Service Function Chaining
Internet-Draft
Intended status: Informational
Expires: April 4, 2015

S. Kumar
J. Guichard
P. Quinn
Cisco Systems, Inc.
J. Halpern
Ericsson

Oct 2014

Service Function Path Optimization
draft-kumar-sfc-sfp-optimization-01

Abstract

Service Function Chaining (SFC) enables services to be delivered by selective traffic steering through an ordered set of service functions. Once classified into an SFC, the traffic for a given flow is steered through all the service functions of the SFC for the life of the traffic flow even though this is often not necessary. Steering traffic to service functions only while required and not otherwise, leads to optimal SFCs with improved latencies, reduced resource consumption and better user experience.

This document describes the rationale, techniques and necessary protocol extensions to achieve such optimization.

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 April 4, 2015.

Copyright Notice

Table of Contents

1. Introduction

Service function chaining involves steering traffic flows through a set of service functions in a specific order. Such an ordered list of service functions is called a Service Function Chain (SFC). The actual forwarding path used to realize an SFC is called the Service Function Path (SFP).

Service functions forming an SFC are hosted at different points in the network, often co-located with different types of service functions to form logical groupings. Applying a SFC thus requires traffic steering by the SFC infrastructure from one service function to the next until all the service functions of the SFC are applied. Service functions know best what type of traffic they can service and how much traffic needs to be delivered to them to achieve complete delivery of service. As a consequence any service function may potentially request, within its policy constraints, traffic no longer be delivered to it or its function be performed by the SFC infrastructure, if such a mechanism is available.

This document outlines mechanisms to not steer traffic to service functions, on request, while still ensuring compliance to the instantiated policy that mandates the SFC.

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 the following terms. Additional terms are defined in [I-D.ietf-sfc-problem-statement], [I-D.ietf-sfc-architecture] and [I-D.quinn-sfc-nsh]. Some are reproduced here only for convenience and the reader is advised to consult the referenced documents.

Service Function (SF): A function that is responsible for specific treatment of received packets. A Service Function can act at the network layer or other OSI layers. A Service Function can be a virtual instance or be embedded in a physical network element. One of multiple Service Functions can be embedded in the same network element. Multiple instances of the Service Function can be enabled in the same administrative domain. A non-exhaustive list of Service Functions includes: firewalls, WAN and application acceleration, Deep Packet Inspection (DPI), server load balancers, NAT44 [RFC3022], NAT64 [RFC6146], HOST_ID

injection, HTTP Header Enrichment functions, TCP optimizer, etc.

Service Node (SN): A virtual or physical device that hosts one or more service functions, which can be accessed via the network location associated with it.

Service Function Forwarder (SFF): A service function forwarder is responsible for forwarding traffic along the service path, which includes delivery of traffic to the connected service functions.

Network Forwarder (NF): The entity, typically part of the network infrastructure, responsible for performing the transport function in traffic forwarding.

Service Controller (SC): The entity responsible for managing the service chains, including create/read/update/delete actions as well as programming the service forwarding state in the network - SFP distribution.

Classifier (CF): The entity, responsible for selecting traffic as well as SFP, based on policy, and forwarding the selected traffic on the SFP after adding the necessary encapsulation. Classifier is implicitly an SFF.

Offload: A request or a directive from the SF to alter the SFP so as to remove the requesting SF from the SFP while maintaining the effect of the removed SF on the offloaded flow.

Un-offload: A request or directive to cancel the effect of Offload - leads to altering the SFP so as to insert the requesting SF back into the SFP and steer the flow to it.

3. Service Function Path Optimization

The packet forwarding path of a SFP involves the classifier, one or more SFFs and all the SFs that are part of the SFP. Packets of a flow are forwarded along this path to each of the SFs, for the life of the flow, whether SFs perform the full function in treating the packet or reapply the cached result, from the last application of the function, on the residual packets of the flow. In other words, every packet on the flow incurs the same latency and the end-to-end SFP latency remains more or less constant subject to the nature of the SFs involved. If an SF can be removed from the SFP, for a specific flow, traffic steering to the SF is avoided for that flow; thus leading to a shorter SFP for the flow. When multiple SFs in a SFP are removed, the SFP starts to converge towards the optimum path, which in its best case starts and terminates at the classifier itself

without incurring any latency associated with traversing the SFP.

