Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 26, 2014

P. Quinn
K. Gray
J. Guichard
S. Kumar
C. Pignataro
M. Smith
N. Yadav
Cisco Systems, Inc.
P. Agarwal
R. Manur
Broadcom
N. Leymann
Deutsche Telekom
A. Beliveau
Ericsson
T. Nadeau
Lucidvision
K. Glavin
Riverbed

January 22, 2014

Service Function Chaining (SFC) Architecture
draft-quinn-sfc-arch-03.txt

Abstract

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

Status of this Memo

Table of Contents

1. Introduction

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

1.1. Scope

The architecture described herein is assumed to be applicable to a single network administrative domain. While it is possible for the principles and architectural 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: 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 SFC network forwarder to one or more connected service functions.

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 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.

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

1.3. Service Function Chaining

Service chaining enables 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 (SFID) 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 ordering and topological independence of those SFs.

A basic SFC may utilize an existing overlay technology alongside service specific forwarding in the network to steer traffic through the ordered set of SFs. However, additional information that is shared across a subset of SFs within an SFC may enable value-added services with a richer set of functionality. For example, shared information, such as the results of a classification function, may be passed to downstream SFs to enable the offloading of service function

processing. As another example, sharing of information derived at one SF with the rest of the SFs in the SFC would obviate the need to re-derive the same information and simplify the service.

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. As previously stated a SF can be a virtual instance or be embedded in a physical network element, and one or more SFs may be deployed within the same physical network element.

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.

```
      ,-+-.         ,---.          ,---.          ,---.
     /     \       /     \        /     \        /     \
    (   1   )+--->(   2   )+---->(   6   )+---->(   8   )
     \     /       \     /        \     /        \     /
      `---'         `---'          `---'          `---'

      ,-+-.         ,---.          ,---.          ,---.          ,---.
     /     \       /     \        /     \        /     \        /     \
    (   1   )+--->(   2   )+---->(   3   )+---->(   7   )+---->(   9   )
     \     /       \     /        \     /        \     /        \     /
      `---'         `---'          `---'          `---'          `---'

      ,-+-.         ,---.          ,---.          ,---.          ,---.
     /     \       /     \        /     \        /     \        /     \
    (   1   )+--->(   7   )+---->(   8   )+---->(   4   )+---->(   7   )
     \     /       \     /        \     /        \     /        \     /
      `---'         `---'          `---'          `---'          `---'
```

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 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. Service Function Chaining Architecture

3.1. 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. SFC Encapsulation: The SFC encapsulation enables the sharing of metadata/context: the network and SFs no longer exist in separate silos. Metadata/context data can be shared amongst SF and classifiers. In addition to metadata, the encapsulation provides information used to identify the SFP. Transit nodes -- such as router and switches -- simply forward SFC encapsulated packets based on the outer (non-SFC) encapsulation.

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.

3.2. Fundamental Components

The following logical components form the basis of the SFC architecture:

1. 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 such, SFs have one or more network locators and a variable set of attributes that describe the function offered. The combination of network locator and attributes are used to construct an SFC. SF send/receive SFC encapsulated data from one or more Service Function Forwarders.

2. Service Function Forwarder (SFF): a service function forwarder provides service layer forwarding. An SFF receives packets from a SFC Network Forwarder (see below) and forwards the traffic to the required associated SF(s).

3. SFC Network Forwarder (SNF): This component is responsible for forwarding traffic flows along the SFPs they belong to based on information contained in the SFC encapsulation. Since SFCs straddle both the service layer (via the SFC encapsulation) and the network layer (via the network transport), SNFs can provide service path load distribution and failover functionality. For example, SNFs might have two network paths between SF1 and SF2 and utilize local metrics for path selection. Similarly, if a path fails, the SFC can utilize local failover to select alternate path(s).

```
       +----------------+
       |Service Function|
       |     (SF)       |
       +-------+--------+
               |
        SFC Encapsulation
               |
       +-------+--------+
       | SF    Forwarder|
       |      (SFF)     |
       +-------+--------+
               |
        SFC Encapsulation
               |
       +-------+--------+
       | SFC Network    |
       | Forwarder (SNF)|
       +----------------+
               |
    Network Overlay Transport
               |
           _,....._
        ,-'        `-.
       /              `.
      |     Network    |
       `.             /
         `.__     _,-'
```

```
             `''''
```

Figure 2: Service Function Components

Classifier: A component that performs traffic classification. Classification is the precursor to the start of an SFP: traffic that matches classification criteria is forwarded along a given SFP to realize the specifications of an SFC. The granularity of classification varies based on operator requirements and device capabilities. While initial classification at a network node starts an SFP, subsequent classifications may occur along the SFC and further alter the SFP. This re-classification may also update the context information (see below).

Overlay Service Topology: A service topology is created to interconnect the elements used to form the SFP. This overlay topology is specific to the SFP: it is created for the express purpose of steering packets or frames through the SFs and optionally passing context data. The overlay is formed between SNF elements. The overlay topology can be constructed using any existing transport, for example IP, MPLS, etc.

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 exist within distributed routing elements as in traditional networks, or in a centralized configuration.

Shared context data: Sharing context data 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 context along the SFP. The shared context serves several possible roles within the SFC architecture:

- o Allows elements that typically operate as ships-in-the-night to exchange information.

- o Encodes information about the network and/or data for post-service forwarding.

o Creates an identifier used for policy binding by SFs.

o Context information can be derived in several ways:

   * External sources

   * Network node classification

   * Service function classification

o 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.

The figure below provides a high level view of the components:

```
                                 +-------+
                     +---------->|control|<----------+
                     |           |plane  |           |
     +-------------->|           +---+---+           |
     |               |               ^               |
     |               |               |               |
     v               v               v               v
+----------+       ,---.           ,---.            ,---.           +----------
+
|classifier|+---> /     \+------->/     \+-------->/     \+-------->|
classifier|
|          |     (   1   )<-----+(   2   )<------+(   3   )<-------+|
|
+----------+      \     /         \     /          \     /          +----------
+
                   `---'           `---'            `---'
```

Figure 3: Service Function Chaining Architecture

4. 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.

5. Security Considerations

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

6. Acknowledgments

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

A special thank you goes to Joel Halpern for his thoughtful, detailed review and guidance.

7. IANA Considerations

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

8. References

8.1. Normative References

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

8.2. Informative References

[NSCprob] "Network Service Chaining Problem Statement", <http://datatracker.ietf.org/doc/draft-quinn-nsc-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.

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].

Authors' Addresses

Paul Quinn
Cisco Systems, Inc.

Email: paulq@cisco.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

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

Puneet Agarwal
Broadcom

Email: pagarwal@broadcom.com

Rajeev Manur
Broadcom

Email: rmanur@broadcom.com

Nic Leymann
Deutsche Telekom

Email: n.leymann@telekom.de

Andre Beliveau
Ericsson

Email: andre.beliveau@ericsson.com

Thomas Nadeau
Lucidvision

Email: tnadeau@lucidvision.com

Kevin Glavin
Riverbed

Email: Kevin.Glavin@riverbed.com