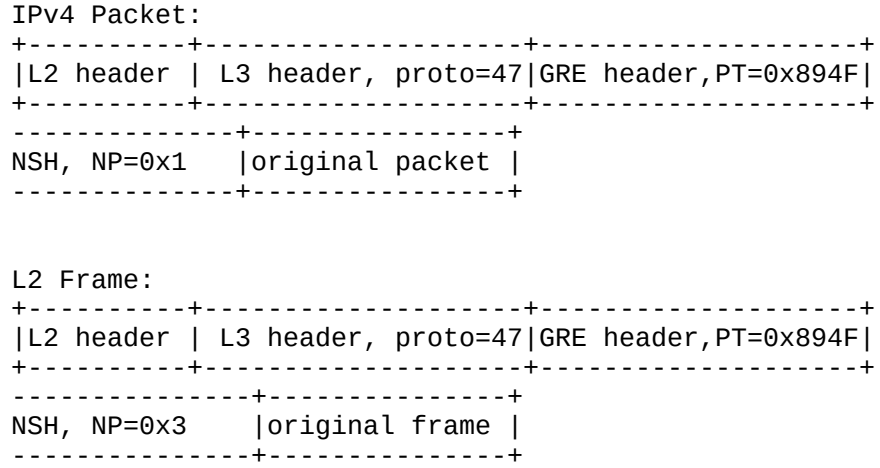


Figure 22: GRE + NSH

11.2. VXLAN-gpe + NSH

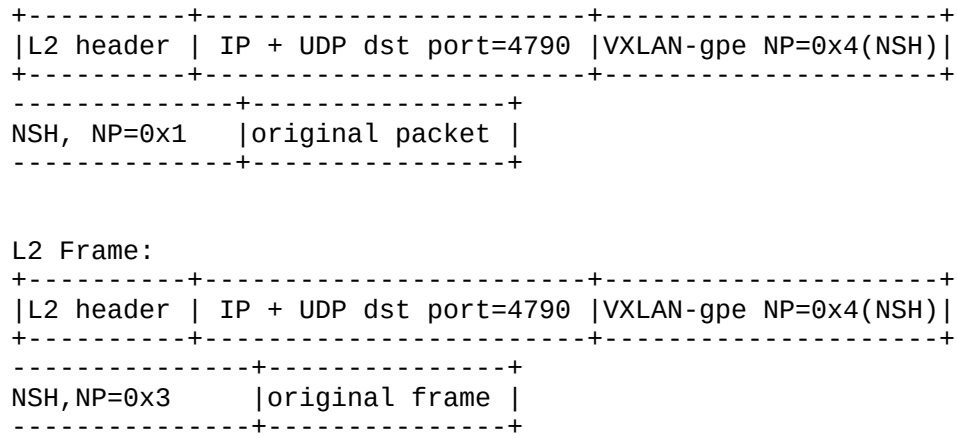


Figure 23: VXLAN-gpe + NSH

11.3. Ethernet + NSH

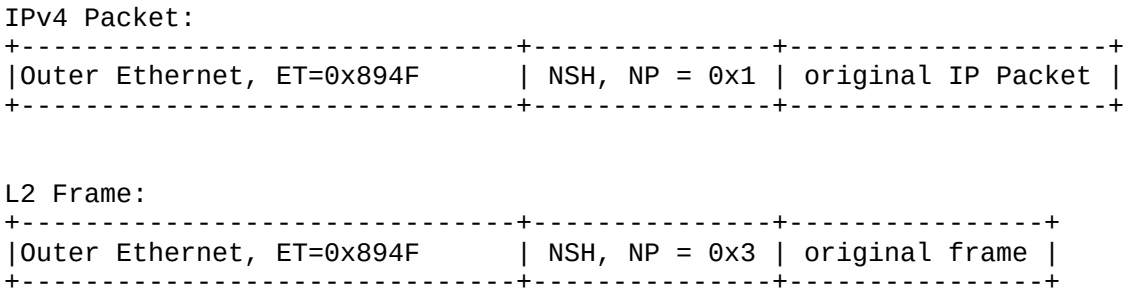


Figure 24: Ethernet + NSH

12. Security Considerations

As with many other protocols, NSH data can be spoofed or otherwise modified. In many deployments, NSH will be used in a controlled environment, with trusted devices (e.g. a data center) thus mitigating the risk of unauthorized header manipulation.

NSH is always encapsulated in a transport protocol and therefore, when required, existing security protocols that provide authenticity (e.g. RFC 2119 [RFC6071]) can be used.

Similarly if confidentiality is required, existing encryption protocols can be used in conjunction with encapsulated NSH.

13. Open Items for WG Discussion

1. MD type 1 metadata semantics specifics
2. Bypass bit in NSH.
3. Rendered Service Path ID (RSPID).

14. Contributors

The following people are active contributors to this document and have provided review, content and concepts (listed alphabetically by surname):

Andrew Dolganow
Alcaltel-Lucent
Email: andrew.dolganow@alcatel-lucent.com

Rex Fernando
Cisco Systems
Email: rex@cisco.com

Praveen Muley
Alcaltel-Lucent
Email: praveen.muley@alcatel-lucent.com

Navindra Yadav
Cisco Systems
Email: nyadav@cisco.com

15. Acknowledgments

The authors would like to thank Nagaraj Bagepalli, Abhijit Patra, Ron Parker, Peter Bosch, Darrel Lewis, Pritesh Kothari, Tal Mizrahi and Ken Gray for their detailed review, comments and contributions.

A special thank you goes to David Ward and Tom Edsall for their guidance and feedback.

Additionally the authors would like to thank Carlos Pignataro and Larry Kreeger for their invaluable ideas and contributions which are reflected throughout this draft.

16. IANA Considerations

16.1. NSH EtherType

An IEEE EtherType, 0x894F, has been allocated for NSH.

16.2. Network Service Header (NSH) Parameters

IANA is requested to create a new "Network Service Header (NSH) Parameters" registry. The following sub-sections request new registries within the "Network Service Header (NSH) Parameters " registry.

16.2.1. NSH Base Header Reserved Bits

There are ten bits at the beginning of the NSH Base Header. New bits are assigned via Standards Action [RFC5226].

Bits 0-1 - Version
Bit 2 - OAM (O bit)
Bits 2-9 - Reserved

16.2.2. MD Type Registry

IANA is requested to set up a registry of "MD Types". These are 8-bit values. MD Type values 0, 1, 2, 254, and 255 are specified in this document. Registry entries are assigned by using the "IETF Review" policy defined in RFC 5226 [RFC5226].

MD Type	Description	Reference
0	Reserved	This document
1	NSH	This document
2	NSH	This document
3..253	Unassigned	
254	Experiment 1	This document
255	Experiment 2	This document

Table 1

16.2.3. TLV Class Registry

IANA is requested to set up a registry of "TLV Types". These are 16-bit values. Registry entries are assigned by using the "IETF Review" policy defined in RFC 5226 [RFC5226].

16.2.4. NSH Base Header Next Protocol

IANA is requested to set up a registry of "Next Protocol". These are 8-bit values. Next Protocol values 0, 1, 2 and 3 are defined in this draft. New values are assigned via Standards Action [RFC5226].

