

Service Function Chaining
Internet-Draft
Intended status: Standards Track
Expires: February 20, 2017

S. Kumar
K. Leung
P. Bosch
Cisco Systems, Inc
D. Lee
SK Telecom
R. Manur
Broadcom
A. Dolganow
P. Muley
Nokia Corp
S. Majee
F5 Networks
J. Halpern
Ericsson
August 19, 2016

Infrastructure Service Forwarding For NSH
draft-kumar-sfc-nsh-forwarding-01

Abstract

This draft describes the separation of service forwarding function and service delivery function abstractions, along with the mechanics of NSH encapsulated packet forwarding with such separation, in SFC deployments.

This separation frees the service functions from making forwarding decisions and the necessary control plane integration, thereby keeping the service functions simple and focused on service delivery. Further, this separation fully contains the forwarding decisions in forwarding functions, thereby allowing implementations to enforce integrity of the forwarding state carried in NSH which in turn is required for correctly forwarding NSH encapsulated packets.

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

Internet-Draft

NSH Service Forwarding

August 2016

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 February 20, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Requirements Language	3
2.	Definition Of Terms	3
3.	NSH And Forwarding	4
3.1.	NSH Background	4
3.2.	NSH Forwarding	5
3.3.	NSH Forwarding Shortcomings	5
4.	SFC Service Forwarding And Service Delivery Separation . . .	6
4.1.	Forwarding Function And Service Function Separation . . .	7
4.2.	NSH Infrastructure Flag	8
4.3.	Rules For Infrastructure Flag Usage	8
4.4.	Service Header Integrity Check	9
4.5.	SF Considerations for Reverse Service Path Packets . . .	10
4.6.	SF Considerations for Spontaneous Packets	12
5.	Infrastructure Forwarding Example	12
6.	Infrastructure Forwarding Advantages	13
7.	Acknowledgements	14
8.	IANA Considerations	14
9.	Security Considerations	15
10.	References	15
10.1.	Normative References	15

10.2. Informative References	15
Authors' Addresses	16

[1.](#) Introduction

SFC involves steering user or application traffic on a service overlay network through a list of ordered service functions before forwarding onwards to its destination, in the process servicing the traffic as per policies in the service functions as well as the SFC infrastructure.

NSH is the encapsulation designed to carry SFC specific forwarding state as well as metadata relevant to service delivery. The forwarding state in the NSH dictates how to forward the encapsulated packet or frame while the metadata aids service delivery by having one SFC entity produce it and the other consume it.

NSH in its current form, as described in [[I-D.ietf-sfc-nsh](#)], blurs the lines between service delivery and service forwarding. This leads to complexities in SFC deployment and operation as the SFC control plane has to deal with a large number of forwarding touch points further challenging the scalability of the deployment. Requiring forwarding decisions be made in the service functions violates operational environment policies due to errors or unintended modification of forwarding state by service functions.

This draft describes the separation of SFC overlay network into a service infrastructure overlay and service function overlay, thereby clearly demarking the boundaries between the two distinct architecture functions. This allows infrastructure components to create and manage the forwarding state as per control plane policy while freeing the service functions to focus on service delivery and not participate in forwarding decisions.

This draft further describes the forwarding process in SFC to achieve such separation that is not only architecturally clean but is friendly to software as well as hardware implementations of SFC infrastructure components.

[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

This document uses some terms defined in SFC architecture [[I-D.ietf-sfc-architecture](#)] and NSH [[I-D.ietf-sfc-nsh](#)] drafts for ease of understanding and the reader is advised to refer to those documents for up to date and additional information.

Kumar, et al.

Expires February 20, 2017

[Page 3]

Internet-Draft

NSH Service Forwarding

August 2016

SFC Infrastructure : A general term used to refer to, collectively, the SFC control plane entity, the classifier, the SFFs. As used freely in the rest of this document, it refers to the infrastructure data plane components - classifiers and SFFs.

Service Infrastructure Overlay : The overlay network extending between SFC infrastructure data plane components. In particular, the overlay network between the SFFs or Classifiers and SFFs

Service Function Overlay : The overlay network extending between the SFF and SF

[3.](#) NSH And Forwarding

[3.1.](#) NSH Background

NSH encapsulation is comprised of three parts as specified in

[[I-D.ietf-sfc-nsh](#)]; namely a base header, a service path header and one or more context headers predicated on MD-type in the base header. The base and service path headers are reproduced in Figure 1 for ease of reading.