Although SFs are removed from the SFP, the corresponding SFC is not changed - this is subtle but an important characteristic of this mechanism. In other words, this mechanism does not alter the SFC and still uses the SFP associated with the SFC.

There are two primary approaches to removing an SF from the SFP. Namely,

- Bypass: Mechanism that alters the SFC. Described in this draft for completeness.

- Simple Offload: Mechanism that alters the SFP alone, does not affect the SFC. This is the primary focus of this draft.

3.1. Bypass

Many service functions do not deliver service to certain types of traffic. For instance, typical WAN optimization service functions are geared towards optimizing TCP traffic and add no value to non-TCP traffic. Non-TCP traffic thus can bypass such a service function. Even in the case of TCP, a WAN optimization SF may not be able to service the traffic if the corresponding TCP flow is not seen by it from inception. In such a situation a WAN optimization SF can avoid the overhead of processing such a flow or reserving resources for it, if it had the ability to request such flows not be steered to it. In other words such service functions need the ability to request they be bypassed for a specified flow from a certain time in the life of that flow.

A seemingly simple alternative is to require service functions pre specify the traffic flow types they add value to, such as the one-tuple: IP protocol-type described above. A classifier built to use such data exposed by SFs, may thus enable bypassing such SFs for specific flows by way of selecting a different SFC that does not contain the SF being removed.

Although knowledge of detailed SF profiles helps SFC selection at the classifier starting the SFC, it leads to shortcomings.

- It adds to the overhead of classification at that classifier as all SF classification requirements have to be met by the classifier.

- It leads to conflicts in classification requirements between the classifier and the SFs. Classification needs of different SFs in the same SFC may vary. A classifier thus cannot classify traffic

based on the classification of one of the SFs in the chain. For instance, even though a flow is uninteresting to one SF on an SFC, it may be interesting to another SF in the same SFC.

- o The trigger for bypassing an SF may be dynamic as opposed to the static classification at the classifier - it may originate at the SFs themselves and involve the control and policy planes. The policy and control planes may react to such a trigger by instructing the classifier to select a different SFC for the flow, thereby achieving SF bypass.

3.2. Simple Offload

Service delivery by a class of service functions involves inspecting the initial portion of the traffic and determining whether traffic should be permitted or dropped. In some service functions, such an inspection may be limited to just the five tuple, in some others it may involve protocol headers, and in yet others it may involve inspection of the byte stream or application content based on the policy specified. Firewall service functions fall into such a class, for example. In all such instances, servicing involves determining whether to permit the traffic to proceed onwards or to deny the traffic from proceeding onwards and drop the traffic. In some cases, dropping of the traffic may be accompanied with the generation of a response to the originator of traffic or to the destination or both. Once the service function determines the result - permit or deny (or drop), it simply applies the same result to the residual packets of the flow by caching the result in the flow state.

In essence, the effect of service delivery is a PERMIT or a DENY action on the traffic of a flow. This class of service functions can avoid all the overhead of processing such traffic at the SF, by simply requesting another entity in the SFP, to assume the function of performing the action determined by the service function. Since PERMIT and DENY are very simple actions other entities in the SFP are very likely to be able to perform them on behalf of the requesting SF. A service function can thus offload simple functions to other entities in the SFP.

Since SFF is the one steering traffic to the SFs and hence is on the SFP, is a natural entity to assume the offload function. An SF not interested in traffic being steered to it can simply perform a simple offload by indicating a PERMIT action along with an OFFLOAD request. The SFF responsible for steering the traffic to the SF takes note of the ACTION and offload request. The OFFLOAD directive and the ACTION received from the requesting SF are cached against the SF for that flow. Once cached, residual packets on the flow are serviced by the cached directive and action as if being serviced by the corresponding

SF.

3.2.1. Stateful SFF

SFFs are the closest SFC infrastructure entities to the service functions. SFFs may be state-full and hence can cache the offload and action in both of the unidirectional flows of a connection. As a consequence, action and offload become effective on both the flows simultaneously and remain so until cancelled or the flow terminates.