Next Protocol	Description	Reference
0	Reserved	This document
1	IPv4	This document
2	IPv6	This document
3	Ethernet	This document
4..253	Unassigned	

Table 2

17. References

17.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

17.2. Informative References

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC6071] Frankel, S. and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap", RFC 6071, February 2011.

[SFC-PS] Quinn, P., Ed. and T. Nadeau, Ed., "Service Function Chaining Problem Statement", 2014, <http://datatracker.ietf.org/doc/draft-ietf-sfc-problem-statement/>.

[SFC-arch] Quinn, P., Ed. and J. Halpern, Ed., "Service Function Chaining (SFC) Architecture", 2014, <http://datatracker.ietf.org/doc/draft-quinn-sfc-arch>.

[VXLAN-gpe] Quinn, P., Agarwal, P., Kreeger, L., Lewis, D., Maino, F., Yong, L., Xu, X., Elzur, U., and P. Garg, "Generic Protocol Extension for VXLAN", <https://datatracker.ietf.org/doc/draft-quinn-vxlan-gpe/>.

[dcalloc] Guichard, J., Smith, M., and S. Kumar, "Network Service Header (NSH) Context Header Allocation (Data Center)", 2014, <https://datatracker.ietf.org/doc/draft-guichard-sfc-nsh-dc-allocation/>.

[moballoc] Napper, J. and S. Kumar, "NSH Context Header Allocation -- Mobility", 2014, <https://datatracker.ietf.org/doc/

draft-napper-sfc-nsh-mobility-allocation/>.

Authors' Addresses

Paul Quinn
Cisco Systems, Inc.

Email: paulq@cisco.com

Jim Guichard
Cisco Systems, Inc.

Email: jguichar@cisco.com

Surendra Kumar
Cisco Systems, Inc.

Email: smkumar@cisco.com

Michael Smith
Cisco Systems, Inc.

Email: michsmit@cisco.com

Wim Henderickx
Alcatel-Lucent

Email: wim.henderickx@alcatel-lucent.com

Tom Nadeau
Brocade

Email: tnadeau@lucidvision.com

Puneet Agarwal

Email: puneet@acm.org

Rajeev Manur
Broadcom

Email: rmanur@broadcom.com

Abhishek Chauhan
Citrix

Email: Abhishek.Chauhan@citrix.com

Joel Halpern
Ericsson

Email: joel.halpern@ericsson.com

Sumandra Majee
F5

Email: S.Majee@F5.com

Uri Elzur
Intel

Email: uri.elzur@intel.com

David Melman
Marvell

Email: davidme@marvell.com

Pankaj Garg
Microsoft

Email: Garg.Pankaj@microsoft.com

Brad McConnell
Rackspace

Email: bmcconne@rackspace.com

Chris Wright
Red Hat Inc.

Email: chrisw@redhat.com

Kevin Glavin
Riverbed

Email: kevin.glavin@riverbed.com

Hong (Cathy) Zhang
Huawei US R&D

Email: cathy.h.zhang@huawei.com

Louis Fourie
Huawei US R&D

Email: louis.fourie@huawei.com

Ron Parker
Affirmed Networks

Email: ron_parker@affirmednetworks.com

Myo Zarny
Goldman Sachs

Email: myo.zarny@gs.com

Service Function Chaining (SFC)
Internet Draft
Intended status: Informational
Expires: August 2014

B. Rijsman
J. Moisand
Juniper Networks

February 12, 2014

Metadata Considerations
draft-rijsman-sfc-metadata-considerations-00.txt

Status of this Memo

Copyright Notice

Abstract

This draft discusses the topic of metadata in the context of Service Function Chaining. It aims to define the problem space for metadata signaling, identify the key challenges, and guide the exploration and comparison of possible solutions.

Table of Contents

1. Introduction

This draft discusses the topic of metadata in the context of Service Function Chaining.

As described in [draft-quinn-sfc-arch] a Service Function Chain (or Service Chain for short) is a sequence of Service Functions such as Network Address Translation (NAT), Firewalls, Deep Packet Inspection (DPI), Intrusion Detection Systems (IDS), content caches, etc.

Service Functions process the traffic flows which traverse the Service Function Chain. In additional to the data which is in the processed traffic flow itself, the Service Functions may also benefit from additional contextual information about the traffic flows. This additional contextual information is referred to as metadata. One example of metadata is an Accounting-ID which is used for accounting and billing purposes.

This draft aims to define the problem space for metadata signaling, identify candidate approaches, describe the key challenges, and guide the exploration and comparison of possible solutions.

Section 2 defines what metadata is and what it can be used for, i.e. use cases for metadata.

Section 3 lists five different approaches for signaling metadata: in-band signaling (attaching metadata to each packet in a traffic flow), signaling metadata in application layer headers, congruent out-of-band metadata signaling, non-congruent out-of-band metadata signaling, and a hybrid approach combining in-band signaling with out-of-band signaling.

Section 4 discusses various challenges with metadata and describes the pros and cons of each of the five approaches in terms of dealing with these challenges.

Section 5 contains a summary and conclusion.

2. Metadata use cases

2.1. What is metadata?

Metadata is "data about data".

In the context of Service Function Chaining, metadata provides contextual information about the data packets which traverse a Service Function Chain.

The following sections provide concrete examples of what metadata can be used for.

2.2. Contextual information which is not locally available

Metadata can be used to transport contextual information which is available at one location in the network to another location in the network where that information is not readily available.

Edge routers often have detailed information about each traffic flow such as the subscriber identity (Subscriber-ID) and the associated policies (i.e. sets of rules which need to be applied to the traffic flow).

This information is available at the network edge as a result of the customer-facing interface on which the traffic arrives, or as a result of information in encapsulations or protocols which are stripped off at the edge of the network, or as a result of interactions with policy servers such as Authentication, Authorization, and Accounting (AAA) servers.

Deeper in the network this detailed information is either not available or difficult to obtain.

We illustrate this using two concrete examples:

- In Long Term Evolution (LTE) mobile networks, the Packet Data Network Gateway (PGW) sits at the edge of the IP network. The PGW decapsulates packets from the General Packet Radio Service Tunneling Protocol (GTP). The PGW uses the Diameter protocol [RFC6733] to interact with the Policy and Charging Rules Function (PCRF) server. As a result, the PGW knows the subscriber identity and policy for each traffic flow. This information is not easily available deeper in the network.

- In fixed broadband networks, the Broadband Network Gateway (BNG) sits at the edge of the IP network. The BNG decapsulates packets from a variety of encapsulations such as Point-to-Point over Ethernet (PPPoE). The BNG interacts with an Authentication, Authorization, and Accounting (AAA) server using the Remote Dial In User Service (RADIUS) protocol. As a result, the BNG know the subscriber identity and policy for each traffic flow. This information is not easily available deeper in the network.

These edge routers (such as PGWs or BNGs) may be used as the starting point for a Service Function Chain.

Metadata can be used to signal information from the place where it is readily available (e.g. the PGW or the BNG at the start of the Service Function Chain) to places deeper in the network where it is not readily available (e.g. the Service Functions in the Service Function Chain).

Examples of such metadata include:

- A Subscriber-ID, or more accurately, an Accounting-ID which maps traffic flows to a unique identifier used for accounting and billing purposes.

- A Service-Profile-ID and/or Service-Profile-Parameters which identify the service which the Service Functions must apply to the traffic flow. This is discussed in more detail in section 2.4.

