Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 13, 2014

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

July 12, 2013

Network Service Header
draft-quinn-nsh-01.txt

Abstract

This draft describes a Network Service Header (NSH) that can be added to encapsulated network packets or frames to create network service paths. In addition to path information, this header also carries metadata used by network devices and/or network services.

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 is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on January 13, 2014.

Copyright Notice

Copyright (c) 2013 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

2. Introduction

Network services are widely deployed and essential in many networks. The services provide a range of functions such as security, WAN acceleration, and server load balancing. Service functions that form part of the overall service may be physically located at different points in the network infrastructure such as the wide area network, data center, campus, and so forth.

The current network service 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 network service 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 forwarding elements.

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

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.

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 Chain: A service chain defines the services required (e.g. FW), and their order (service1 --> service2) that must be applied to packets and/or frames. Service chains need not be linear, rather they are represented by a graph topology.

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 (tracked in an NSH).

Service Path: Forwarding path used for delivery of traffic along a service chain. A service path is a series of service hops.

Service Node: Physical or virtual element providing one or more service functions. Service nodes utilize NSH for path and other information.

Service Path Segment: A segment of a service path between two service nodes.

Network Service: An externally visible service offered by a network operator; a service may consist of a single service function or a composite built from several service functions executed in one or more pre-determined sequences.

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

Service Function: A service function (NAT, FW, DPI, IDS, application based packet treatment), application, compute resource, storage, or content used singularly or in collaboration with other service functions to enable a service offered by a network operator.

2.2. Problem Statement

Network Service Header (NSH) addresses several limitations associated with network service deployment today:

1. Topological Dependencies: Network service deployments are often coupled to the physical network topology creating artificial constraints on delivery. These topologies serve only to "insert" the service function; they are not required from a native packet delivery perspective. For example, firewalls often require an "in" and "out" L2 segment and adding a new firewall requires changing the topology (i.e. adding new L2 segments).

   As more services are required - often with strict ordering - topology changes are needed before and after each service resulting in complex network changes and device configuration. In such topologies, all traffic, whether a service needs to be applied or not, will often pass through the same strict order. A common example is web servers using a server load balancer as the default gateway. When the web service responds to non-load balanced traffic (e.g. administrative or backup operations) all traffic from the server must traverse the load balancer forcing network administrators to create complex routing schemes or create additional interfaces to provide an alternate topology.

2. Service Chaining: Service functions are typically independent; service function_1 (Sf1)...service function_n (SFn) are unrelated and there is no notion at the service layer that Sf1 occurs before Sf2. However, to an administrator many service functions have a strict ordering that must be in place, yet have no consistent way to impose and verify the deployed service ordering.

   Service chains can be unidirectional, bidirectional, symmetric or asymmetric.

3. Service Policy Application: Service functions rely on either topology information such as VLANs or packet (re)classification to determine service policy selection, the service action taken. Topology information is increasingly less viable due to scaling, tenancy and complexity reasons. Per-service function packet classification is inefficient and prone to errors, duplicating functionality across services. Furthermore packet classification is often too coarse lacking the ability to determine class of traffic with enough detail.

4. Per-Service (re)Classification: Classification occurs at each service, independently from previous service functions. These

unrelated classification events consume resources per service. More importantly, the classification functionality often differs per service and services cannot leverage the results from other deployed network or service.

5. Elastic Service Delivery: Given the current state of the art for adding/removing services largely centers around VLANs and routing changes, rapid changes to the service layer can be hard to realize due to the risk and complexity of such changes.

6. 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.

7. Limited End-to-End Service Visibility: Troubleshooting service related issues is a complex process that involve network and service expertise.

8. Transport Agnostic: Services can and will be deployed in networks with a range of transports, including under and overlays. The coupling of services to topology requires services to support many transports or for a transport gateway function to be present.

3. Network Service Header

A Network Service Header (NSH) is metadata added to a packet or frame that is 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.