SFFs may not always honor the offload requests received from SFs. This does not affect the correctness of the SFP in any way. It implies that the SFs can expect traffic to arrive on a flow, which it offloaded, and hence must service them, which may involve requesting an offload again. It is natural to think of an acknowledgement mechanism to provide offload guarantees to the SFs but such a mechanism just adds to the overhead while not providing significant benefit. Offload serves as a best effort mechanism.

3.2.2. Packet Re-ordering

Offload mechanism creates short time-windows where packet re-ordering may occur. While SFs request flows be offloaded to SFFs, packets may still be in flight at various points along the SFP, including some between the SFF and the SF. Once the offload decision is received and committed into the flow entry at the SFF, any packets arriving after and destined to the offloading SF are treated to the offload decision and forwarded along (if it is a PERMIT action). Inflight packets to the offloading SF may arrive at the SFF after one or more packets are already treated to the offload decision and forwarded along.

This is a transitional effect and may not occur in all cases. For instance, if the decision to offload a flow by an SF is based on the first packet of TCP flow, a reasonable time window exists between the offload action being committed into the SFF and arrival of subsequent packet of the same flow at that SFF. Likewise, request/response based protocols such as HTTP may not always be subject to the re-ordering effects.

3.2.3. Policy Implications

Offload mechanism may be controlled by the policy layer. The SFs themselves may have a static policy to utilize the capability offered by the SFC infrastructure. They could also be dynamic and controlled by the specific policy layer under which the SFs operate.

Similarly, the SFC infrastructure, specifically the classifiers and

the SFFs, may be under the SFC infrastructure control plane policy controlling the decision to honor offloads from an SF. This policy in turn may be coarse-grain, at the SF level, and hence static. It can also be fine grain and hence dynamic but it adds to the overhead of policy distribution.

Policy model related to offloads is out of scope of this document.

3.2.4. Capabilities Exchange

Simple offloads can be exposed and negotiated a priori as a capability between the SFFs and the SFs or the corresponding control layers. In the simplest of the implementations, this is provided by the SFC infrastructure and the SFs are statically configured to utilize them without capabilities negotiation, within the constraints of the SF specific policies.

Capabilities exchange is outside the scope of this document.

4. Methods For SFP Optimization

There are a number of different models that may be used to facilitate shortest SFP realization. We present two here.

The shortest SFP methods discussed in the following sections require signaling among the participant components to communicate offload and permit/deny actions. The signaling may be performed in the data-plane or in the control plane.

a. Data-plane: An SFC specific communication channel is needed for SNs to communicate the offload request along with the SF treated packet. [NSH] defines a header specifically for carrying SFP along with metadata and provides such a channel for use with offloads. Necessary bits need to be allocated in NSH to convey the action as well as the offload directive. This signaling may be limited to SN and SFF or may continue from one SFF to another SFF or the classifier. It may also involve signaling directly from the SF to the classifier.

b. Control-plane: Messages are required between the SN and the service controller as well as between the SFF and the control plane. Service controller messaging is out of scope of this document and it is assumed to be service controller specific.