1. The base header provides the structure to NSH with code points to signal the payload carried in addition to control bits in the form of flags.
2. The service path header is the forwarding state carried in NSH and consists of a "Service Path ID" (SPI) and a "Service Index" (SI).
3. The context headers carry the metadata produced or consumed by the SFC infrastructure or the service functions.

Figure 1 reproduces the base and service path headers from [[I-D.ietf-sfc-nsh](#)], which are relevant to this discussion.

```

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 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Service Path ID (SPI)                               | Service Index |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Figure 1: NSH Base and Service Path Headers

[3.2.](#) NSH Forwarding

Traffic requiring servicing is forwarded on a network overlay constructed using NSH and an outer transport. The forwarding of traffic on this overlay is specified by the NSH. In particular, NSH asserts the following as described in the NSH Actions section of [[I-D.ietf-sfc-nsh](#)].

1. Mandates service functions to update the service index (SI).
2. Mandates the service function forwarders to make forwarding decisions based on the contents in the service path header,

namely SPI and SI.

These assertions essentially require the modification of the service path header in NSH by the SFs. The SFs are required to control the packet forwarding by making those decisions for the SFFs to use and forward NSH encapsulated packets on.

3.3. NSH Forwarding Shortcomings

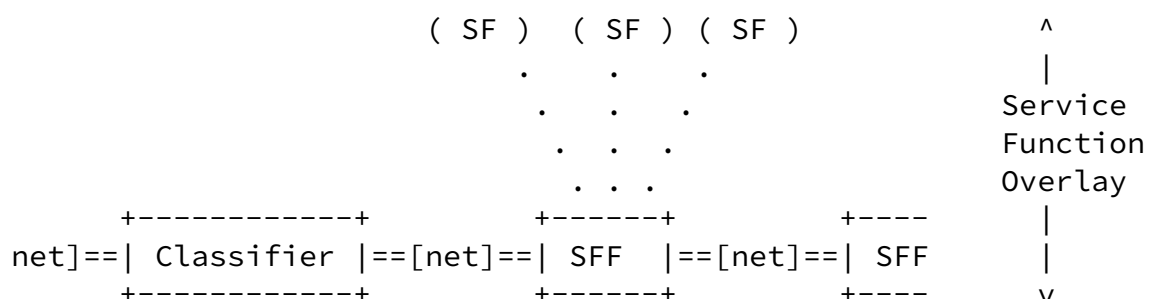
NSH's inability to separate packet forwarding and service delivery leads to the following disadvantages.

- o NSH is based on a model where SFs are fully trusted to maintain the integrity of the encapsulation, thereby allowing SFFs to forward on the decision made by SFs. However, this may not be acceptable in all network environments. Strict infrastructure and application boundaries in operators' environments essentially disallow such a method of packet forwarding.
- o Since forwarding decisions are made at SFs, including non-classifier SFs, by way of SI updates in NSH, the control plane has to program the SFs with information to aid such updates. This includes the SI updates corresponding to each SPI the SF belongs to. This approach impacts scalability as the number of SFs are significantly greater in number as compared to SFFs in a typical deployment.

- o Since non-classifier SFs require forwarding information programmed as described above and the SFs can be from any vendor or third party, with even their own control and management planes in some cases, programming SFs and SFC infrastructure leads to increased control plane complexity. This in turn impacts scalability of SFC deployment, weakening the SFC architecture.
- o Since service forwarding is required at the SFFs, which in turn is solely based on the SPI and SI fields in the NSH encapsulation header, it leaves SFFs vulnerable to forwarding on decisions not made by itself. These decisions are made by SFs as listed in the assertions. If an SF is buggy or compromised or doing incorrect manipulation of the service index, it may lead to many issues including, packet loop(when SF does not update SI), bypass SF(when SF over decrements or increments SI), etc.

- o Forwarding decisions at SFF cannot be avoided as many use cases require that decision to be performed in the SFF, making it inconsistent. For instance, when flows are offloaded to SFF by an SF, as described in [[I-D.kumar-sfc-offloads](#)] SFF MUST update the service path header as the SF will be bypassed.
- o Inspecting service path header in NSH on the wire, it is not possible to determine what service function the packet is associated with and where along the path it is at any moment, due to the fact that SFs update the SI and hence the Service Path Header. For instance, the SI inside NSH of a packet being serviced, points to one SF while inflight from SFF to SF and another when it is inflight from that same SF back to SFF. In other words, additional context is required to make such an assertion making troubleshooting cumbersome.

This section describes the separation of the forwarding and service delivery functions into separate abstract planes in the context of NSH but is generally applicable to the SFC architecture. Figure 2 depicts the separation of the service plane into service infrastructure and service function overlays. For the sake of simplicity SFC Proxy function is not shown as it is equivalent of an NSH aware SF in its handling of NSH encapsulation. The network connectivity between Classifier and SFF or SFFs or SFF and SF(or SFC Proxy) can be any network overlay over which NSH encapsulation can be transported, such as [[I-D.kumar-sfc-nsh-udp-transport](#)].