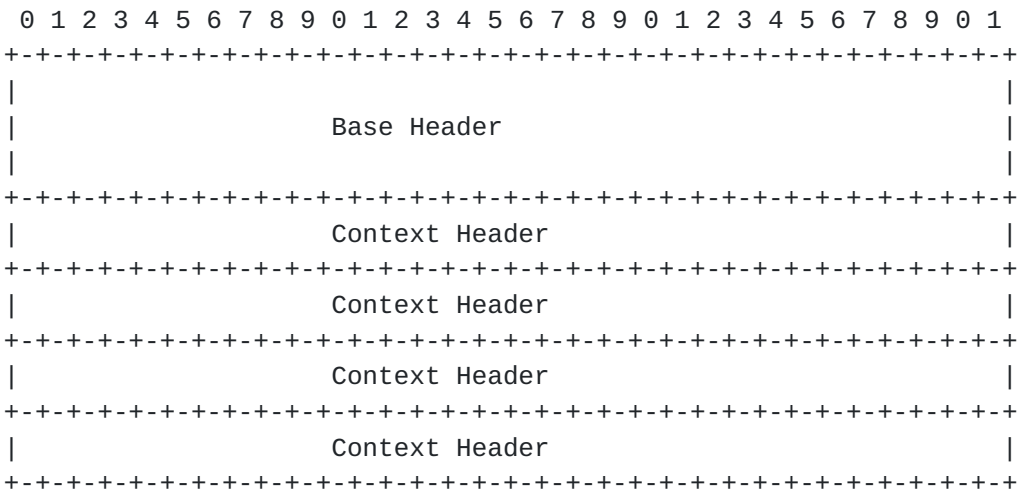


Figure 1: Network Service Header

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, services nodes, NSH proxies and forwarding elements in the service plane, have several possible header related actions:

1. Insert/remove service header: These actions can occur at the start and end respectively of a service path or can be performed by a service function that determines that a service path must change due to local policy. Data is classified, and if

determined to require servicing, a service header imposed.

A service function can re-classify data as required. A service classifier MUST insert a NSH. At the end of a service chain, the last node operating on the service header MUST remove it. If a node performs re-classification that requires 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.

The last node is signaled via the control plane, i.e. the next-hop communicated by a control plane to the last node is "end of chain".

2. Forward based on header fields: The base header provides service chain information and is used by participating nodes to determine correct service path selection and forwarding as well as loop detection. Participating nodes MUST use the base header for selecting the next service in the service path.

3. Update a service header: Services MUST decrement the service index. A service index = 0 indicates that a packet MUST be dropped by the node 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 service for NSH actions), then the proxy MUST update service index and MAY update contexts.

4. Service policy selection: Service 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 the metadata 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. Non-participating 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 in the outer encapsulation or, in the case of MPLS, the presence of the GAL label as defined in [draft-guichard-mpls-metadata-00].

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. Services have a locator (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 chain and the services within that chain.

2. Service Chaining: NSH contains forwarding 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 service chain information can be used by administrators (via, for example a traffic analyzer) to verify (account, ensure correct chaining, provide reports, etc.) the chain specifics of packets being forwarded along a service chain.

3. Metadata Sharing: NSH provides a mechanism to carry shared metadata between network devices and services, and between services. 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 nodes, an NSH proxy is used. The proxy node removes the NSH header and delivers, to the service node, the payload packet/frame via a local attachment circuit. Examples of a local attachement circuit include, but aren not limited to: VLANs, IP in IP, GRE, VXLAN. When complete, the service return

the packet to the NSH-proxy via the same or different attachment circuit.

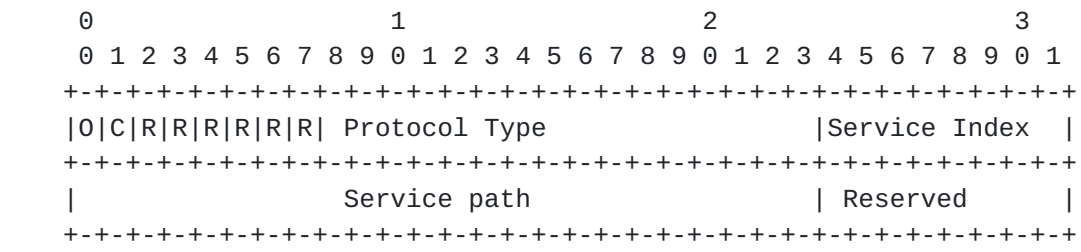
NSH is re-imposed on packets returned to the proxy from the non-NSH aware service.

For example a network node -- physical or virtual -- attached to the non-NSH service provides proxy functionality

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

4. Header Format

Base Service Header:



Flags: 8
Protocol Type (PT): 16
Service Index: 8
Service path: 24
Reserved: 8

Figure 2: NSH Base Header

Base Header Field Descriptions

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

OAM message spefifics and handling details are outside the scope of this document and will be adressed in a future draft.

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: TTL functionality and location within the service path. Service index MUST be decremented by service nodes 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 path and indicates the location of a packet in a service chain.

