

SFC WG
Internet-Draft
Intended status: Experimental
Expires: 22 September 2022

CJ. Bernardos
UC3M
A. Mourad
InterDigital
21 March 2022

NSH extensions for local distributed SFC control
draft-bernardos-sfc-nsh-distributed-control-04

Abstract

Service function chaining (SFC) allows the instantiation of an ordered set of service functions and subsequent "steering" of traffic through them. In order to set up and maintain SFC instances, a control plane is required, which typically is centralized. In certain environments, such as fog computing ones, such centralized control might not be feasible, calling for distributed SFC control solutions. This document specifies several NSH extensions to provide in-band SFC control signaling.

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 22 September 2022.

Copyright Notice

Copyright (c) 2022 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 (<https://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 Revised BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Revised BSD License.

Table of Contents

1.	Introduction	2
2.	Terminology	4
3.	Local SFC control signaling extending NSH	4
4.	IANA Considerations	7
5.	Security Considerations	8
6.	Acknowledgments	8
7.	References	8
7.1.	Normative References	8
7.2.	Informative References	8
	Authors' Addresses	8

[1.](#) Introduction

Virtualization of functions provides operators with tools to deploy new services much faster, as compared to the traditional use of monolithic and tightly integrated dedicated machinery. As a natural next step, mobile network operators need to re-think how to evolve their existing network infrastructures and how to deploy new ones to address the challenges posed by the increasing customers' demands, as well as by the huge competition among operators. All these changes are triggering the need for a modification in the way operators and infrastructure providers operate their networks, as they need to significantly reduce the costs incurred in deploying a new service and operating it. Some of the mechanisms that are being considered and already adopted by operators include: sharing of network infrastructure to reduce costs, virtualization of core servers running in data centers as a way of supporting their load-aware elastic dimensioning, and dynamic energy policies to reduce the monthly electricity bill. However, this has proved to be tough to put in practice, and not enough. Indeed, it is not easy to deploy new mechanisms in a running operational network due to the high dependency on proprietary (and sometime obscure) protocols and interfaces, which are complex to manage and often require configuring multiple devices in a decentralized way.

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 data center, the WAN, the RAN, and even on mobile nodes.

Service functions (SFs), also referred to as VNFs, or just functions, are hosted on compute, storage and networking resources. The hosting environment of a function is called Service Function Provider or NFVI-PoP (using ETSI NFV terminology).

Services are typically formed as a composition of SFs (VNFs), with each SF providing a specific function of the whole service. Services also referred to as Network Services (NS), according to ETSI terminology.

With the arrival of virtualization, the deployment model for service function is evolving to one where the traffic is steered through the functions wherever they are deployed (functions do not need to be deployed in the traffic path anymore). For a given service, the abstracted view of the required service functions and the order in which they are to be applied is called a Service Function Chain (SFC). An SFC is instantiated through selection of specific service function instances on specific network nodes to form a service graph: this is called a Service Function Path (SFP). The service functions may be applied at any layer within the network protocol stack (network layer, transport layer, application layer, etc.).

The concept of fog computing has emerged driven by the Internet of Things (IoT) due to the need of handling the data generated from the end-user devices. The term fog is referred to any networked computational resource in the continuum between things and cloud. A fog node may therefore be an infrastructure network node such as an eNodeB or gNodeB, an edge server, a customer premises equipment (CPE), or even a user equipment (UE) terminal node such as a laptop, a smartphone, or a computing unit on-board a vehicle, robot or drone.

In fog computing, the functions composing an SFC are hosted on resources that are inherently heterogeneous, volatile and mobile [[I-D.bernardos-sfc-fog-ran](#)]. This means that resources might appear and disappear, and the connectivity characteristics between these resources may also change dynamically. These scenarios call for

distributed SFC control solutions, where there are SFC pseudo controllers, enabling autonomous SFC self-orchestration capabilities. The concept of SFC pseudo controller (P-CTRL) is described in [\[I-D.bernardos-sfc-distributed-control\]](#), as well different procedures for their discovery and initialization.

This document specifies several NSH extensions to provide in-band SFC control signaling.

[2.](#) Terminology

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 [\[RFC2119\]](#).

The following terms used in this document are defined by the IETF in [\[RFC7665\]](#):

Service Function (SF): a function that is responsible for specific treatment of received packets (e.g., firewall, load balancer).

