SRv6OPS
Internet Draft
Intended status: Standards Track
Expires: April 22, 2026

J. Ye, Ed.
China Mobile
C. Lin, Ed.
New H3C Technologies

October 20, 2025

SRv6 SFC Deployment Options
draft-ye-srv6ops-sfc-deployment-00

Abstract

This document mainly introduces the factors to consider for supporting SRv6 SFC from the aspects of deployment and reliability methods.

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 https://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 22, 2026.

Copyright Notice

Table of Contents

1. Introduction

Service Function Chaining (SFC) is a network architecture that realizes ordered traffic processing by combining multiple service functions (such as firewall, load balancing, DPI, etc.) into a logical chain in a specific sequence. In traditional networks, service functions are typically deployed separately in the form of physical or virtual devices, whereas SFC dynamically steers traffic through these functions via standardized mechanisms, enhancing flexibility and automation capabilities.Service Function Chaining (SFC) enables directed traffic processing by sequentially orchestrating service functions such as firewall, load balancing, and deep packet inspection (DPI) into logical chains. Compared to traditional networks with scattered physical/virtual devices, SFC significantly enhances network flexibility and automation through standardized mechanisms that dynamically steer traffic through Service Function nodes (SF).

[RFC7665] defines the SFC architecture, while the Network Service Header (NSH) proposed in [RFC8300] serves as a dedicated encapsulation protocol, ensuring precise forwarding of service chains by carrying metadata.

Although NSH is a key implementation solution for SFC, its practical deployment faces three core issues:

1) Insufficient compatibility: Mainstream network devices (especially hardware switches) and operating systems have limited support for NSH, with only a few routers capable of delivering full functionality;

2) Scalability bottleneck: Requires maintaining per-path state on all Service Function nodes (SF) and Service Function Forwarders (SFF), causing management complexity to rise sharply with scale;

3) Architectural fragmentation: Relies on tunneling mechanisms like VxLAN/GRE for cross-node forwarding, making it difficult to achieve true integration between Overlay and Underlay.

Figure 1 shows the NSH-based SFC solution, requiring the SFF to maintain state for each SF, and the SFs need to be upgraded to support NSH.

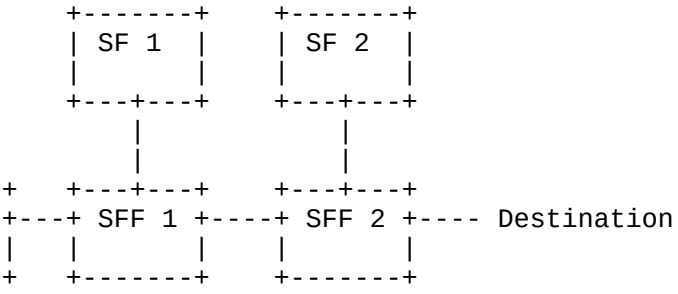

Figure 1: SRv6 SFC

The Service Function Chaining begins with the Classifier, which splits incoming traffic and inserts Network Service Headers (NSH) to initiate service chaining while embedding relevant metadata. The packet then proceeds to the Service Function Forwarder (SFF), which maintains per-service-chain state information and intelligently routes packets either to appropriate Service Functions (SFs) or subsequent SFFs based on the NSH directives. Each SF operates as a specialized Functional Node (FN) that executes designated operations on the traffic flow before returning processed packets back to the forwarding path. This coordinated interaction between classifier, forwarder, and functional nodes ensures seamless traversal through the configured service chain.

Segment Routing over IPv6 (SRv6) further enhances SFC capabilities by leveraging IPv6 extension headers to enable source-based routing. This approach not only simplifies the forwarding process but also significantly improves the flexibility and programmability of SFC deployments, allowing operators to dynamically control traffic paths while maintaining scalability and efficiency.

SRv6 SFC Network Simplified Architecture:

No need for dedicated intermediate equipment (e.g., OpenFlow controllers); paths are directly encoded using SRv6 Segment Lists, reducing protocol stack complexity.

Natively supports Overlay, avoiding the overhead of tunnel encapsulation such as VXLAN/GRE.