Service Path: identifies a service path. Participating node 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:

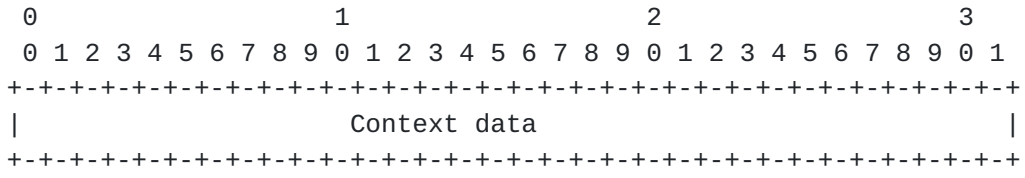


Figure 3: Context Data

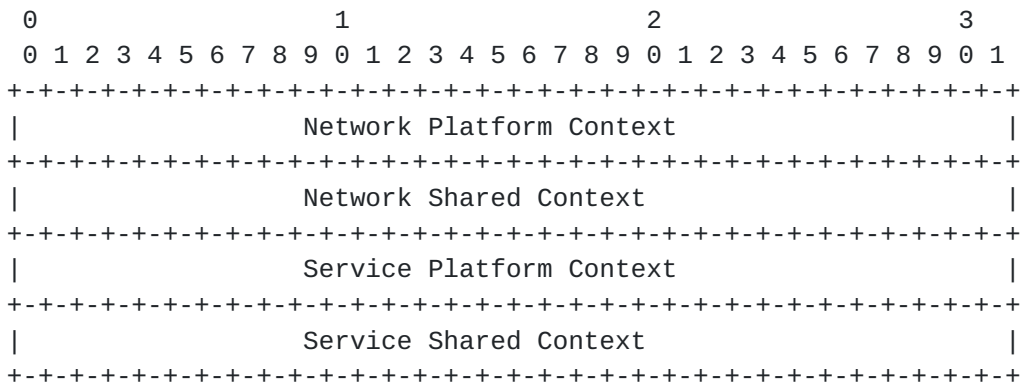


Figure 4: Context Data Significance

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 indentifier used for load balancing decisions.

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

5. NSH Example: GRE

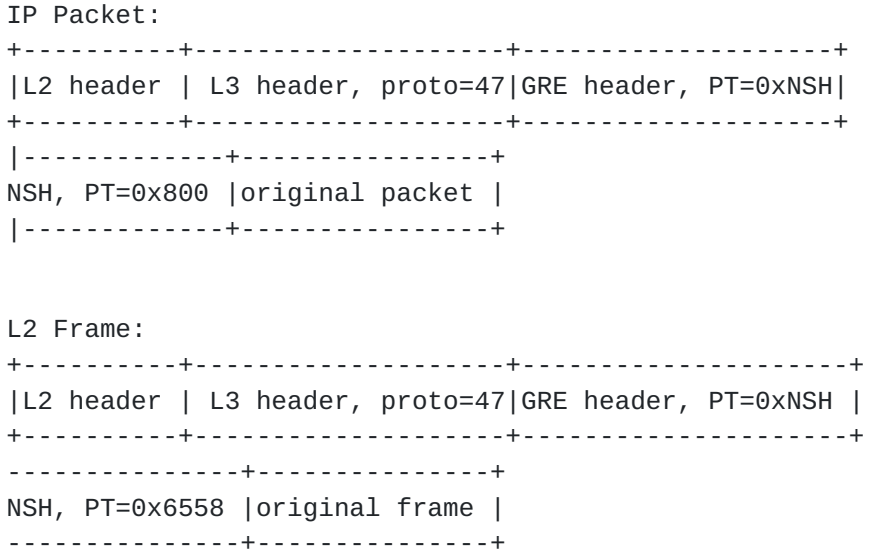


Figure 5: GRE + NSH

Note: 0xNSH is a placeholder for a NSH specific Ethertype that will be requested.

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. [IPSec]) 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 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 will be requested 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.

Authors' Addresses

Paul Quinn
Cisco Systems, Inc.

Email: paulq@cisco.com

Rex Fernando
Cisco Systems, Inc.

Email: rex@cisco.com

Jim Guichard
Cisco Systems, Inc.

Email: jguichar@cisco.com

Surendra Kumar
Cisco Systems, Inc.

Email: smkumar@cisco.com

Puneet Agarwal
Broadcom

Email: pagarwal@broadcom.com

Rajeev Manur
Broadcom

Email: rmanur@broadcom.com

Abhishek Chauhan
Citrix

Email: Abhishek.Chauhan@citrix.com

Michael Smith
Insieme Networks

Email: michsmit@insiemenetworks.com

Navindra Yadav
Insieme Networks

Email: nyadav@insiemenetworks.com

Brad McConnell
Rackspace

Email: bmcconne@rackspace.com

Chris Wright
Red Hat Inc.

Email: chrisw@redhat.com