Service Function Chain (SFC): for a given service, the abstracted view of the required service functions and the order in which they are to be applied. This is somehow equivalent to the Network Function Forwarding Graph (NF-FG) at ETSI.

Service Function Forwarder (SFF): A service function forwarder is responsible for forwarding traffic to one or more connected service functions according to information carried in the SFC encapsulation, as well as handling traffic coming back from the SF.

SFI: SF instance.

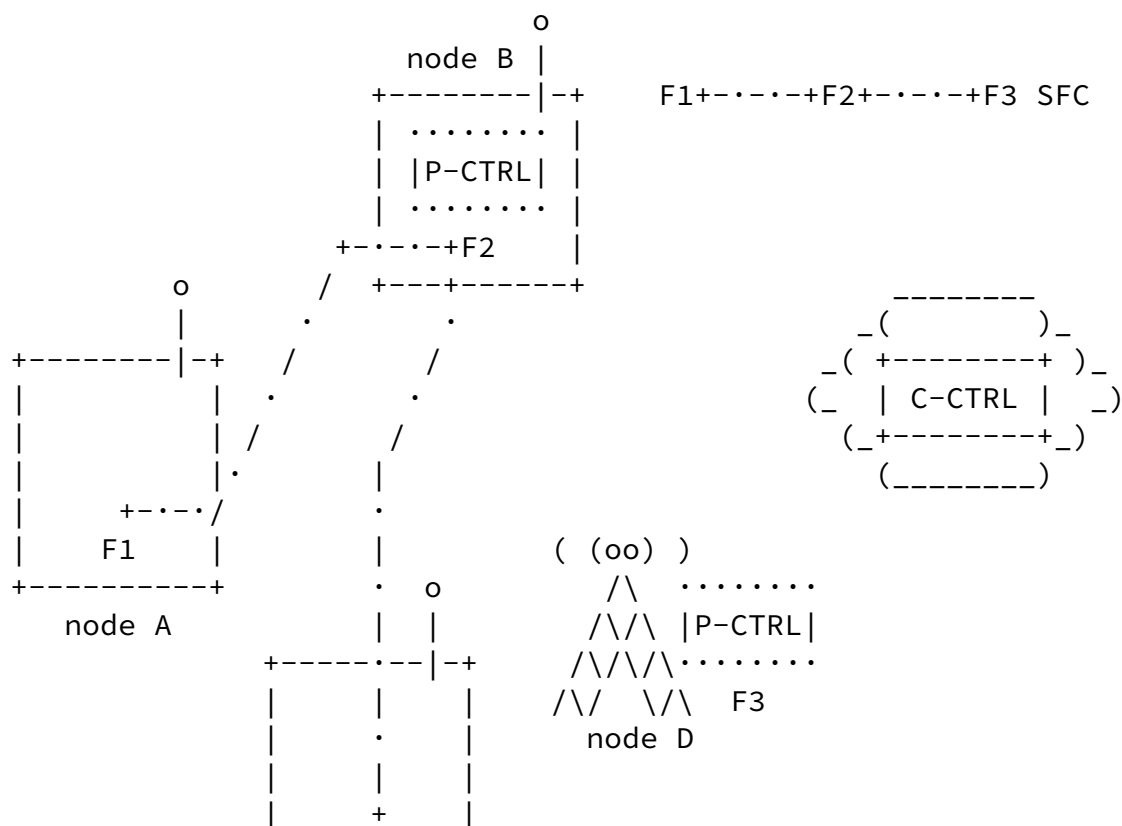
Service Function Path (SFP): the selection of specific service function instances on specific network nodes to form a service graph through which an SFC is instantiated.

The following terms are used in this document:

SFC Pseudo Controller (P-CTRL): logical entity [[I-D.bernardos-sfc-distributed-control](#)], complementing the SFC controller/orchestrator found in current architectures and deployments. It is service specific, meaning that it is defined and meaningful in the context of a given network service. Compared to existing SFC controllers/orchestrators, which manage multiple SFCs instantiated over a common infrastructure, pseudo controllers are constrained to service specific lifecycle management.

SFC Central Controller (C-CTRL): central control plane logical entity in charge of configuring and managing the SFC components [[RFC7665](#)].

