Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 18, 2014

P. Quinn
J. Guichard
R. Fernando
S. Kumar
M. Smith
N. Yadav
Cisco Systems, Inc.
P. Agarwal
R. Manur
Broadcom
A. Chauhan
Citrix
U. Elzur
Intel
B. McConnell
Rackspace
C. Wright
Red Hat Inc.

February 14, 2014

# Network Service Header
# draft-quinn-sfc-nsh-02.txt

Abstract

This draft describes a Network Service Header (NSH) added to encapsulated packets or frames to realize service function paths. NSH also provides a mechanism for metadata exchange along the instantiated service path.

## 1. Requirements Language

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

Status of this Memo



Copyright Notice

Table of Contents

## 2. Introduction

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

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

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

The approach taken by NSH is composed of two elements:

1. Fixed size, transport independent per-packet/frame service meta-data

2. Data plane encapsulation that utilizes the network overlay topology used to deliver packets to the requisite services.

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 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 (SFCNF): 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 SFCNF to one or more connected service functions, and from service functions to the SFCNF.

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, including under and overlays. The coupling of service functions to topology requires service functions to support many transports 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.

A NSH is composed of a 64-bit base header, and four 32-bit context headers as shown in figure 1 below.

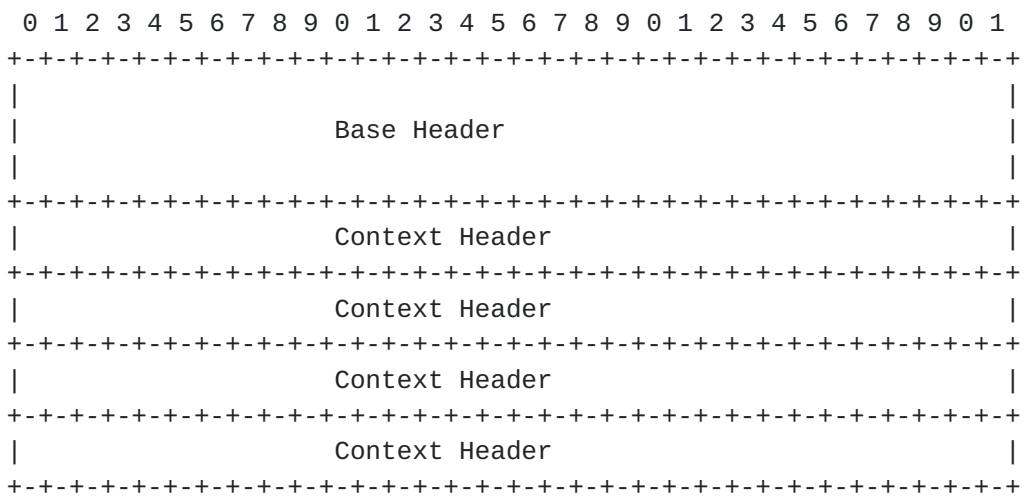



Figure 1: Network Service Header

Base header: provides information about the service header and service path identification.

Context headers: carry opaque metadata.

### 3.1. 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 chain, an SF, or an

associated 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.

## 3.2. 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 4 for an example using GRE and NSH encapsulation.

### 3.3. 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 (see section 4 for header specifics). 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.

### 3.4. 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.

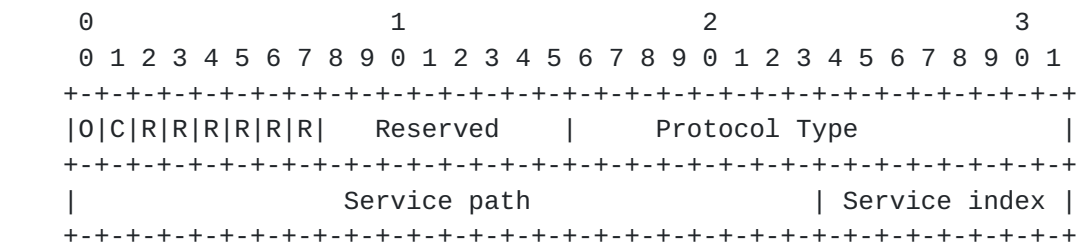
NSH is re-imposed on packets returned to the proxy from the non-NSH aware service.

Typically, a SFCNF will act as a NSH-proxy when required.

An NSH proxy MUST perform NSH actions as described in section 3.1.

## 4. Header Format

Base Service Header:




Flags: 8
Reserved: 8
Protocol Type (PT): 16
Service path (SP): 24
Service index (SI): 8

Figure 2: NSH Base Header

Base Header Field Descriptions

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 (i.e. return status information).

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

C bit: Context headers MUST be present. When C is set, one or more contexts are in use (i.e. a value placed in a context is significant). The C bit specifies that their ordering and sizing is as per figure 4: network platform (32 bits), network shared (32 bits), service platform (32 bits), service shared (32 bits).

A C bit equal to zero indicates that no contexts are in use (although they MUST be present to ensure a fixed size header) and that they can be ignored.

If a context header is not in use, the value of that context header MUST be zero.

All other flag fields are reserved.

Protocol type: indicates the protocol type of the original packet or frame as per [ETYPES]

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.