2.3. Avoiding repeated execution of expensive operations

Certain types of information are computationally expensive to extract from the traffic flow.

For example, it is computationally expensive to perform Deep Packet Inspection (DPI) because the TCP stream may have to be reconstructed and sophisticated layer-7 pattern matching algorithms must be applied.

If multiple Service Functions need the same information, it is more efficient to perform the computationally expensive operation (e.g. DPI) only once and to share the result with other Service Function in Service Function Chain using metadata.

As a concrete example of a use case that can benefit from not performing the same expensive operation more than once, we consider some examples of Service Functions which could benefit from the availability of metadata (namely an Application-ID) provided by a DPI Service Function earlier in the Service Function Chain.

A caching Service Function may only want to attempt to cache YouTube video traffic but not Netflix video traffic.

A firewall Service Function for a human resources department may want to block Facebook games but allow other type of Facebook traffic.

A WAN optimization Service Function may want to optimize database synchronization traffic but not video conference streaming.

Each of these Service Functions needs to know what type of application layer traffic is carried in each traffic flow.

Rather than repeatedly doing DPI at each Service Function (at the cache, the firewall, and the WAN optimizer) we can do DPI once and associate the resulting Application-ID as metadata with the traffic flow.

2.4. Fine-grained policies

For coarse-grained policies, i.e. if the number of policies is small, it is feasible to create a separate Service Function Chain for each policy. The Service Classifier can apply the correct policy to each traffic flow by steering that traffic flow into the corresponding Service Function Chain.

As a concrete example, consider a scenario with two Service Functions: a firewall and a cache. Some traffic flows only visit the firewall, some traffic flows visit only the cache, and some traffic flows visit both the firewall and the cache. This can be implemented by creating three separate Service Function Chains:

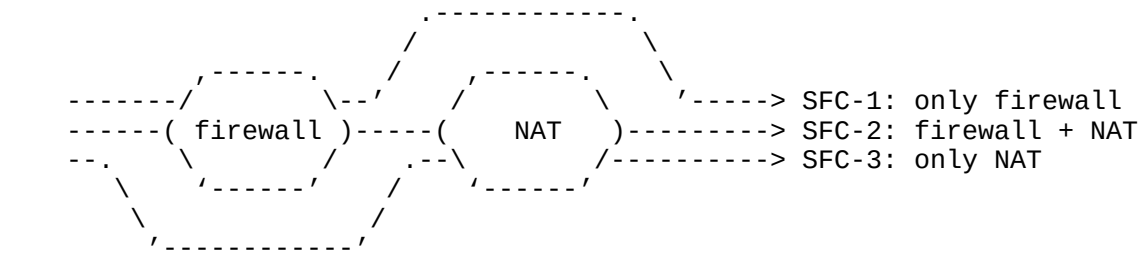


Figure 1: Combinations of Service Functions

As the number of Service Functions increases, the number of possible combinations an permutations increases exponentially. Hence, it may not be feasible or efficient to create separate Service Function Chains.

Furthermore, it may be necessary to apply different service policies to different traffic flows which visit the same sequence of Service Functions.

For example, the firewall may block certain websites for one traffic flow and other websites for a different traffic flow.

One way of implementing this is to create a separate Service Function Chain for each unique combination of service policies.

In the following example we have three Service Function Chains which all visit a firewall and a throttle service. The Service Function Chain implicitly determines which service policy is applied at the firewall and at the throttle service. For example, the third Service Function Chain (SFC-3) receives blocks Facebook at the firewall and is limited to 30 Mbps at the throttle service:

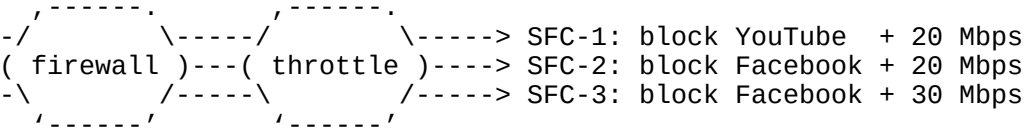


Figure 2: Combinations of service policies

Once again, as the number of service policies increases at each Service Function, the number of possible combinations of service policies increases exponentially. Hence, it may not be feasible or efficient to create separate Service Function Chains.

This problem is aggravated when the service policies become more "fine grained". For example, in the example in figure 2 above we only have two possible values for the bandwidth namely 20 Mbps or 30 Mbps. Consider what would happen if we had ten or more possible bandwidth values: the number of possible service policies combinations would suffer a combinatorial explosion.

In the extreme case, if each customer could have his own individual bandwidth limit, this would imply a Service Function Chain for each individual customer.

Instead of creating separate Service Function Chains, we can use metadata to signal the fine grained policy information. The metadata can identify which service policy needs to be applied at each Service Function, and the metadata can carry fine grained service policy parameters (such as the bandwidth in the above example) in the form of parameter values for a policy profile or policy template.

There is a trade-off between simplicity and flexibility. If the operator provides a small number of uniform network services, then the simple approach of using one Service Function Chain per network service suffices and no fine grained policy metadata is needed. On

the other hand, if the number of unique network services is large or variable (i.e. dependent on the subscriber) then the additional complexity of fine grained policy metadata is justified by the additional flexibility it provides.

3. Metadata signaling approaches

This section describes various approaches for signaling metadata.

3.1. In-band marking

The metadata can be transferred by attaching a metadata field to each packet which traverses the Service Function Chain.

In this draft we refer to this "in-band" marking because the metadata and the data are carried in the same communications channel, i.e. stored in the same packet.

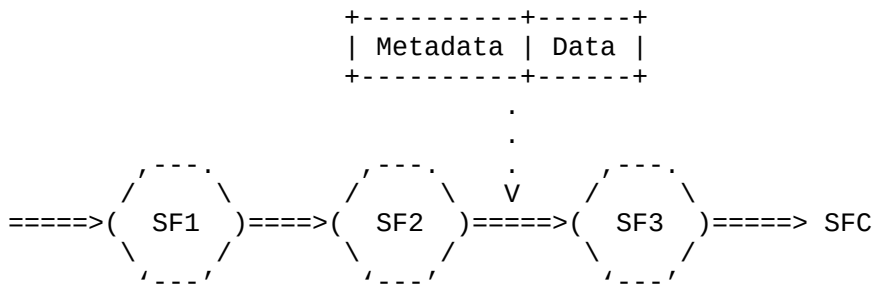


Figure 3: Metadata in each packet

The metadata may simply be attached to each data packet as shown above. Or, the extra field which is attached to each packet may be used to implement a more sophisticated protocol in which case it would be more appropriate to speak of in-band signaling instead of in-band marking. There is a fine line between in-band signaling and congruent out-of-band signaling which is discussed in section 3.3.

The figure above is purposely vague on where exactly in the data packet the metadata is carried. There are several options:

The field which carries the metadata can be a new field which is introduced specifically for the purpose of carrying metadata. The Network Service Header (NSH) described in [draft-quinn-sfc-nsh] is an example of this approach.

Alternatively, the field can be an existing field such as a new IPv4 option or a new IPv6 extension header. This has the advantage of minimizing the impact on existing routers, switches, hosts, and applications; they are able to receive and forward packets with options or extension headers which they do not understand.