<-----Service Infrastructure----->
Overlay

Figure 2: SFC Network Overlay Separation

4.1. Forwarding Function And Service Function Separation

- o We propose the separation of forwarding and servicing planes in NSH encapsulation to render the SFC architecture cleanly. This enables forwarding from Classifier to SFF or one SFF to another or SFF to SF(or SFC Proxy) to be fully owned and controlled by the service chaining infrastructure while service delivery is the sole responsibility of SFs. This allows for scaling the service plane independent of the forwarding plane while avoiding forwarding conflicts that may otherwise arise. In other words, SFC forwarding is fully controlled by the SFFs and any forwarding-state carried in NSH, be in NSH service context header or meta-data context header, is fully opaque to the SFs.
- o We propose the overlay network be separated into infrastructure overlay and the service overlays as depicted in Figure 2. Infrastructure overlay extends between SFFs or Classifier and SFFs while the service overlay extends between SFFs and SFs.
- o We propose that only the SFFs and Classifiers update the service path header. This restriction limits the forwarding decisions to SFC infrastructure components. These steps make the service path header (or SPI and SI) opaque to SFs and immutable as it passes through an SF. Since SFs performing re-classification do so within the purview of the SFC control plane, re-classification SFs are an exception to maintaining this immutable property and are allowed to update the service path header.
- o We propose the update operation on service index in NSH service path header at the SFFs be controlled by the presence of a signal or a flag that indicates whether the packet is on the Infrastructure overlay or the Service overlay. [Section 4.2](#)

describes the allocation specifics in NSH to achieve this, which

is software as well hardware friendly.

- o We further propose that SFFs verify the integrity of the service path header every time a NSH packet is received from a SF. A simple approach to such verification is described in [Section 4.4](#).

4.2. NSH Infrastructure Flag

Figure 3 shows the format of the NSH encapsulation with the I flag in the 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|I|R|R|R|R|   Length   |   MD Type   | Next Protocol |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               | Service Index |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Figure 3: NSH Base and Service Path Header with 'I' Flag

I Bit: Infrastructure overlay flag

I = 1: Packet or frame is on the infrastructure overlay

I = 0: Packet or frame is on the service-function overlay

4.3. Rules For Infrastructure Flag Usage

The 'I' flag acts as a discriminator identifying the sender of the NSH encapsulated packet as SFC infrastructure component or service function. This becomes essential architecturally, as the same interface at a SFF may receive NSH encapsulated packets from both the SFC infrastructure components and the service functions.

The following rules MUST be observed by the SFC components in updating the 'I' flag and the service path header in NSH encapsulation header.

1. Classifier MUST set the 'I' flag to '1' when sending the NSH encapsulated packet or frame to the next SFF (or SFF)
2. SFF MUST set the 'I' flag to '1' when sending the NSH encapsulated packet to the next SFF

3. Classifier and SFF MUST set the 'I' flag to '0' in all other circumstances when forwarding NSH encapsulated packet
4. SF and SFC Proxy MUST NOT set the I flag
5. SFF MUST update the Service Index in NSH only when a packet with NSH is received with the 'I' flag set to '0' before making the next forwarding decision.

In addition to the above rules guiding the I flag usage, the following constraints must be met.

- o When more than one classifier exists in the deployment, all classifiers MUST adhere to the above rules.
- o Non-classifier SFs MUST NOT update the NSH service path header
- o Control plane or static configuration at SFs and SFFs (outside the scope of this draft) MUST control the use of I flag and the overall behavior described in this draft. This is recommended as the default behavior of SFFs and SFs.

[4.4.](#) Service Header Integrity Check

The separation of service function and forwarding function responsibilities with respect to forwarding state, allows the service function forwarders to enforce integrity checks to verify the immutable aspect of the service path header. Implementations are recommended to use an appropriate method to verify the integrity of the service path header.