End-to-End Programmability:

Service functions are identified by SRv6 SIDs, enabling dynamic insertion, deletion, or reordering of service chains.

Combined with network slicing, it enables customized SFCs for different services (e.g., low-latency chains, high-security chains).

Fine-Grained Traffic Control:

Explicit path specification via Segment Lists eliminates the overhead of traditional NSH (Network Service Header).

Supports TE (Traffic Engineering) to dynamically mitigate congestion nodes

Whether NSH is supported during deployment mainly depends on whether the current SFC network has already deployed NSH. If NSH is already in place, the newly added part can include an SFC Proxy before the NSH-unaware SF nodes to remove the NSH header and convert it to an SRv6 header. If it is a newly deployed SFC network, it is recommended to avoid using the NSH header as much as possible.

As a transition for SFC networks that already support NSH, SFC proxies can be used to enable newly added FC nodes to operate without NSH support.

This document focuses on the deployment practices and reliability design of SRv6 SFC, proposing the use of SRv6 to address the inherent defects of NSH and providing a transition solution for hybrid networking.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terminology

SFC: Service Function Chaining

NSH: Network Service Header

3. Deployment Considerations

This section discusses the key aspects to consider for SRv6 SFC deployment, including the use of stateful or stateless SRv6 SFC, deployment of proxies, and support for multi-tenancy.

3.1. Implementation Methods

Stateless SRv6 SFC

Principle: Achieves path programming by combining SRv6 SIDs (Segment IDs), using the SRH (Segment Routing Header) to carry complete forwarding path information. Each node performs actions based on the SID without maintaining per-flow state.

Advantages: Simplifies the control plane (CP) and reduces resource consumption; suitable for scenarios where service function (SF) nodes natively support SRv6, offering strong expandability.

Use Cases: Network function nodes (e.g., firewall, load balancing) already support SRv6, and highly flexible cloud-native environments.

Stateful SRv6 SFC

Principle: Combines SRv6 with NSH (Network Service Header), using NSH to carry service chain metadata. SF nodes process traffic based on NSH information, while SRv6 is used only for underlying transport.

Advantages: Compatible with legacy SF nodes that do not support SRv6; mature transition solution.

Use Cases: Existing networks containing SF nodes without SRv6 support, or environments requiring coexistence with existing NSH architectures.

SRv6 network itself is a stateless network architecture. In general, when deploying a brand-new SRv6 SFC network, it is preferable to use stateless SRv6 SFC to simplify the control plane (CP), which makes maintenance easier.

However, in most cases, networks are deployed incrementally-that is, upgraded based on existing infrastructure-making SRv6 SFC promotion more practical. When current devices do not support SRv6, a proxy approach can be adopted for deployment; specific proxy modes are described in Section 3.3.

Additionally, for NSH support, refer to section 3.2. If the current SFC network supports NSH but a newly added device does not, a code-based deployment approach can also be adopted. In this case, the SFC proxy device removes the NSH header, records the NSH state, forwards the service packet to the FC, then upon receiving the processed packet from the FC, adds back the NSH header according to the recorded NSH state and continues processing.

3.2. SRv6 SFC with SFC Proxy

If an SF node can support SRv6, it is called an SRv6 aware SF; otherwise, it is called an SRv6 unaware SF. If there exists an SRv6 unaware SF in the network, an SFC Proxy needs to be deployed in front of it. The SRv6 proxy caches the mapping relationship between the SRH and the virtual interface connecting the SF, which is used to recover SRv6 packets based on the virtual interface when packets return from the SF.