Note that the presence of IPv4 options or IPv6 extension headers may cause the packets to be processed in the so-called "slow path" of some core routers instead of the fast path and hence impair forwarding performance.

3.2. Metadata in application layer headers

Another approach is to carry metadata in the headers of application-layer messages.

This approach works well the Hypertext Transfer Protocol (HTTP) [RFC2616] where it easy to introduce new header fields into messages.

Hypothetically, this approach can also be done with similar text-based protocols such as the Simple Mail Transfer Protocol (SMTP) [RFC5321] or the Real Time Streaming Protocol (RTSP) [RFC2326], although in practice it is mostly used with HTTP. This approach does not work well with binary application layer protocols.

Only Service Functions which process traffic at the application layer (e.g. parse HTTP) have access to this kind of metadata.

Routers and switches and Service Functions which process traffic at layer 1 through 4 do not have access to the metadata. This is not necessarily always a disadvantage - the expense of parsing the metadata is incurred only at the places which need it.

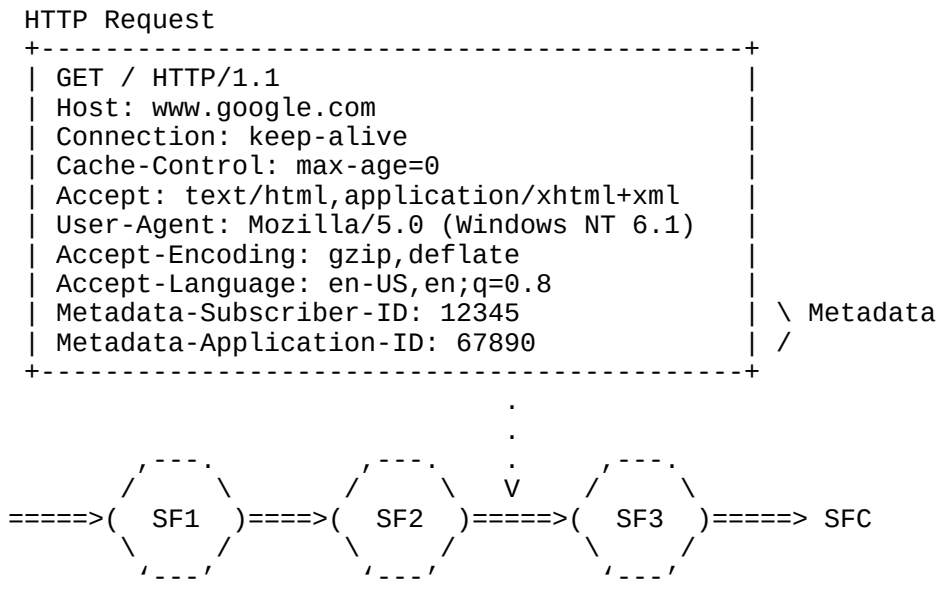


Figure 4: Metadata in HTTP headers

The above example uses hypothetical non-standard headers Metadata-Subscriber-ID and Metadata-Application-ID to carry the metadata. [RFC2774] defines an HTTP extension framework (which is not used in this example) for introducing new HTTP headers. The practice of prefixing non-standard headers with X- has been deprecated by [RFC6648].

3.3. Congruent out-of-band metadata signaling

Metadata can be signaled using a congruent out-of-band control plane protocol.

"Out-of-band signaling" means that the data plane and the control plane for signaling the metadata are carried in different communication channels (i.e. different packets). This is opposed to "in-band marking" which means that the data and metadata are carried in the same communication channel (i.e. in the same packet as discussed in section 3.1.).

"Congruent" means that the data plane protocol and the control plane protocol follow the same path through the network.

The classical example of a congruent out-of-band protocol is the File Transfer Protocol (FTP) [RFC959] which has separate data and control connections, but they follow the same path through the network.

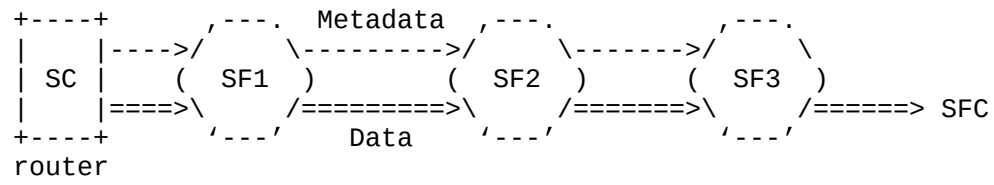


Figure 5: Congruent out-of-band metadata signaling

With congruent out-of-band control plane signaling, the router at the head of the service chain sends control plane messages to the first Service Node (SN) in the Service Function Chain to associate metadata with a particular traffic flow. The Service Node makes the metadata available to the Service Function to consume. The Service Function also has the opportunity to add or modify metadata. The Service Node then uses control plane signaling to propagate the new metadata for the traffic flow to the next Service Node. Etcetera.

New control plane protocols could be introduced for the congruent out-of-band metadata signaling, or existing protocols such as the Layer 2 Tunneling Protocol (L2TP) [RFC2661,RFC3931], or the Resource Reservation Protocol (RSVP) [RFC2205] could be extended to do metadata signaling.

3.4. Non-congruent out-of-band metadata signaling

Metadata can be signaled using a non-congruent out-of-band control plane protocol, separate from the data plane protocol which carries the packets through the Service Function Chain.

"Non-congruent" means that the data plane protocol and the control plane protocol follow different paths through the network.

A classic example of a non-congruent out-of-band protocol is an Interior Border Gateway Protocol (IBGP) [RFC4271] session to a route reflector: the BGP session and the data traffic can follow completely different paths through the network.

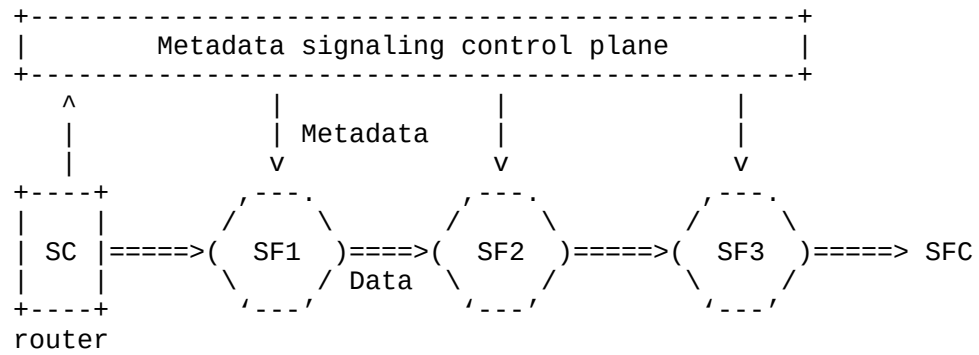


Figure 6: Non-congruent out-of-band metadata signaling

The control plane could consist of a request-response model where the Service Functions explicitly request the metadata when a traffic flow is first observed.

New control plane protocols could be introduced for the non-congruent out-of-band metadata signaling, or existing protocols such as Remote Authentication Dial In User Service (RADIUS) [RFC2865], Diameter [RFC6733], or the Session Initiation Protocol (SIP) [RFC3261] could be extended to do metadata signaling.