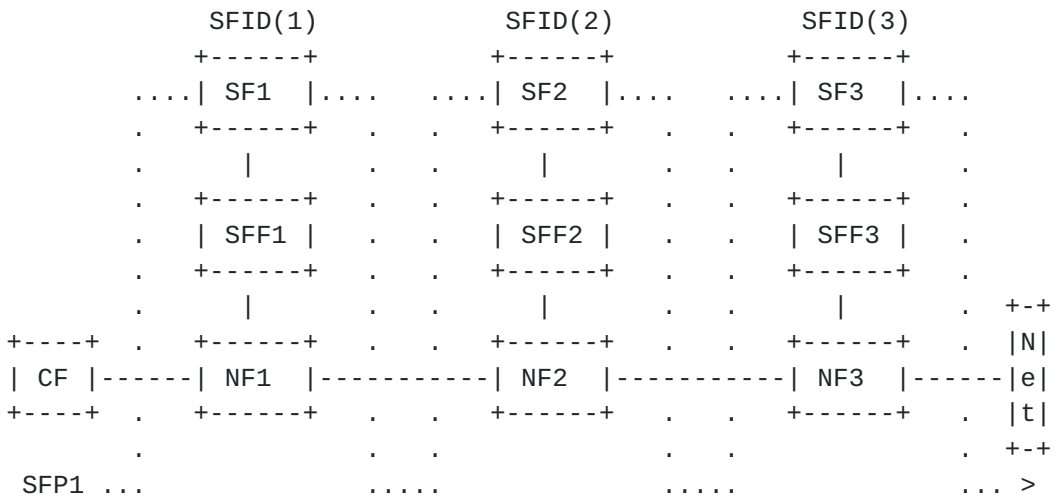
4.1. Hop-by-hop Offload

SNs receive traffic on an overlay from the SFF. SNs service the traffic and turn them back to the SFF on an overlay or forward the traffic on the underlay. In the former case, along with returning the traffic to SFF, they can perform simple offload by signaling OFFLOAD and ACTION to the SFF. SFF caches the OFFLOAD and ACTION while forwarding the serviced packet onwards to the next service hop on the SFP or dropping it. SFF can now enforce the OFFLOAD and ACTION on the residual packets of the flow.

Additionally, SFF may choose to signal the upstream SFFs of the OFFLOAD and ACTION received from an SF. This may continue recursively until the first SFF is reached, which is the classifier itself.

By performing such hop-by-hop offloads, SFP can be reduced to an optimum one, steering traffic to only those SFs that really need to see the traffic.

Figure 1 to Figure 3 show an example of SF and SFF performing an offload operation and the effect thereafter on the SFP.



Service Function Chain, SFC1 = {SF1, SF2, SF3}
where SF1, SF2 and SF3 are three service functions.
Service Function Path SFP1 is the SFP for SFC1.
Classifier CF starts SFP1 based on policy.

