SFC WG
Internet-Draft
Intended status: Experimental
Expires: January 27, 2021

CJ. Bernardos
UC3M
A. Mourad
InterDigital

July 26, 2020

Distributed SFC control for fog environments
draft-bernardos-sfc-distributed-control-02

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 introduces the role of SFC pseudo-controller and specifies solutions to select and initialize such new logical function.

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 January 27, 2021.

Table of Contents

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. This document introduces this concept and presents first ideas on mechanisms to select and initialize a service-specific SFC pseudo controller among host nodes which are participating in the SFC.

2. Terminology

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.

3. Problem statement

[RFC7665] describes an architecture for the specification, creation, and ongoing maintenance of Service Function Chains (SFCs) in a network. It includes architectural concepts, principles, and components used in the construction of composite services through deployment of SFCs.

The SFC architecture assumes there is a control plane that configures and manages the SFC components. This role is typically assumed to be played by a centralized controller/orchestrator. This implies that dynamic changes on the SFC (composition, function migration, scaling, etc) can only be performed by the centralized controller, which needs to always have connectivity with the functions, and have updated information on the status of all the nodes hosting the functions. Also, multiple services are managed by the same controller/ orchestrator, even if they provide different functionalities with disparate requirements.

In a fog environment, with current management and orchestration solutions, SFCs cannot operate if the nodes hosting the functions get disconnected from the infrastructure. This implies that the lifecycle management of an SFC cannot be managed if disconnected from the centralized controller, which means that important actions (e.g., scaling, migrating a function or updating the data plane) might not take place due to the lack of connectivity with the controller/ orchestrator (even if connectivity issues are just temporal ones). Additionally, lifecycle management of SFCs require up-to-date

monitoring information, with a refresh frequency that is service-specific and might involve a very high overhead with the controller. This severely limits the capability of fast reacting to events local to the nodes hosting the functions, as the SFC cannot autonomously self-orchestrate (decisions can only be taken by the centralized controller/orchestrator).

Figure 1 shows an exemplary scenario where a drone makes use of a network service composed of the chain of functions F1-F2-F3. F1 runs on the drone itself (node A), F2 runs in another drone (node B) and F3 runs in a gNB on the ground (node D). The service might be, for example, an autonomous video surveillance activity in which a couple of drones with different types of cameras make use of image recognition to decide where to go next. If the drones move out of the coverage of the node D, the service chain needs to be reconfigured (for example migrating F3 to node C) so it can remain operative (as node D is hosting one function of the SFC). Since node D is also providing the drones with connectivity to the network infrastructure where the SFC controller is located, this type of events cannot be resolved by the SFC controller, as the nodes hosting the functions would be disconnected from the controller.