Alternatively, the control plane could consist of a publish-subscribe ("pub-sub") message bus. Any Service Function in the Service Function Chain or the router at the start of the Service Function Chain can publish metadata for individual traffic flows. Any Service Function in the Service Function chain can create a subscription to receive metadata for traffic flows. The subscription can include a filter to receive only specific type of metadata of interest to the Service Function.

One could use a pub-sub message bus such as ZeroMQ [zeromq.org] or a pub-sub database such as Redis [redis.io].

3.5. Hybrid in-band marking and out-of-band signaling

Metadata can be signaled using a hybrid approach combining in-band marking and out-of-band signaling.

Each data packet can carry a small fixed-length field which serves as a "correlator". An out-of-band signaling protocol can then be used to map the correlator value to the actual metadata value(s).

The correlator field is typically a form of session identifier; we will use the term Session-ID in this draft. All data packets with the same Session-ID belong to the same session.

The concept of a correlator field is very similar to the concept of a Forwarding Equivalence Class (FEC) which defines a group of packets which need to be treated in the same way.

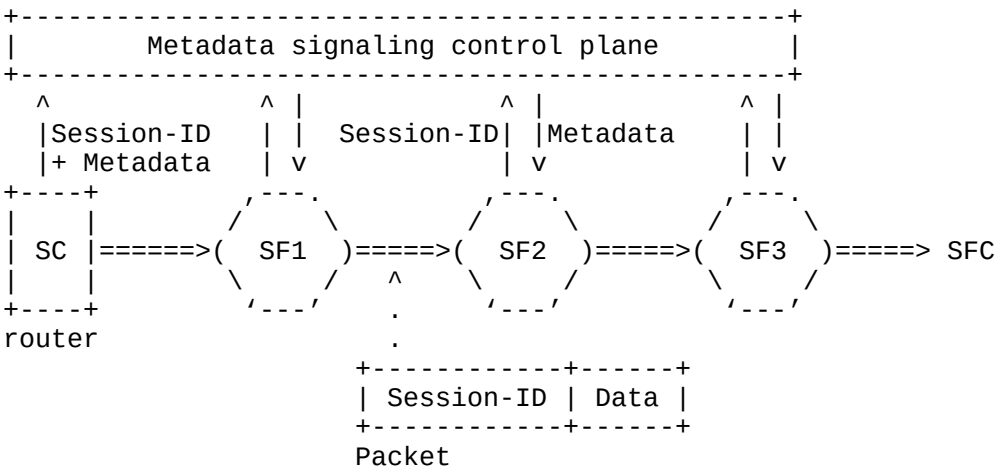


Figure 6: Hybrid in-band marking and out-of-band metadata signaling

The Layer 2 Tunneling Protocol (L2TP) [RFC2661,RFC3931] and the General Packet Radio System Tunneling Protocol (GTP) [GTP] are both examples of hybrid protocols. Each has a data plane protocol (GTP-U in the case of GTP) which carries a Session-ID. The Session-ID is used as a correlator to a separate control plane protocol (GTP-C in the case of GTP).

4. Metadata challenges

This section discusses the challenges associated with metadata.

4.1. Support for metadata-unaware Service Functions

There is already a large number of Service Functions on the market such as firewalls, load balancers, WAN optimizers, caches, DDoS mitigation, etc. These existing Service Functions are available in both physical form and virtual form.

Since metadata has not yet been standardized, the existing Service Functions generally don't understand how to extract and process metadata. They generally expect to receive and send normal IP over Ethernet packets, possibly with a VLAN tag. We refer to such Service Functions as being metadata-unaware.

If the IETF defines a new type of encapsulation header specifically for the purpose of carrying metadata, it will not be possible to inject packets with the new metadata header into existing metadata-unaware Service Functions. Such packets would be dropped because the new encapsulation is not recognized.

Thus, for backwards compatibility it will be required to strip any new metadata headers from the packet before it is injected into a metadata-unaware Service Function.

One possible implementation choice is to always strip the metadata from the packet before injecting it to any Service Function, both metadata-aware and metadata-unaware Service Functions. The metadata-aware Service Functions could choose to retrieve the metadata, for example by calling an API (e.g. a socket call) to retrieve the metadata.

The other approaches (i.e. other than introducing a new header for metadata) can handle metadata-unaware Service Functions more easily.

Existing Service Functions are able to ignore new HTTP headers. Many existing HTTP-aware Service Functions can be configured to recognize specific HTTP headers, even non-standard HTTP headers, and take specific actions based on the presence or value of specific HTTP headers.

Existing Service Functions are able to ignore new IPv4 options or new IPv6 extension headers. However, most existing Service Functions can NOT be configured to take action based on the presence or value of a specific option or extension header.

If an out-of-band control plane protocol is used to signal metadata, metadata-unaware Service Functions can ignore the metadata by simply not participating in the control plane protocol.

4.2. Preserving metadata through metadata-unaware Service Functions

As described in the previous section, there are ways to support a metadata-unaware Service Function in a Service Function Chain, at least in the sense that the metadata-unaware Service Function will not choke on the traffic flow. The metadata-unaware Service

Function will see what appears to be a completely normal IP over Ethernet traffic flow (possibly with a VLAN tag).

However, there is another problem. The metadata should be preserved when it passes through a metadata-unaware Service Function.

Consider, for example, a scenario where the metadata is stripped from the packet before the packet is injected into a metadata-unaware Service Function. For Virtual Service Functions running in a virtual machine, the metadata is typically stripped in the virtual switch (vSwitch) or virtual router (vRouter) in the hypervisor of the virtualized server. For Physical Service Functions running on hardware, the metadata is typically stripped in the proxy node (sometimes called gateway node) in front of the Physical Service Function.

Now imagine there is a subsequent Service Function later in the Service Function Chain which is metadata-aware. In this case we need to re-attach the metadata to the packet after it leaves the metadata-unaware Service Function and before it reaches the metadata-aware Service Function.

Clearly, the metadata-unaware Service Function cannot re-attach the metadata because it is, by definition, not aware of the existence of the metadata.

At first sight, it seems reasonable to expect the vSwitch or vRouter or proxy node which stripped the metadata to also re-attach the metadata. In practice, however, this is extremely difficult to implement. The algorithm for stripping and re-attaching the metadata would look something like this:

1. For every received packet P do the following

2. Strip the metadata from P

3. Store the stripped metadata

4. Inject the packet into metadata-unaware Service Function F

5. Service Function F receives the packet P

6. Service Function F processes packet P

7. Service Function F sends packet P

8. Retrieve the stored metadata for packet P

9. Re-reattach the metadata to packet P

10. Send packet P

This seems simple enough but there are two major problems.

The first problem is in steps 3 and 8. What is the key which we use to store and retrieve the stored metadata? For every given flow, there are going to be many packets P1, P2, ..., Pn which have exactly the same Ethernet source and destination address, the same IP source and destination address, the same protocol ID, the same TCP source and destination port etc. However, packets P1, P2, ..., Pn may have different metadata. Furthermore, certain Service Functions such as Network Address Translation (NAT) and HTTP Proxies change the source or destination addresses or ports of the packets.

The second problem is in steps 5, 6 and 7. There is an implicit assumption that Service Function F sends exactly one packet P for every received packet P. In reality, this is not a valid assumption. Service Function F may drop packets, re-order packets, duplicate packets, insert new packets, or even completely change the traffic flow including the addresses (e.g. a proxy).