Figure 1: SFC1 with corresponding SFP1

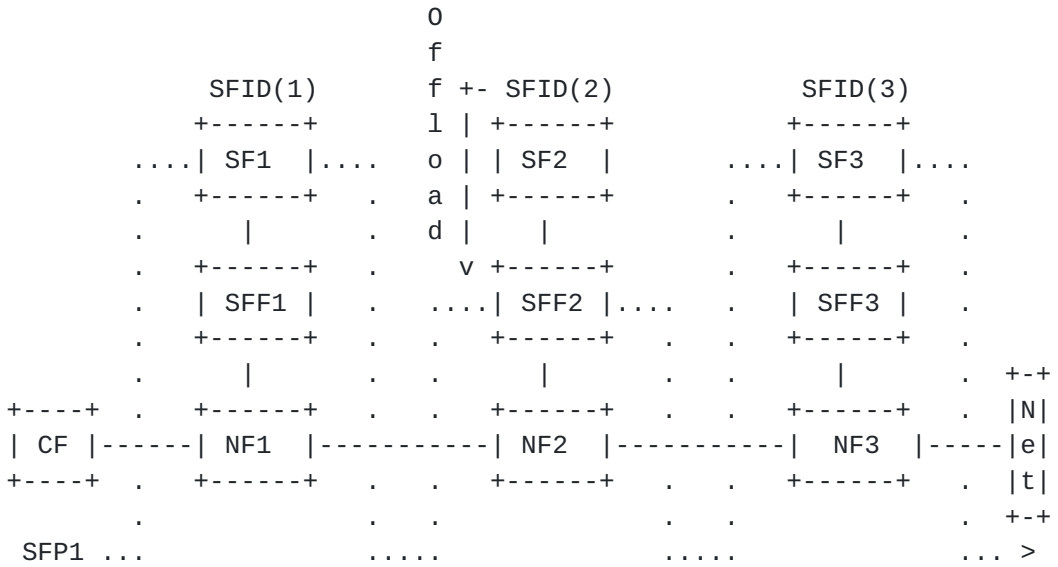


Figure 2: SFP1 after SFID(2) performs an Offload

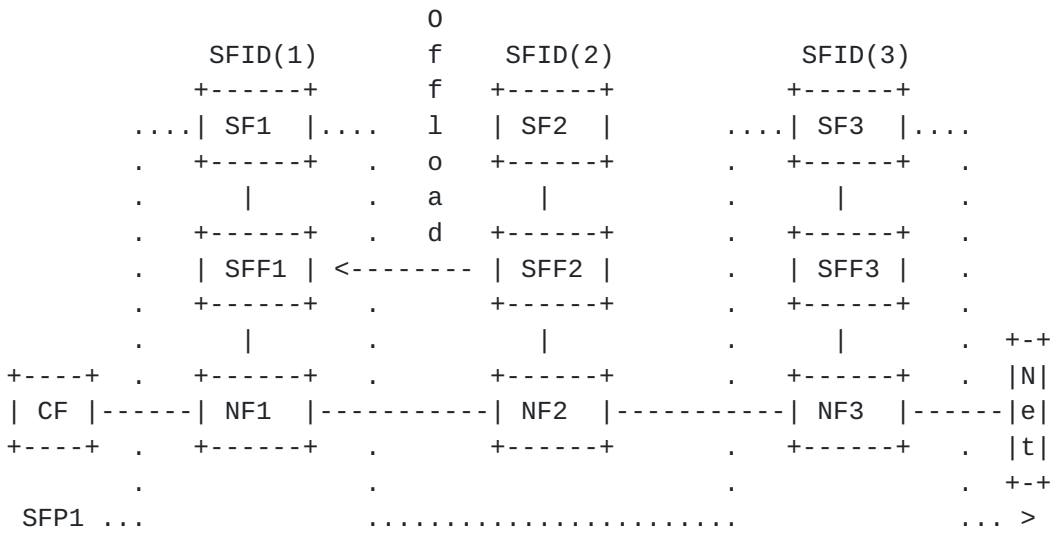


Figure 3: SFP1 after SFF2 propagates the Offload upstream to SFF1

4.1.1. Progression Of SFP Optimization

SFP optimization happens at two levels:

Level-1: Collapsing of the SFF-to-SF hops into the SFF or the SFC infrastructure.

Level-2: Reduction of the (collapsed) SFP within the SFC infrastructure to the shortest possible.

Figure 1 to Figure 3 show one sequence of offload events leading to a shorter SFP. Although not shown in these figures, further offload events ultimately lead to an optimum SFP.

Below steps show one such sequence of offload events that lead to such an optimum SFP.

Stage-1: Prior to any offloads, service function path SFP1 (corresponding to SFC1) has the following actual forwarding path as shown in Figure 1:
CF ->
NF1 -> SFF1 -> SF1 -> SFF1 -> NF1 ->
NF2 -> SFF2 -> SF2 -> SFF2 -> NF2 ->
NF3 -> SFF3 -> SF3 -> SFF3 -> NF3 ->

Stage-2: After SF2 performs a simple offload, which is signaled to upstream SFFs - SFF1, Classifier, SFP1 forwarding path changes to the below as shown in Figure 3:
CF ->
NF1 -> SFF1 -> SF1 -> SFF1 -> NF1 ->
NF3 -> SFF3 -> SF3 -> SFF3 -> NF3 ->

Stage-3: After SF3 performs simple offload, which is signaled to upstream SFFs - SFF2, SFF1, and Classifier, SFP1 forwarding path changes to the below (figure not shown) />:
CF ->
NF1 -> SFF1 -> SF1 -> SFF1 -> NF1 ->

Stage-4: After SF1 performs simple offload, which is signaled to upstream SFFs - Classifier, SFP1 forwarding path changes to the below (figure not shown):
CF ->