There are many approaches to performing the integrity checks. Actual method is out of scope for this document. A few methods are briefly summarized here as mere examples and implementations must devise their own.

- o Every NSH packet received from a SF, 'I=0' in NSH base header, is checked against the three tuple: <SF-Transport-Address, SPI, SI> programmed in the SFF, by the control plane, for that SF. This method is simple and works well when a SF appears only once across all service paths.
- o SFFs compute a hash of a n-tuple or a pseudo header and transport this hash, as opaque metadata in NSH, through the SFs on a service path. When SFF receives the opaque metadata back, post servicing

of the packet, re-computes the hash of the same n-tuple and checks against the hash received in NSH. The n-tuple may include inner

payload, outer transport, service path header and SFF local data among others. Implementations must determine the n-tuple based on the SFC deployment requirements.

- o SFFs that are stateful, use flow state to record SPI and SIs and validate the same when the packet is received back from a SF. This works well as long as an SF appears only once in a given SPI. If multiple instances of the same SF within the same SPI is needed, additional check to protect the SI must be used.
- o As a generalized approach, control plane programs a mask to be applied to the NSH header to select the bits to perform integrity checks against. In the simplest case, the mask represents just the service path header.

These methods do not protect against threats such as packet replay or spoofing attacks, which do not violate the integrity of the service path header. These methods protect only against modification of the NSH service path header accidentally or otherwise thus ensuring the integrity of the same.

[4.5.](#) SF Considerations for Reverse Service Path Packets

Service functions are essentially applications servicing the traffic steered to them over NSH. Some service functions simply service the traffic received, by transmitting every packet along the path, in the same direction as received, after servicing it. Some service functions on the other hand, for instance service functions that act as proxies, often terminate the TCP flows locally before re-originating them towards the ultimate destination. Termination of a TCP flow locally at the SF requires completion of the TCP handshake, which further requires responding to the first sign of life packet or the TCP SYN packet.

SFs must be able to generate a NSH payload packet, in response to one received from a SFF, that flows in the opposite direction of the received payload packet. These response packets must thus traverse the service chain in the reverse direction of the one received from the SFF. However, NSH has provision to carry only one service path

in the service path header; a SFF cannot convey both the forward and the reverse SPIs to the SFs, to enable SFs to use the reverse SPIs in such scenarios. SFs that need to send a packet on the reverse service path must thus know how to fill the service path header with the correct SPI and SI. One approach is to have the control plane provision such information for SFs to use. However, this requires SFs to integrate with control plane leading to all the issues as discussed in [Section 3.3](#). Moreover, as discussed in this draft, SFs

benefit from focusing on service delivery while leaving the service forwarding decisions to SFFs.

This draft requires the service path header to be not updated by non-classifier SFs. In order to enable the SFs to send packets on reverse path while not modifying the service path header, we propose the SF request the SFF to move the packet to the appropriate service path. This is achieved by the use of the critical flag in NSH Base Header and a critical flag in the context header or TLV as shown in Figure 4 and Figure 5.

SF that wants to send a packet on the reverse path MUST insert a new CriticalFlags TLV and set the 'C' flag in the NSH Base Header to 1, in case of MD Type-2. The SF MUST set the 'B' flag to 1 to request forwarding of the packet on the reverse path.

SFF that receives a NSH packet with 'B' flag set to 1 in case of MD Type-1 or 'C' and 'B' flags set in case of MD Type-2 MUST transition the packet to the reverse path associated with the service path in the received NSH service path header. This transitioning involves SFF updating the NSH service path header to the right SPI and SI based on SFF configuration, policy or state.