A similar problem occurs for metadata which is stored in an existing header such as an IPv4 option or an IPv6 extension header. In this case it is not necessary to strip the metadata from the packet before it is injected into the metadata-unaware Service Function. However, the metadata is typically lost when it traverses the metadata-unaware Service Function. Existing Service Functions on the market are often implemented by opening a pair of sockets, reading packets from one socket, processing the packets, and sending the packets on the other socket. Typically implementations using socket APIs do not preserve IPv4 options or IPv6 extension headers.

How can we deal with this problem?

One possible approach is to simply accept that the metadata is lost once the traffic flow visits a metadata-unaware Service Function. Any subsequent Service Functions in the Service Function Chain do not have access to the metadata anymore.

Another possible approach is to introduce the restriction that if a Service Function Chain contains at least one metadata-aware Service Function, then all Service Functions in the Service Function Chain must be metadata-aware.

A third possible approach is to modify the Service Function implementation to retrieve and maintain the metadata. That is just a different way of saying that all Service Function should be metadata-aware which will take time.

Some of the other approaches (i.e. other than storing the metadata in the same packet as the data) can more easily preserve metadata as it passes through a metadata-unaware Service Function.

All HTTP headers, even unrecognized HTTP headers, are generally preserved when they pass through a Service Function.

The control plane based approaches can simply skip the metadata unaware service. For non-congruent out-of-band metadata signaling this is trivial. For congruent out-of-band metadata signaling this is also easy, but care must be taken that the control plane does not run in the Service Function itself but in the vSwitch or vRouter or proxy node 'below' the Service Function.

4.3. Metadata encoding

There is a tradeoff between flexibility and performance when making a decision on how to encode metadata.

We can use a fixed number of metadata fields or a variable number of metadata fields.

Each metadata field can have a fixed length format or a variable length format.

We can support a fixed set of metadata field types, or we can support an extensible framework where different vendors and users can introduce new metadata field types using some sort of allocation mechanism.

Clearly, supporting a variable number of metadata fields, each with a variable length, each with a different syntax, and with some registration mechanism to introduce new metadata types provides the most flexibility and extensibility.

Such flexibility and extensibility is perfectly feasible for some approaches, namely when we carry metadata in HTTP header, or when we use extensible out-of-band signaling protocols such as Diameter.

However, when the metadata is attached to each packet (either in a new header or in an existing field such as an IPv4 option or an IPv6

extension header), flexibility and extensibility come at such a high cost that it is not practical.

When metadata is attached to a packet it must still be possible to forward packets in the fast path of a router or switch. It is very difficult and expensive to deal with variable-length encoding and with fields with an unbounded length in hardware. At first sight, this is not a problem because the metadata can be stored behind the Ethernet header and the IP header. However, in practice it is a problem. The fast path must often look deeper into the packet for Access Control Lists (ACLs), Quality of Service (QoS), firewalling, tunnel decapsulation, etc.

4.4. Scalability and performance of the control plane

Scalability and performance of the control plane is a concern when we use out-of-band metadata signaling.

Typically there will be at least one control plane message exchange for every new traffic flow or for every new session which passes through a service chain to associate metadata with that traffic flow. The rate at which traffic flows or sessions appear and disappear can be very high: thousands per second or more. This requires a very high performance control plane. The number of simultaneously active traffic flows or sessions, each with its own state, can be very high: millions or more. This requires a very scalable control plane.

The control plane interaction also causes extra latency for the data plane traffic.

4.5. Synchronization between the control plane and the data plane

In the out-of-band signaling based approaches, the data and the metadata are carried over separate communication channels. The data is carried over a data plane channel, and the metadata is carried over a control plane channel.

The data and the metadata for a new flow do not arrive simultaneously at a Service Function. The data may arrive before the metadata, or the other way around.

As a result, the Service Function may have to buffer the data packets while waiting for the metadata to arrive. This requires stateful processing of traffic flows which is difficult and expensive to implement in hardware. Also, the memory for the additional buffers increases the cost.

Alternatively, the Service Function may simply drop the data while waiting for the corresponding metadata, relying on the application layer to retransmit the data. This increases latency and decreases end-user perceived quality.

The delay and buffering problem is aggravated for non-congruent out-of-band signaling (compared to congruent out-of-band signaling) because the control plane uses a different path for the control plane which is typically much slower than the data plane path because it uses slower interfaces and passes through intermediate control plane nodes.

4.6. Distributing metadata only to interested Service Functions

It is highly desirable that metadata is only signaled to those Service Functions which actually need it.

In other words, ideally a Service Function should only receive:

- Metadata for those traffic flows which actually visit that instance of the Service Function. For example, if a Service Function Chain contains multiple parallel scale-out instances of a firewall, each firewall should only receive metadata for the traffic flows which actually pass through that instance of the firewall.

  This is easy to achieve with in-band marking and with congruent out-of-band signaling. With non-congruent out-of-band signaling is also easy to achieve if a "pull" model is used, but it is non-trivial to achieve if a "publish" model is used.

- Metadata which is of interest to that particular Service Function. For example is a Service Function chain contains a sequence of a firewall and a cache, then the firewall should only achieve the metadata which of interest to the firewall and the cache should only receive the metadata which is of interest to the cache. There could of course also be metadata which is of interest to both the firewall and the cache.

  This is easy to achieve with non-congruent out-of-band signaling. It is more difficult to achieve with in-band marking and with congruent out-of-band signaling.

4.7. Associating data plane flows with control plane signaling

If we use out-of-band control plane signaling for metadata, there must be a way to associate a control plane message with a data plane flow.

There will be a control plane message which says "use metadata M for traffic flow F" or some variation thereof.

The question is: how is traffic flow F identified?

A seemingly obvious answer is to use the 5-tuple (source IP, destination IP, protocol, source port, destination port) of the flow or some subset of the 5-tuple.

This approach is thwarted by the fact that many (possibly most) service chains contain one or more services which modify the 5-tuple, such as for example Network Address Translation (NAT) or HTTP proxies.

One possible approach is to use the IPv6 flow label field. Unfortunately, there is not standard flow label field for IPv4. Plus, existing Service Functions are not likely to preserve the flow label as discussed in section 4.2.

Another possible approach is to use an approach which is a hybrid between the in-band marking approach and the out-of-band signaling approach as described in section 3.5.

4.8. Layering considerations

As described in [draft-quinn-sfc-nsh] Service Function Chains consists of one or more parallel Service Paths.

Each Service Path consists of a concatenation of Service Functions which are connected by Service Path Segments. These Service Functions may be Virtual Service Functions on Service Nodes. Or, they may be Physical Service Functions connected to proxies or gateways.

The Service Classifier at the head of the Service Function Chain perform load balancing by steering each traffic flow in the chosen Service Path.