[3.](#) Local SFC control signaling extending NSH



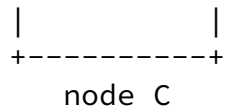
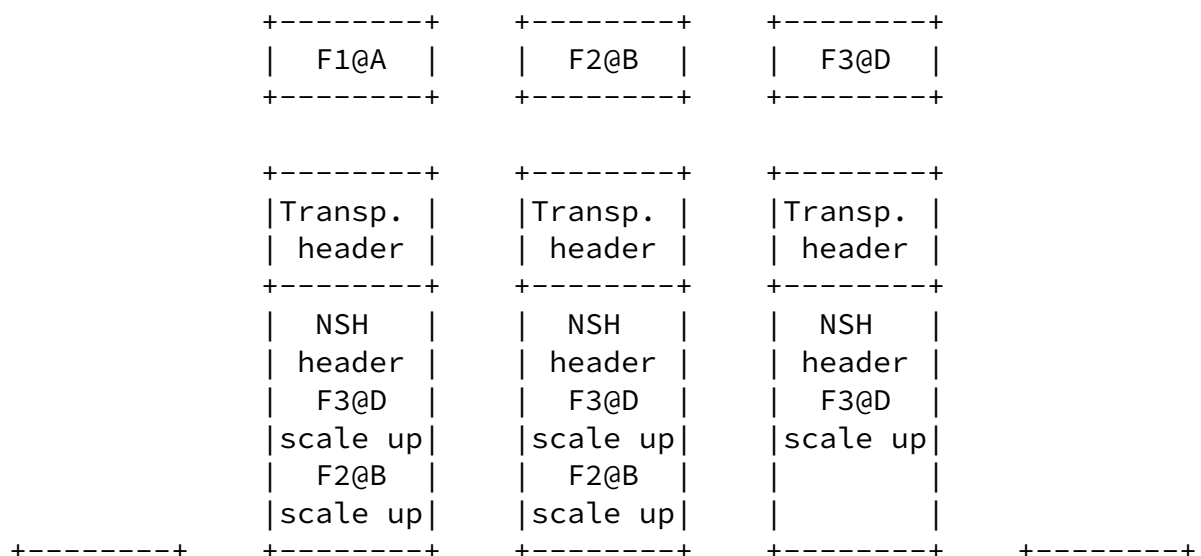


Figure 1: Example SFC scenario

Figure 1 shows an exemplary scenario to show the use of the new NSH extensions. In this scenario, there is no mobility, so nodes are not moving out of radio coverage. In this scenario, at a given point in time the service demands increase, which requires F2 (running at node B) and F3 (running at node D) to have more resources allocated, as otherwise the service would not meet the required SLA. This is detected by the P-CTRL through service-specific local OAM monitoring. Once detected the need of scaling up the resources at nodes B and D, P-CTRL notifies this through in-band signaling in the actual data packets processed by the SFC. This is shown in Figure 2. Note that the use of in-band signaling provides a more efficient way of conveying the signaling, as well as supports multiple NS lifecycle management operations (even addressing different nodes) to be conveyed in a single message.