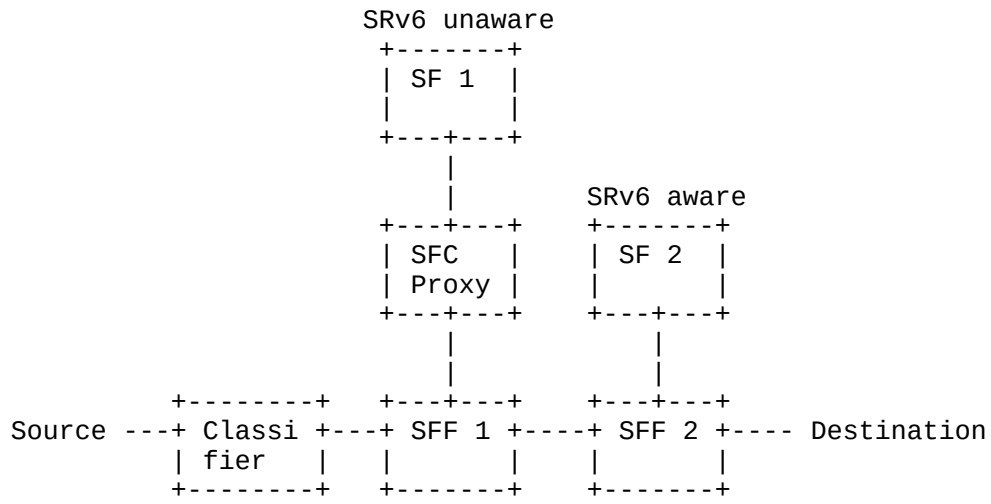

Figure 2: SRv6 SFC with SFC Proxy

SF:

Functional node (FN) that performs specific operations on traffic.

If it can process the SRv6, it is called SRv6-aware SF; otherwise, it is called SRv6-unaware SF.

SFC Proxy:

Caches the mapping relationship between SRH and the virtual interface connecting SF, which is used to recover SRv6 packets based on the virtual interface when packets return from SF.

Whether to deploy the SFC Proxy depends on whether all SF nodes in the current network are SRv6-aware nodes.

The choice of proxy method depends on the support status of the SF node.

Below are several proxy methods available for selection.

Static SR Proxy SFC:

The static service chain proxy belongs to SRv6-unaware Service. Since the SF cannot identify SRv6 packets, the SFF needs to decapsulate the SRv6 packet and forward the raw data packet from the

customer network to the SF for processing. After the SF processes the original packet, it forwards it back to the SFF node, which then decides whether to continue forwarding the packet within the SRv6 SFC network. If forwarding continues within the SRv6 SFC network, the SFF node must re-encapsulate the processed service packet with an SRv6 header based on a manually configured SID list.

Dynamic SR Proxy SFC:

Compared to static SR Proxy, dynamic SR Proxy has the ability to learn mapping relationship cache entries. The mapping relationship cache between SRH and the interface connecting SF is dynamically generated based on the SRH of the messages received by SFF.

After enabling SFF redundant backup protection function, when the primary SFF senses that it is unreachable to SF, the SRH are not removed. Instead, SFF1 needs to carry the original SRH to the backup SFF.

Masquerading SR Proxy SFC:

The service chain pseudo proxy also belongs to SRv6-unaware Service. However, the SF can identify IPv6 packets, and the SFF forwards SRv6 packets directly to the SF for processing. After the SF completes the application service processing, it does not process the SRH header information of SRv6, but directly forwards the packet back to the SFF node, where the SFF processes the packet according to the normal SRv6 forwarding procedure.

3.3. Multi-tenancy support

During deployment, support for multi-tenancy must be considered. One approach is for the Service Function (SF) to assign different SRv6 SIDs for each tenant, with the Classifier encapsulating distinct SRv6 SID sequences per tenant. This method allows the SF to identify the tenant based on the SRv6 SID, offering simple implementation. However, its drawback is that assigning individual SRv6 SIDs for each tenant consumes significant SID resources, which must be evaluated against the number of tenants. An alternative approach is to carry tenant information within the service packet itself. This avoids the need to allocate separate SRv6 SIDs per tenant but results in a more complex system implementation.

4. Reliability Considerations

This section discusses the main aspects to consider for SRv6 SFC reliability, including reliability mechanisms and fault detection.

4.1. Reliability mechanism

When the Service Function (SF) is unreachable without any protection, packets requiring processing by the application service node (SF) will be dropped upon reaching the Service Function Forwarder (SFF), failing to be forwarded in the intended logical order or processed by the SF.

To enhance the reliability of the SRv6 SFC service chain proxy, mechanisms are needed to protect the SFF and improve reliability. For example, dual-homing protection or Bypass protection can be selected to protect the SFF, or a combination of both.