```

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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|R|R|B|R|R|R|R|R| Context Header 1                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Context Header 2                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Context Header 3                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

|----- Context Header 4 -----|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

B : Backwards Flag;
 SF requests packet to be sent on the reverse service path

Figure 4: Reverse Path Request Messages with NSH Type-1

```

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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|      STANDARD CLASS      |1|CriticalFlags|0|0|0|    0x2    |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|B|                        Reserved                        |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

B : Backwards Flag;
 SF requests packet to be sent on the reverse service path

Figure 5: Reverse Path Request Messages with NSH Type-2

4.6. SF Considerations for Spontaneous Packets

SFs in most cases process and service NSH packets inline and in direction as a response to receiving the NSH packet from a SFF. However, some SFs generate packets in the middle of a flow spontaneously without receiving any NSH packet from a SFF. This is typical in SFs terminating TCP or proxies that need to act on a timer or an internal event.

In order for SFF to process these spontaneous packets, the SFs MUST encapsulate them in NSH, which in turn requires the service path header to be filled with the right SPI and SI.

Stateful SFs MUST cache the service path header in the flow state, received in each direction from the SFF, and use the appropriate cached service path header in the NSH encapsulation for sending spontaneous packets. SFs MUST treat the service path header as opaque metadata while caching or encapsulating with NSH.

SFs that have no flow-state MUST host a classifier or interact with one to obtain the right content for the service path header.

5. Infrastructure Forwarding Example

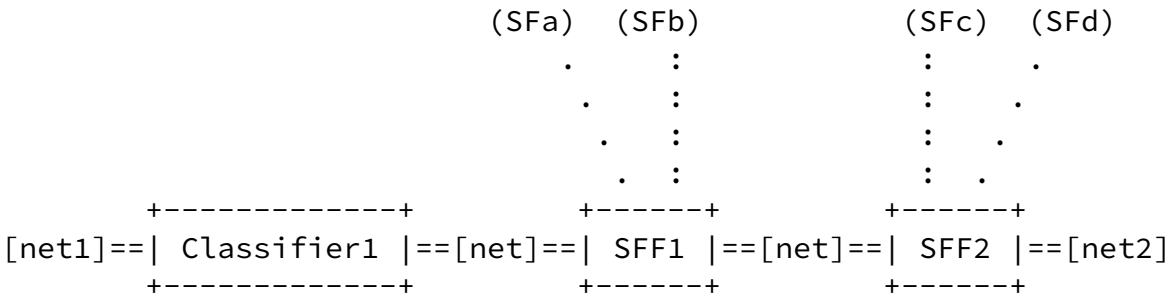


Figure 6: SFC Infrastructure Overlay Separation Example

This section outlines a typical packet flow to show the working of this behavior through an example SPI1 = SFa@SFF1,SFb@SFF1,SFc@SFF2 with the topology depicted in Figure 6.

1. Packet enters the system via the SFC ingress network (net1) reaching the classifier function
2. Classifier determines the SPI and SI as part of classification
3. Classifier formulates the NSH infrastructure overlay packet, sets the 'I' flag among other header updates and forwards it onwards to SFF1
4. SFF receives the NSH infrastructure overlay packet, skips the decrement operation due to I=1, performs a forwarding lookup to determine the next-hop
5. SFF1 determines SFa as the next-hop, formulates the NSH service

overlay packet, clears the 'I' flag among other header updates and forwards it onwards to SFa

6. SFa services the packet by consuming metadata or producing metadata and forwards the packet back to SFF1
7. SFF1 receives the NSH service overlay packet and decrements the SI, due to I=0, before performing a forwarding lookup
8. SFF1 determines the next-hop as SFb and the process repeats with SFb as before with SFa, with I=0
9. SFF1 receives the SFb serviced packet, decrements the SI due to I=0 and determines the next-hop to be SFC. It sets I=1 and forwards the packet to SFF2 on NSH infrastructure overlay.
10. SFF2 receives the packet from SFF1 and repeats the process through SFC similar to the steps for SFa performed by SFF1, by setting I=0
11. SFF2 receives the SFC serviced packet, decrements the SI due to I=0 and determines SPI1 is fully executed and proceeds with forwarding on the SFC egress network (net2)

6. Infrastructure Forwarding Advantages

The following are some of the benefits of separating the SFC overlay into infrastructure overlay and service function overlay.