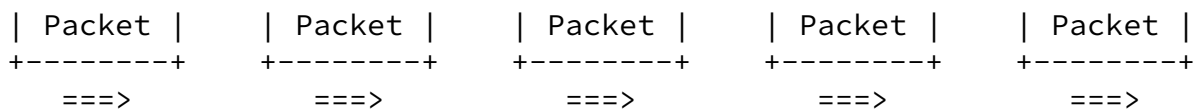
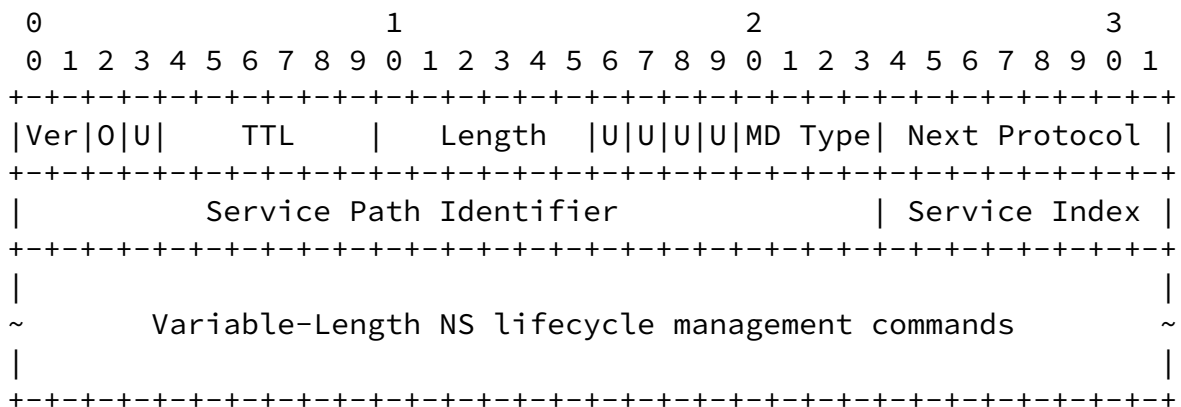
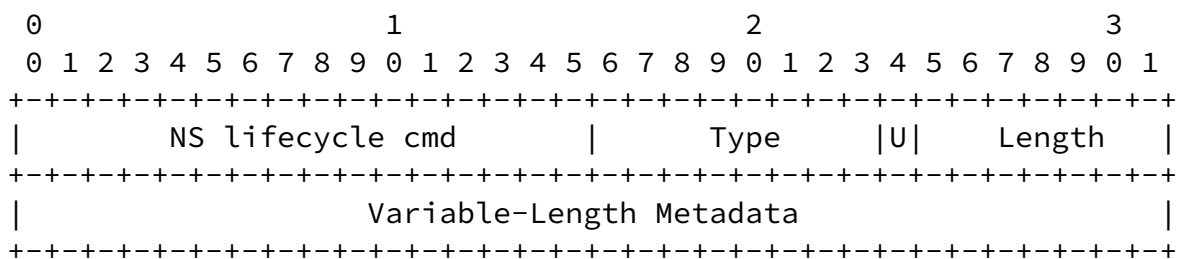


Figure 2: In-band NS lifecycle management signaling extending NSH

The NS lifecycle management commands conveyed in the NSH are transported as a new NSH metadata (MD) type (e.g., Type 3, as current NSH specifications only support 2 types), as shown next:



The format of the new variable-length field for NS lifecycle management commands is shown next:



* NS lifecycle cmd: the NS lifecycle management command. This is a non-limiting list of the commands:

- Scale in.
- Scale out.
- Scale up.

- Scale down.
 - Instantiate function.
 - Terminate function.
 - Configure function.
 - Upgrade function.
 - Update function.
 - Update function.
 - Onboard VNFD.
 - Onboard OAMD.
 - Sync state.
 - Request to overcome CTRL.
 - CTRL activation.
- * Type: indicates the explicit type of command carried out. This depends on the orchestration framework implementation.
 - * Unassigned bit: one unassigned bit is available for future use. This bit MUST NOT be set, and it MUST be ignored on receipt.
 - * Unassigned bit: one unassigned bit is available for future use. This bit MUST NOT be set, and it MUST be ignored on receipt.

[4.](#) IANA Considerations

N/A.

[5.](#) Security Considerations

TBD.

6. Acknowledgments

The work in this draft has been partially supported by the H2020 5Growth (Grant 856709) and 5G-DIVE projects (Grant 859881).

7. References

7.1. Normative References

[I-D.bernardos-sfc-distributed-control]

Bernardos, C. J. and A. Mourad, "Distributed SFC control for fog environments", Work in Progress, Internet-Draft, [draft-bernardos-sfc-distributed-control-05](https://www.ietf.org/archive/id/draft-bernardos-sfc-distributed-control-05), 27 January 2022, <<https://www.ietf.org/archive/id/draft-bernardos-sfc-distributed-control-05.txt>>.

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

7.2. Informative References

[I-D.bernardos-sfc-fog-ran]

Bernardos, C. J. and A. Mourad, "Service Function Chaining Use Cases in Fog RAN", Work in Progress, Internet-Draft, [draft-bernardos-sfc-fog-ran-10](https://www.ietf.org/archive/id/draft-bernardos-sfc-fog-ran-10), 22 October 2021, <<https://www.ietf.org/archive/id/draft-bernardos-sfc-fog-ran-10.txt>>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", [RFC 7665](#), DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.

Authors' Addresses

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
28911 Leganes, Madrid
Spain
Phone: +34 91624 6236
Email: cjbc@it.uc3m.es

URI: <http://www.it.uc3m.es/cjbc/>

Alain Mourad

InterDigital Europe

Email: Alain.Mourad@InterDigital.com

URI: <http://www.InterDigital.com/>

