Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 18, 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
S. Majee
F5
U. Elzur
Intel
D. Melman
Marvell
P. Garg
Microsoft
B. McConnell
Rackspace
C. Wright
Red Hat Inc.
K. Glavin
Riverbed

December 15, 2014

Network Service Header
draft-quinn-sfc-nsh-04.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

This Internet-Draft will expire on June 18, 2015.

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 is 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

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

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Base Header                                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Service Path Header                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~           Optional Variable Length Context Headers            ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

Variable length context headers: carry opaque metadata and variable length encoded information.

3.2. NSH Base Header

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver|O|C|R|R|R|R|R|R|   Length  |    MD Type    | Next Protocol |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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 and may contain optional variable length 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 MAY 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

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          Service Path ID                      | Service Index |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

```
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Ver|O|C|R|R|R|R|R|R|   Length  |  MD-type=0x1  | Next Protocol |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          Service Path ID                      | Service Index |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                Mandatory Context Header                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                Mandatory Context Header                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                Mandatory Context Header                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                Mandatory Context Header                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    ~           Optional Variable Length Context Headers            ~
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: NSH MD-type=0x1

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Context data                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Context Header

3.4.1. Mandatory Context Header Allocation Guidelines

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Network Platform Context                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Network Shared Context                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Service Platform Context                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Service Shared Context                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

3.4.2. Optional Variable Length Metadata

In addition to the required, fixed size context headers, MD Type 1 NSH MAY also contain optional variable length context headers. These header are formatted as TLVs.

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |          TLV Class            |      Type     |R|R|R|   Len   |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                      Variable Metadata                        |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: 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.

```
  +-+--------+
  |C| Type   |
  +-+--------+
```

Figure 8: 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.

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.

```
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Ver|O|C|R|R|R|R|R|R|   Length  |  MD-type=0x2  | Next Protocol |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          Service Path ID                      | Service Index |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    ~           Optional Variable Length Context Headers            ~
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: 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 in section 3.4.2 above.

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, a SFF, removes NSH. A service classifier MUST insert a 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. If a SF performs re-classification that results in a change of service path, it MUST remove the existing NSH and MUST imposes 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 is in use (acting on behalf of a non-aware service function for NSH actions), then the proxy MUST update service index and MAY update contexts.

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.

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

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 header 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 can be used with overlay and underlay forwarding topologies.

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, to the service node, the original packet/frame via a local attachment circuit. 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, a SFF will act as a 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 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 provide a clear, unambiguous identification and location of a SF (locator and order). 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.).

The mapping of SPI to transport occurs on a 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 a 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 10: 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 11: 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 12: SF Locator Mapping Example

Since the SPI is an 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 simple preference (similar to routing next-hop), to a true dynamic composite metric based on 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 13: NSH Weighted Service Path

9.2. Mapping NSH to Network Overlay

As described above, the mapping for SPI to the 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 abililty 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 such configure and 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 it's 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 traverse, however, increasingly, the "path" is actually a directed graph, and as such support cycles. 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.

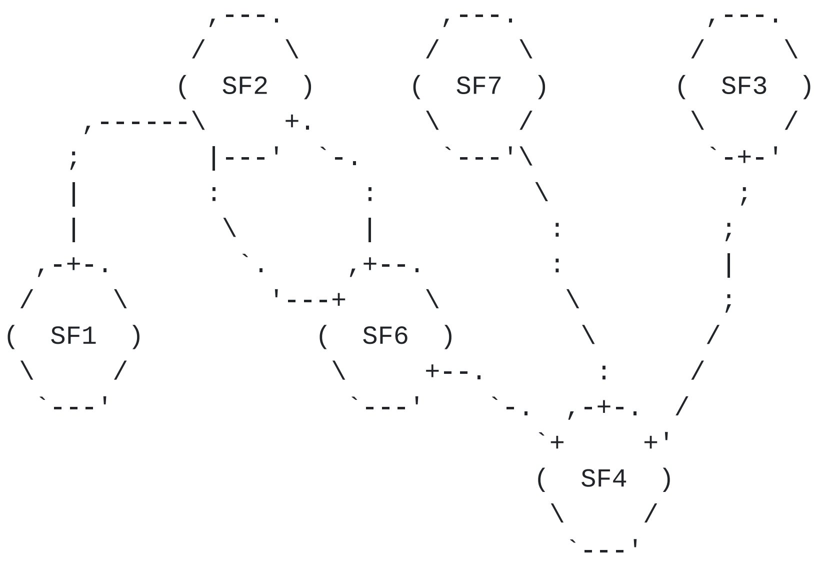

Figure 14: Service Graph Example

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 suport 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 participant 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.

```
 +---------------------------------------------------------------------+
 |                                                SPI: 21 Bob: SF7     |
 |                  SPI: 20 Bad : SF2 --> SF6 --> SF4                  |
 |SPI: 10 SF1 --> SF2 --> SF6                     SPI: 22 Alice: SF3   |
 |                  SPI: 30 Good: SF4                                  |
 |                             SPI:31 Bob: SF7                         |
 |                             SPI:32 Alice: SF3                       |
 +---------------------------------------------------------------------+
```

Figure 15: 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 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, often contain information that valuable for service function policy decisions. In most cases, this information cannot be deduced by network nodes. For example is a 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 5-tuple, whereas, a service function may be able to inspect application information. Regardless of granularity, the classification information is represented in NSH.