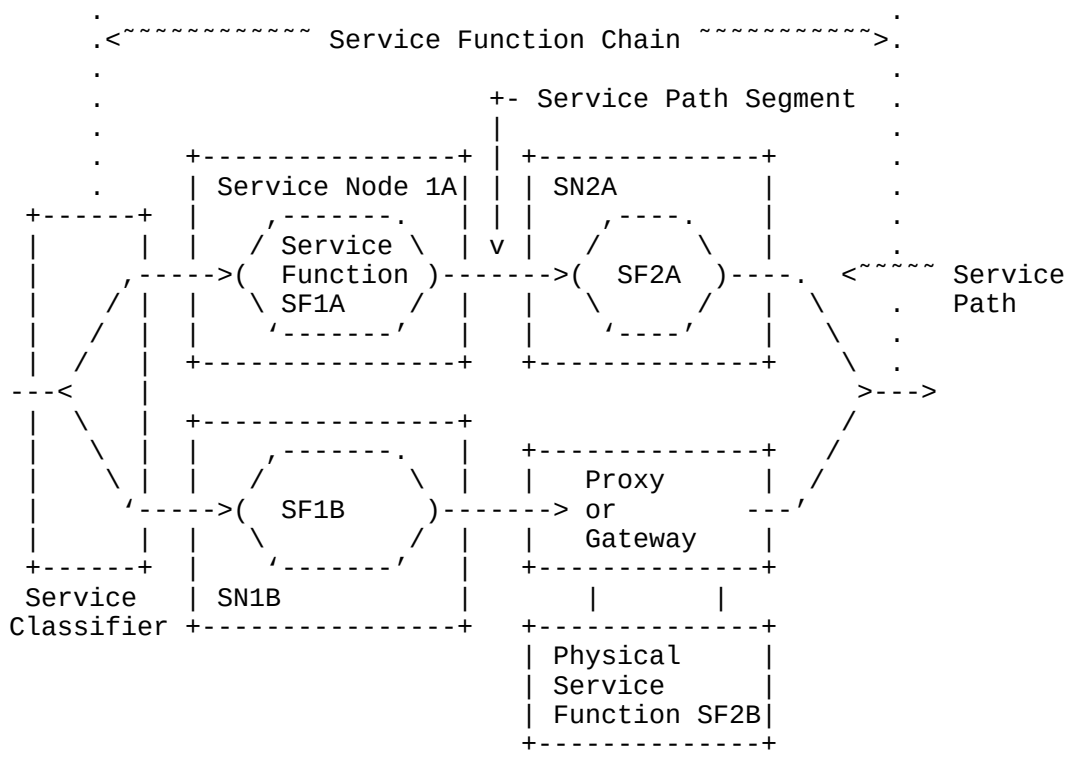


Figure 7: Service Function Chains, Paths, and Path Segments

It is useful to think of this architecture in terms of layers:

1. The service function layer is responsible for performing the service at each Service Function and for steering the packets into the right Service Path.

2. The service path layer is responsible for transporting data packets from one Service Function to the next Service Function in the Service Path.

3. The traditional network layer provided by the underlay network is responsible for forwarding packets from Service Node to Service Node.

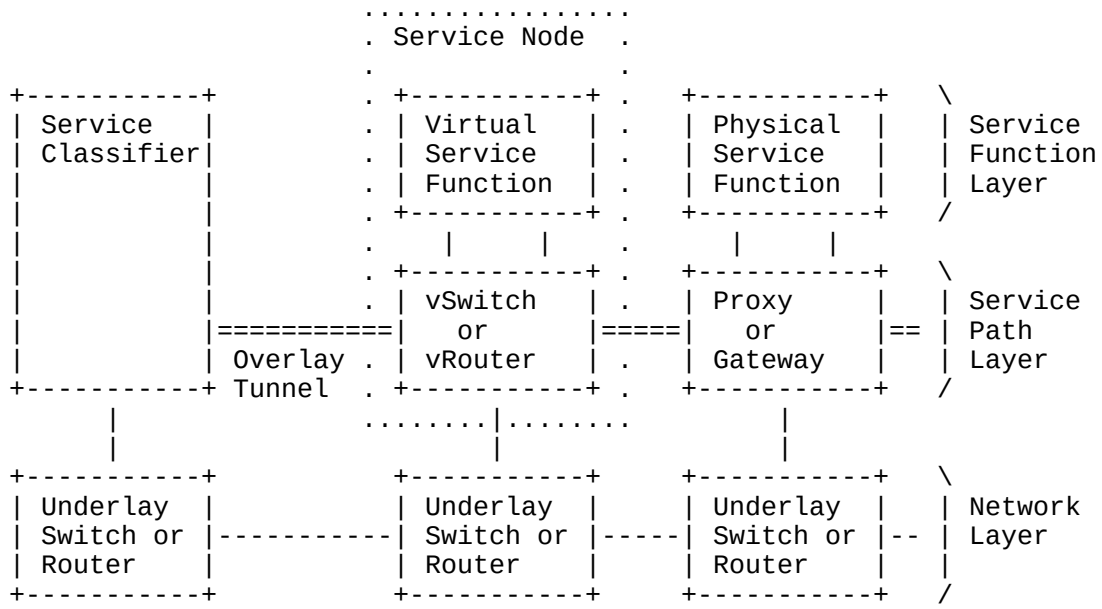


Figure 8: Layers

In the service path layer, each Service Path Segment is typically implemented using some sort of overlay tunnel.

The overlay tunnel can be implemented in various ways, including Virtual Local Area Networks (VLANs) [802.1Q], Virtual Extensible Local Area Networks (VXLANs) [draft-mahalingam-dutt-dcops-vxlan], Generic Route Encapsulation (GRE) [RFC2784] tunnels, or Label Switched Paths (LSPs) [RFC3031].

At the end of each Service Path segment there is some entity which decapsulates the packets from the overlay tunnel and dispatches the packets into the right Service Function. For Virtual Service Functions may be a virtual switch (vSwitch) or virtual router (vRouter). For Physical Service Functions this may be a proxy or gateway.

The tunnel encapsulation needs a field to identify the Service Path Segment. The vSwitch or vRouter or proxy or gateway uses this field to dispatch each packet into the correct Service Function. This field can be implemented in multiple ways, including a VLAN tag, a Virtual Network Identifier (VNID) in the VXLAN header, a Multi-Protocol Label Switching (MPLS) label, or the service path field in

the base service header of the Network Service Header [draft-quinn-sfc-nsh].

In addition to the field which identifies the Service Path as just discussed, there may be more fields in the packet which contain additional metadata such for example as the Session-ID or the Accounting-ID.

These additional metadata fields are only relevant in the service function layer. They are attached by the Service Classifier. They are read and acted upon by the Service Functions. The vSwitches, vRouters, proxies, or gateways never need to read or write this additional metadata.

Thus, it is important to make a distinction between fields which are used at the service path layer to identify the Service Path Segment, and additional fields which carry metadata which is imposed and interpreted at the service function layer. Combining both types of fields into a single header should probably be avoided from a layering point of view.

The need for such separation is reinforced by the existence of various ways to convey metadata that were discussed in chapter 3. It isn't necessarily obvious that packet marking is the best way to implement the service function layer.

4.9. Load balancing and symmetry

In a scaled-out service, i.e. when the Service Function Chain consists of more than one Service Path, load balancing is used to assign each traffic flow to exactly one of the available Service Paths.

Often the Service Functions in a Service Function chain are "stateful" and as a result they may need to see all packets in a flow in both directions. In that case, the forward and the reverse traffic flow must be assigned to the same Service Path, i.e. load balancing must be symmetric. (This is actually difficult to achieve when the different Service Paths terminate on different Service Nodes, but that discussion is beyond the scope of this draft.)