Service Path: 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.

Context Headers:

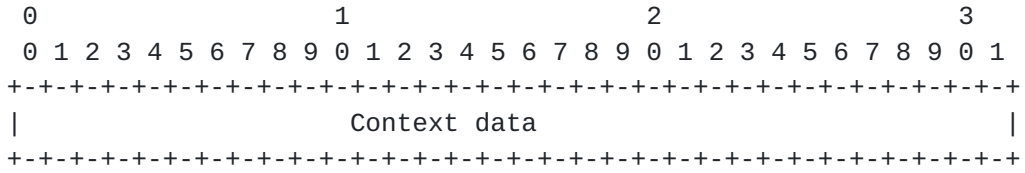



Figure 3: Context Data

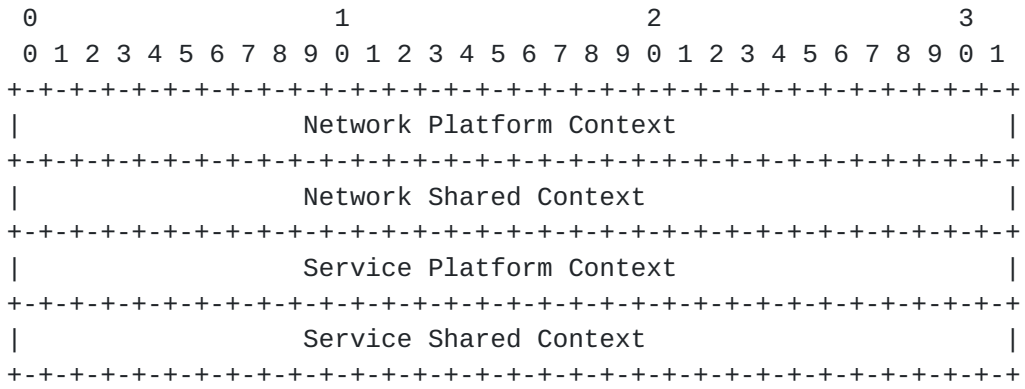



Figure 4: Context Data Significance

The following examples of context header allocation are guidelines that illustrate how various forms of information can 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.

## 5. NSH Example: GRE

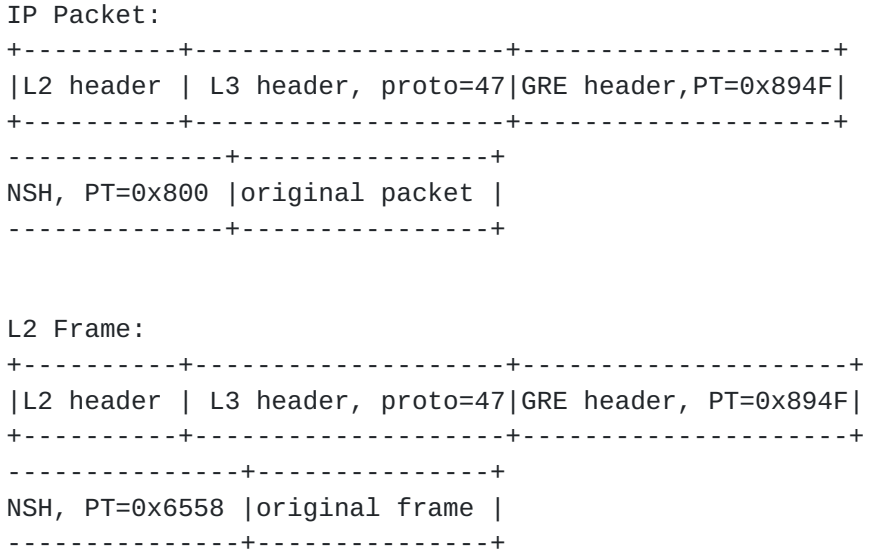



Figure 5: GRE + NSH

## 6. 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.

## 7. Acknowledgments

The authors would like to thank Nagaraj Bagepalli, Abhijit Patra, Carlos Pignataro, Ron Parker, Peter Bosch, Tom Nadeau, 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.

## 8. IANA Considerations

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

# 9. References

## 9.1. 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.

## 9.2. Informative References

[ETYPES] The IEEE Registration Authority, "IEEE 802 Numbers", 2012, <http://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xml>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[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 A. Beliveau, Ed., "Service Function Chaining (SFC) Architecture", 2014, <http://datatracker.ietf.org/doc/draft-quinn-sfc-arch>.

Authors' Addresses

Paul Quinn
Cisco Systems, Inc.

Email: paulq@cisco.com

Jim Guichard
Cisco Systems, Inc.

Email: jguichar@cisco.com

Rex Fernando
Cisco Systems, Inc.

Email: rex@cisco.com

Surendra Kumar
Cisco Systems, Inc.

Email: smkumar@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

Abhishek Chauhan
Citrix

Email: Abhishek.Chauhan@citrix.com

Uri Elzur
Intel

Email: uri.elzur@intel.com

Brad McConnell
Rackspace

Email: bmcconne@rackspace.com

Chris Wright
Red Hat Inc.

Email: chrisw@redhat.com