Once the data is added to NSH, it is carried along the service path, participant SF 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:

```
    +-------------------------------------------------+
    |   ,---.             ,---.              ,---.    |
    |  /     \           /     \            /     \   |
    | (  SCL  )-------->(  SF1  )--------->(  SF2  )  |
    |  \     /           \     /            \     /   |
    |   `---'             `---'              `---'    |
    |5-tuple:            Permit             Inspect   |
    |Tenant A            Tenant A           AppY      |
    |AppY                                             |
    +-------------------------------------------------+
```

Figure 16: Metadata and Policy

```
    +-------------------------------------------------+
    |    ,---.             ,---.              ,---.   |
    |   /     \           /     \            /     \  |
    |  (  SCL  )-------->(  SF1  )--------->(  SF2  ) |
    |   \     /           \     /            \     /  |
    |    `-+-'             `---'              `---'   |
    |      |              Permit            Deny AppZ |
    |  +---+---+          employees                   |
    |  |       |                                      |
    |  +-------+                                      |
    |  external                                       |
    |  system:                                        |
    |  Employee                                       |
    |  App Z                                          |
    +-------------------------------------------------+
```

Figure 17: 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: An NSH may add information to existing metadata, as depicted in figure 18. For example, if the initial classification returned 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 classification may update the initial classification if it is determined to be incorrect or not descriptive enough. For example, the initial classifier imposed 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.

```
     +-------------------------------------------------+
    |    ,---.             ,---.              ,---.   |
    |   /     \           /     \            /     \  |
    |  (  SCL  )-------->(  SF1  )--------->(  SF2  ) |
    |   \     /           \     /            \     /  |
    |    `-+-'             `---'              `---'   |
    |      |              Inspect           Deny      |
    |  +---+---+          employees         employee+ |
    |  |       |          Class=AppZ        appZ      |
    |  +-------+                                      |
    |  external                                       |
    |  system:                                        |
    |  Employee                                       |
    |                                                 |
    +-------------------------------------------------+
```

Figure 18: Metadata Augmentation

```
    +-------------------------------------------------+
    |   ,---.             ,---.              ,---.    |
    |  /     \           /     \            /     \   |
    | (  SCL  )-------->(  SF1  )--------->(  SF2  )  |
    |  \     /           \     /            \     /   |
    |   `---'             `---'              `---'    |
    |5-tuple:           Inspect             Deny      |
    |Tenant A           Tenant A            attack    |
    |                    --> attack                   |
    +-------------------------------------------------+
```

Figure 19: 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 metdata, 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.

```
    +----------------------------------------------------+
    |   ,---.             ,---.              ,---.       |
    |  /     \           /     \            /     \      |
    | (  SCL  )-------->(  SF1  )--------->(  SF2  )     |
    |  \     /           \     /            \     /      |
    |   `---'             `---' \              `---'     |
    |5-tuple:            Inspect \            Original   |
    |Tenant A            Tenant A \           next SF    |
    |                     --> DoS  \                     |
    |                                 \                  |
    |                                  ,---.             |
    |                                 /     \            |
    |                                (  SF10 )           |
    |                                 \     /            |
    |                                  `---'             |
    |                                   DoS              |
    |                                "Scrubber"          |
    +----------------------------------------------------+
```

Figure 20: 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

```
    IPv4 Packet:
    +----------+--------------------+--------------------+
    |L2 header | L3 header, proto=47|GRE header,PT=0x894F|
    +----------+--------------------+--------------------+
    --------------+----------------+
    NSH, NP=0x1   |original packet |
    --------------+----------------+


    L2 Frame:
    +----------+--------------------+--------------------+
    |L2 header | L3 header, proto=47|GRE header,PT=0x894F|
    +----------+--------------------+--------------------+
    ---------------+---------------+
    NSH, NP=0x3    |original frame |
    ---------------+---------------+
```

Figure 21: GRE + NSH

11.2. VXLAN-gpe + NSH

```
    IPv4 Packet:
    +----------+--------------------+---------------------+
    |L2 header | UDP dst port=4790  |VXLAN-gpe NP=0x4(NSH)|
    +----------+--------------------+---------------------+
    --------------+----------------+
    NSH, NP=0x1   |original packet |
    --------------+----------------+


    L2 Frame:
    +----------+--------------------+---------------------+
    |L2 header | UDP dst port=TBD   |VXLAN-gpe NP=0x4(NSH)|
    +----------+--------------------+---------------------+
    ---------------+---------------+
    NSH,NP=0x3     |original frame |
    ---------------+---------------+
```

Figure 22: VXLAN-gpe + NSH

11.3. Ethernet + NSH

```
  IPv4 Packet:
  +----------+--------------------+---------------+--------------------+
  |Outer Ethernet, ET=0x894F      | NSH, NP = 0x1 | original IP Packet |
  +----------+--------------------+---------------+--------------------+


  L2 Frame:
  +----------+--------------------+---------------+----------------+
  |Outer Ethernet, ET=0x894F      | NSH, NP = 0x3 | original frame |
  +----------+--------------------+---------------+----------------+
```

Figure 23: 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. 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

14. Acknowledgments

The authors would like to thank Nagaraj Bagepalli, Abhijit Patra, Ron Parker, Peter Bosch, Darrel Lewis, Pritesh Kothari 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.

15. IANA Considerations

15.1. NSH EtherType

An IEEE EtherType, 0x894F, has been allocated for NSH.

15.2. 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.

15.2.1. 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

15.2.2. 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

15.2.3. 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].

15.2.4. 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

16. References

16.1. 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.

16.2. 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
Broadcom

Email: pagarwal@broadcom.com

Rajeev Manur
Broadcom

Email: rmanur@broadcom.com

Abhishek Chauhan
Citrix

Email: Abhishek.Chauhan@citrix.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