SF Dual-Homing Protection:

Dual-homing protection refers to an SFF or SFC Proxy being dual-homed to two Services Functions (SFs), one primary and one backup. When the primary SF becomes unreachable, the service switches over to the backup SF for processing.

As shown in Figure 3, when a link fault occurs between SF 1 and SFF, the packet is forwarded to SF 1’ to handle the service request.

As shown in Figure 3, when a link fault occurs between SF 1 and the SFC Proxy, the packet is forwarded to SF 1’ to handle the service request.

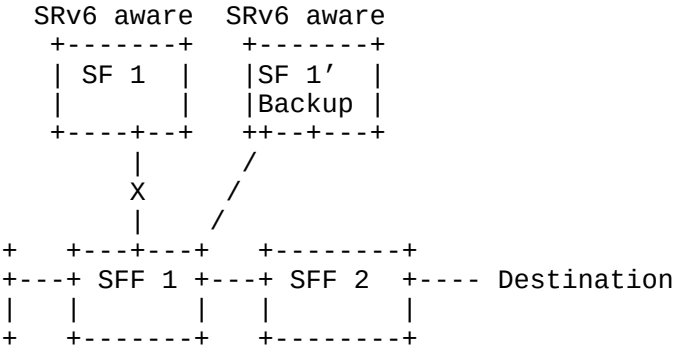

Figure 3: Dual-Homing Protection by SFF

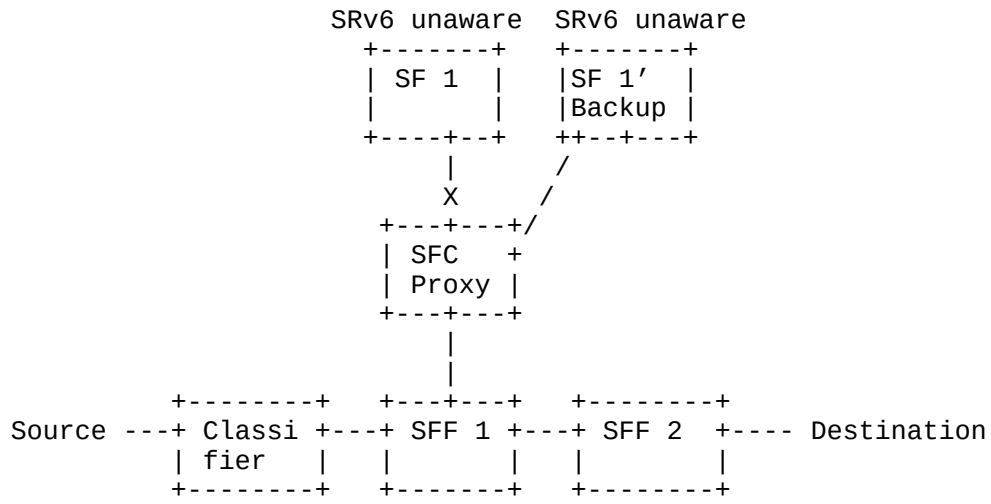

Figure 4: Dual-Homing Protection by SFC Proxy

SF Bypass Protection:

Bypass protection refers to the scenario where, in the event of a failure of the application service node SF or SFC Proxy, packets bypass that service node and no longer go through it for processing.

As shown in Figure 5, when a link fault occurs between SF 1 and SFF 1, skip the service operation of SF 1 and send the packet to SF 2 to continue executing the service operation of SF 2.

As shown in Figure 6, when a link fault occurs between SF 1 and the SFC Proxy, skip the service operation of SF 1 and send the packet to SF 2 to continue executing the service operation.