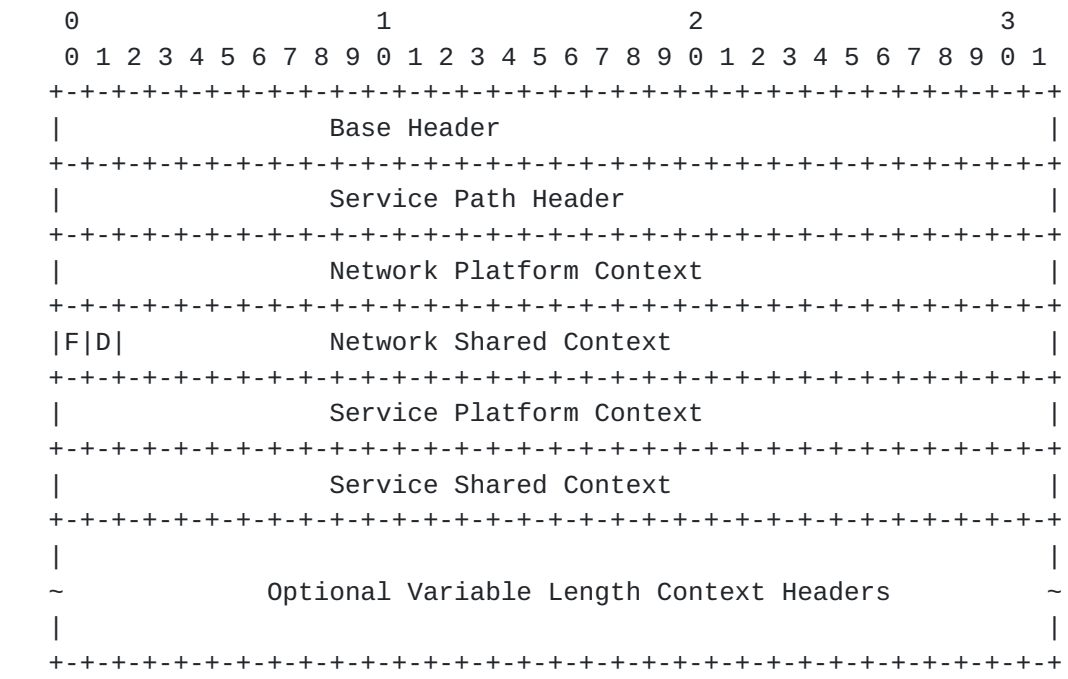
4.2. Service Controller Offload

Each SN signals the service controller of the OFFLOAD and ACTION via control plane messaging for a specific flow. The service controller then signals the appropriate SFFs to offload the requested SFs, there by achieving the hop-by-hop offload behavior.

The service controller has full knowledge of all the SFs of the SFP offloading the flow and hence can determine the optimum SFP within the Service Controller and program the appropriate SFFs to achieve SFP optimization.

5. Offload Data-plane Signaling

Since Offload and action are signaled at the time of returning the traffic to SFF, post servicing the traffic, such signaling can be integrated into the service header of the packet. Figure 4 shows the bits necessary to achieve the signaling using the SFC encapsulation as described in [I-D.quinn-sfc-nsh].



F : Offload bit; indicates offload request when F is set to 1
D : Drop (or Deny) bit, drop when D=1 and permit when D=0;
Drop bit is valid only when F is set to 1

Figure 4: NSH Offload and ACTION Bits

Although SFs can signal SFFs by piggy backing on the serviced packet, SFFs cannot signal in a similar fashion. This is because the traffic is forwarded along the SFP to the next or downstream SFF. The offload signaling has to go to the upstream SFFs, in the opposite direction. SFFs have a choice: perform out-of-band signaling towards the upstream SFFs or wait until traffic arrives on the opposite flow on the reverse SFP, where SFPs are symmetric. SFFs can then piggyback the offload signaling on the reverse traffic towards the upstream SFFs. The actual method employed to signal offload between the SFFs is implementation specific.

6. Acknowledgements

The authors would like to thank Nagaraj Bagepalli for his review comments.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

Security of the offload signaling mechanism is very important. This document does not advocate any additional security mechanisms other the data plane and control plane signaling security mechanisms.

9. References

9.1. Normative References

[I-D.ietf-sfc-architecture]
Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", draft-ietf-sfc-architecture-02 (work in progress), September 2014.

[I-D.quinn-sfc-nsh]
Quinn, P., Guichard, J., Fernando, R., Surendra, S., Smith, M., Yadav, N., Agarwal, P., Manur, R., Chauhan, A., Elzur, U., Garg, P., McConnell, B., and C. Wright, "Network Service Header", draft-quinn-sfc-nsh-03 (work in progress), July 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[I-D.ietf-sfc-problem-statement]
Quinn, P. and T. Nadeau, "Service Function Chaining Problem Statement", draft-ietf-sfc-problem-statement-10 (work in progress), August 2014.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

Authors' Addresses

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

Email: smkumar@cisco.com

Jim Guichard
Cisco Systems, Inc.

Email: jguichar@cisco.com

Paul Quinn
Cisco Systems, Inc.

Email: paulq@cisco.com

Joel Halpern
Ericsson

Email: joel.halpern@ericsson.com