The metadata signaling protocol must ensure that the metadata for a given traffic node is signaled to the same Service Path (i.e. the same set of Service Nodes) which process the data for that traffic flow. Ideally, the metadata is not signaled to any other Service Nodes, i.e. not signaled to places where it is not needed.

This is easy to achieve when the metadata is attached to the data and it is also easy to achieve with congruent out-of-band signaling. It is more challenging to achieve for non-congruent out-of-band signaling, particularly in a pub-sub model.

4.10. High availability and geographic dispersion

Any kind of construct requiring high-rate out-of-band communication with a policy or signaling server is subject to all sorts of challenges if said server is suddenly no longer reachable (not necessarily a crash, but a simple connectivity loss). In other words, one has to factor in the geographical distribution of the problem.

In the non-congruent out-of-band signaling approach there may be logically centralized servers (e.g. RADIUS or Diameter servers) that communicate with the Service Nodes and the Service Classifiers.

The Service Nodes and Service Classifiers may be geographically dispersed. A single Service Function Path may visit Service Functions which are located in a central office, a distributed regional data/service center, and a more centralized data/service center.

Using a single (logical) server to control all geographic locations introduces challenges from a scaling and high availability perspective.

Using multiple (logical) servers improves scalability and reduced the size of failure domains, but introduces complexity because the servers must use federation protocols for coordination.

4.11. Multiple sources of metadata

In simple use cases it may be sufficient for all metadata to be produced and associated with the traffic flow once at the start of the service chain.

On more sophisticated use case it may be useful to allow Service Functions in the Service Function Chain to associate additional metadata with a traffic flow as the data traverses the service chain.

In the latter case, the signaling mechanism must support multiple sources of metadata.

4.12. Extensibility

The metadata mechanism must be extensible.

As new Service Functions are introduced, it must be possible for to introduce new types of metadata using some sort of registration process.

The extension mechanism must support backward compatibility: it must be possible to distribute a new type of metadata in a Service Chain which includes some Service Functions that do not understand the new type of metadata.

Extensibility is particularly important for the service function layer, more so than for the service path layer (see section 4.8. for layering).

5. Conclusion

In this draft we discussed several metadata signaling approaches:

- In-band marking
- Metadata in application layer headers
- Congruent out-of-band metadata signaling
- Non-congruent out-of-band metadata signaling
- Hybrid in-band marking and out-of-band signaling

We also discussed metadata signaling challenges:

- Support for metadata-unaware service function
- Preserving metadata through metadata-unaware service functions
- Metadata encoding
- Availability and performance of the control plane
- Synchronization between the control plane and the data plane
- Distributing metadata only to interested service functions

- Associating data plane flow with control plane signaling
- Layering considerations
- Load balancing and symmetry
- High availability and geographic dispersion
- Multiple sources of metadata
- Extensibility

We also discussed how each of the signaling approaches was affected by each of the challenges.

The problem of signaling metadata is complex. Whatever signaling approach is chosen multi-vendor interoperability is required, hence the need for standardization.

In general, it is desirable to solve simple problems with simple solutions and to make more complex solutions and mechanisms optional.

A productive path forward could be to divide and conquer: to clearly separate the problem of Service Function Path topology from the problem of metadata. In other words, to make a clear distinction between what we called the service path layer and the service function layer in section 4.8.

The Service Function Chaining working group could initially focus on the service path layer, i.e. the mechanism for steering packets through the right sequence of Service Functions in the Service Path.

Conveying contextual metadata could be viewed as optional, and not applicable to all service chains. Furthermore, the means to address the needs of the service function layer aren’t necessarily the same as those required for the service path layer.

In this draft shy away from recommending one of the approaches as the best approach of all use cases. Different approaches may make sense for different use cases.

6. Security Considerations

The metadata signaling protocol must be secure in the sense that the authenticity (i.e. the validity of the content) and the authority (i.e. the source) must be guaranteed. Non-renumeration (i.e. encryption) of the metadata may or may not be a requirement.

7. IANA Considerations

None.

8. References

8.1. Normative References

None.

8.2. Informative References

[802.1Q] IEEE Std. 802.1Q-2011, Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks.

[draft-mahalingam-vxlan] M. Mahalingam, et al. "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks.", draft-mahalingam-dutt-dcops-vxlan-08, February 3, 2014.

[draft-quinn-sfc-arch] P. Quinn, at al, "Service Function Chaining (SFC) Architecture", draft-quinn-sfc-arch-04, January 28, 2014.

[draft-quinn-sfc-nsh] P. Quinn, et al, "Network Service Header", draft-quinn-sfc-nsh-00, April 10, 2014.

[GTP] 3GPP TS 32.295 V6.1.0 (2005-06), 3rd Generation Partnership Project.

[in-band-control] Wikipedia, "In-band control", http://en.wikipedia.org/wiki/In-band_control

[out-of-band-control] Wikipedia, "Out-of-band control", http://en.wikipedia.org/wiki/Out-of-band_control

[redis.io] "Redis website", http://redis.io

[RFC959] J. Postal and J. Reynolds. "File Transfer Protocol (FTP)", RFC959, October 1985.

[RFC1661] W. Simpson, et al. "The Point-to-Point Protocol (PPP)", RFC1661, July 1994.

[RFC2205] R. Braden, et al. "Resource ReSerVation Protocol", RFC2205, September 1997.

[RFC2326] H. Schulzrinne, et el. "Real Time Streaming Protocol (RSTP)", RFC2326, April 1998.

[RFC2616] R. Fielding, et al. "Hypertext Transfer Protocol -- HTTP/1.1", RFC2616, June 1999.

[RFC2661] W. Townsley, et al. "Layer Two Tunneling Protocol L2TP", RFC2661, August 1999.

[RFC2774] H. Nielsen, et al. "An HTTP Extension Framework", RFC2774, February, 2000.

[RFC2784] D. Farinacci, et al. "Generic Route Encapsulation (GRE)", RFC2784, March 2000.

[RFC2865] C. Rigney, et al. "Remote Authentication Dial In User Service (RADIUS)", RFC2865, June 2000.

[RFC3031] E. Rosen, et al. "Multiprotocol Label Switching Architecture", RFC3031, January 2001.

[RFC3261] J. Rosenberg, et al. "SIP: Session Initiation Protocol", RFC3261, June 2002.

[RFC3931] J. Lau, et al. "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC3931, March 2005.

[RFC4271] Y. Rekhter, et al. "A Border Gateway Protocol 4 (BGP-4)", RFC4271, January 2006.

[RFC5321] J. Klensin. "Simple Mail Transfer Protocol", RFC5321, October 2008.

[RFC6648] P. Saint-Andre, et al. "Deprecating the "X-" Prefix and Similar Constructs in Application Protocols", RFC6648, June, 2012

[RFC6733] V. Fajardo, et al. "Diameter Base Protocol", RFC6733, October 2012.

[zeromq.org] "ZeroMQ website", http://zeromq.org

9. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

The authors thank Ross Callon, Sankar Ramamoorthi, Gary Greenberg, Galina Pildush, Jaspal Kohli, Stuart Mackie, and James Connolly for their review and comments.

Authors' Addresses

Bruno Rijsman
Juniper Networks
brijsman@juniper.net

Jerome Moisand
Juniper Networks
jmoisand@juniper.net