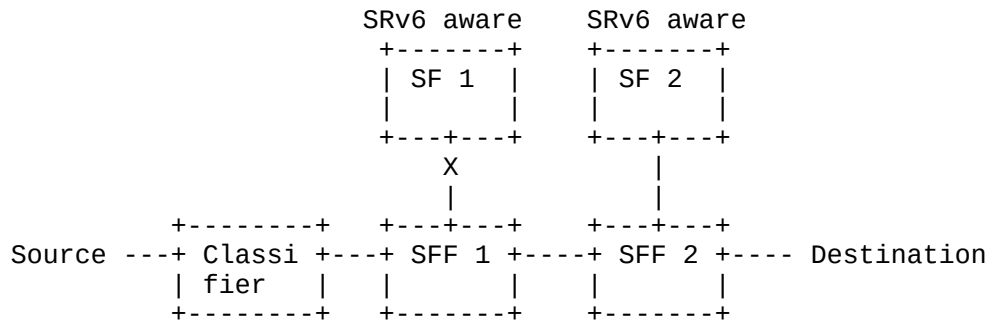

Figure 5: SF Bypass Protection by SFF

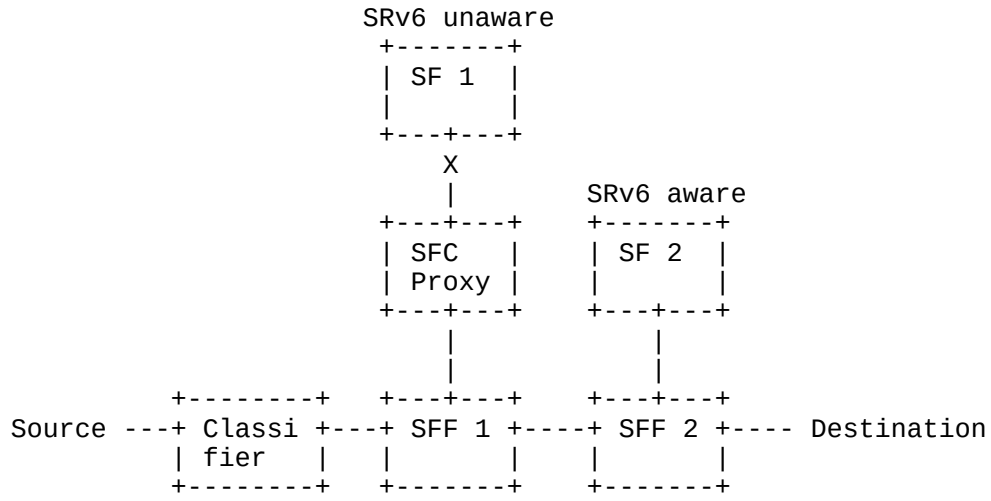

Figure 6: SF Bypass Protection by SFC Proxy

Mixed SF Dual-Homing Protection and SF Bypass Protection:

When SF dual-homing protection and Bypass protection are used together, prioritize executing the dual-homing protection procedure; switch to the Bypass protection procedure only when dual-homing protection fails.

When deploying, select whether to support high availability (HA) based on the service’s importance and consideration of device costs.

4.2. Fast Fault Detection:

There needs to be a fast failure detection mechanism to detect reachability between SFF and SF, or between SFF and SFC Proxy. When the SF becomes unreachable, the protection mechanism can be quickly triggered.

You can choose to use BFD to detect reachability between SFF and SF, or between SFC Proxy and SR. When the SF becomes unreachable, the dual-homing protection or Bypass protection mechanism for SRv6 SFC service chain can be quickly triggered.

The choice of fast fault detection method depends on the requirements for service fault response speed and deployment cost.

5. IANA Considerations

TBD.

6. Security Considerations

TBD.

7. Acknowledgements

TBD.

8. References

8.1. Normative References

[RFC7665] J. Halpern, Ed., Ericsson, C. Pignataro, Ed., Cisco, "Service Function Chaining (SFC) Architecture", BCP 78, RFC 7665, October 2015.

[RFC8300] P. Quinn, Ed., Cisco, U. Elzur, Ed., Intel, C. Pignataro, Ed., Cisco, "Network Service Header (NSH)", BCP 78, RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

8.2. Informative References

TBD.

Contributors

Authors' Addresses

Jiaming Ye (editor)
China Mobile
China
Email: yejiaming@chinamobile.com

Changwang Lin (editor)
New H3C Technologies
China
Email: linchangwang.04414@h3c.com

1. introduction

2. terminology

3. implementation methods

SRv6 xx SRv6 SFC

3.1 stateful SRv6 SFC

NSH

3.2 stateless SRv6 SFC

SF SRv6 SR proxy SRv6

3.2.1

4. reliability consideration

5. fast fault detection