- o Constrains the SFC forwarding decisions to SFFs where it belongs, providing meaning to the last 'F' in 'SFF'
- o Frees the SFs to focus purely on service delivery and avoid complexities associated with forwarding decisions
- o Enables validation of forwarding state carried in NSH, thereby maintaining the integrity of the forwarding state used to forward the packets along the service path. This removes issues arising from incorrect updates to service path header by SFs, accidentally or otherwise

- o Allows the service index in NSH packet to be always associated with the service function as indicated by the service index, whether the packet is in transit from SFF to the SF or from SF to the SFF
- o Allows additional security policies to be enforced between the infrastructure and the service functions by the network operators
- o Allows snooping tools or any type of middle boxes to clearly tell whether the NSH encapsulated packet is going between SFFs or between SFF and SF (without relying on the source and destination locators), due to the 'I' flag, which is useful in tracing and debugging, especially in cloud deployments
- o Allows the service chaining control plane to scale independent of the number of service functions

7. Acknowledgements

The authors would like to thank Abhijit Patra for his guidance and Mike Leske for his review comments.

8. IANA Considerations

IANA is requested to allocate a "STANDARD" class from the TLV Class registry as already requested in [[I-D.kumar-sfc-offloads](#)].

IANA is requested to allocate TLV type with value 0x2 from the STANDARD class TLV registry. The format of the "CriticalFlags" TLV is as defined in this draft, which can further be extended to define new flags in other drafts.

TLV#	Name	Description	Reference
2	Critical	Flags that are critical to SFC	This

	Flags	functionality	document

Table 1: New TLV in Standard Class Registry

9. Security Considerations

Separating forwarding decisions from service functions allows for additional constraints to be enforced by the infrastructure controlling the forwarding decisions. This separation enables additional security methods in the infrastructure and does not itself mandate any new security considerations.

10. References

10.1. Normative References

[I-D.ietf-sfc-nsh]

Quinn, P. and U. Elzur, "Network Service Header", [draft-ietf-sfc-nsh-05](#) (work in progress), May 2016.

[I-D.kumar-sfc-offloads]

Surendra, S., Guichard, J., Quinn, P., and J. Halpern, "Service Function Simple Offloads", [draft-kumar-sfc-offloads-02](#) (work in progress), March 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

10.2. Informative References

[I-D.ietf-sfc-architecture]

Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", [draft-ietf-sfc-architecture-11](#) (work in progress), July 2015.

[I-D.kumar-sfc-nsh-udp-transport]

Surendra, S., Kreeger, L., Majee, S., Haeffner, W., Manur, R., and D. Melman, "UDP Transport for Network Service Header", [draft-kumar-sfc-nsh-udp-transport-02](#) (work in progress), February 2016.

Authors' Addresses

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

Email: smkumar@cisco.com

Kent Leung
Cisco Systems, Inc
170 W. Tasman Dr.
San Jose, CA 95134
US

Email: kleung@cisco.com

Peter Bosch
Cisco Systems, Inc
Haarlerbergpark Haarlerbergweg 13-19
Amsterdam, NOORD-HOLLAND 1101 CH
Netherlands

Email: pbosch@cisco.com

Dongkee Lee
SK Telecom
9-1 Sunae-dong, Pundang-gu
Sunnam-si, Kyunggi-do
South Korea

Email: dongkee.lee@sk.com

Rajeev Manur
Broadcom

Email: rmanur@broadcom.com

Internet-Draft

NSH Service Forwarding

August 2016

Andrew Dolganow
Nokia Corp
Block 750 Dune, #06-06 Chai Chee Road
Technopark@Chai Chee, Singapore 469004
Singapore

Email: andrew.dolganow@nokia.com

Praveen Muley
Nokia Corp

Email: praveen.muley@nokia.com

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

Email: S.Majee@F5.com

Joel Halpern
Ericsson

Email: joel.halpern@ericsson.com

Kumar, et al.

Expires February 20, 2017

[Page 17]