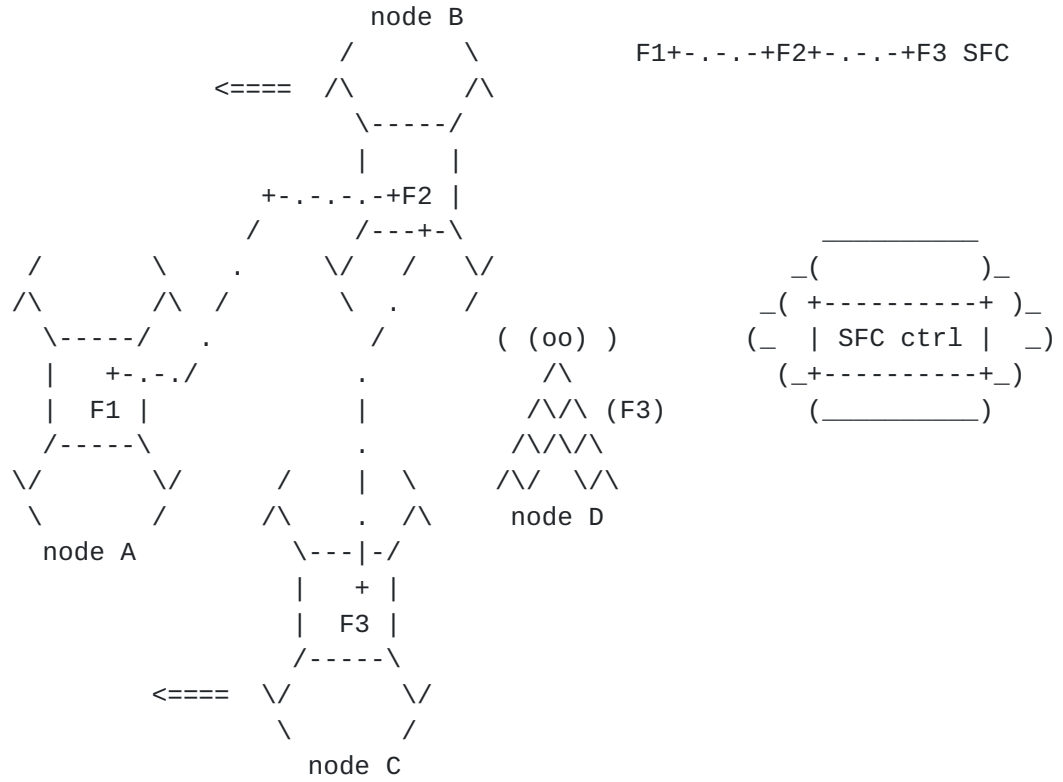

Figure 1: SFC example scenario

Another scenario that cannot be properly tackled with current SFC orchestration approaches control appears with highly mobile/volatile environments and/or latency-demanding services, in which centralized lifecycle management is unfeasible due to its high signaling cost (e.g., they require frequent measurements sent from remote nodes to a centralized controller, generating too much signaling, and involving a delay that might be too long to meet the service requirements).

4. Distributed SFC control

The fact that -- in fog computing environments -- SFC functions are hosted in heterogeneous, volatile and mobile resources, calls for new orchestration solutions able to cope with dynamic changes to the resources in runtime or ahead of time (in anticipation through prediction) as opposed to today's solutions which are inherently reactive and static or semi-static.

These new orchestration solutions have to enable SFCs to autonomously self-orchestrate without having to rely on a centralized controller. The idea introduced in this draft is to enable one of the nodes involved in a service function chain (i.e., in a specific service) to be prepared to take over the control of the SFC and perform network service lifecycle management decisions, replacing at least temporary and at least partially the centralized SFC controller.

This draft proposes a new logical entity, complementing the SFC controller/orchestrator found in current architectures and deployments. We refer to this new entity as SFC pseudo controller, and it is characterized by the following:

- 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. These SFC pseudo controllers synchronize with the SFC centralized controller/orchestrator to ensure proper resource orchestration.

- Potentially, any node involved in a network service might play the role of SFC pseudo controller. But note that it is not mandatory that all nodes are willing/capable to play that role. Therefore, we consider that on a given deployment, only a subset of all the involved nodes are willing or capable of doing so. We refer to these nodes as candidate pseudo controllers. During the initialization phase, out of the candidates, one will be chosen by the SFC controller as selected SFC pseudo controller. Each candidate pseudo controller maintains a local copy of the information required to properly perform lifecycle management of

the service. This includes not only information about the network service (e.g., the Network Service Descriptor, NSD, and the Virtual Network Function Descriptors, VNFDs, as defined by ETSI NFV, and the characterization of the resource capabilities of the nodes hosting the functions), but also the information related to performing an efficient monitoring of the service. We refer to this new descriptor as Operations, Administration and Maintenance Descriptor (OAMD).

- From the set of available candidate SFC pseudo controllers, one is chosen as selected pseudo controllers for a network service. This active pseudo controller performs monitoring activities at service and resource level and synchronizes periodically with the centralized SFC controller/orchestrator. Note that this is performed in an opportunistic way, if connectivity is available, and that disconnected operation is possible.

Candidate pseudo controller instances might be located at any node hosting a service function. The SFC controller typically runs in the network core, at a server (either as a physical or virtual function). The SFC controller and the candidate pseudo controller instances exchange signaling to initialize the selected SFC pseudo controller. This includes signaling to describe the service, signaling to express the readiness and preference from the candidates to play the role of pseudo controller, and the signaling from the SFC controller to indicate the selected one. This is explained in more detail next.

4.1. SFC pseudo controller initialization

This section describes how SFC pseudo controller candidates are determined and selected SFC pseudo controllers are chosen from the candidate set.

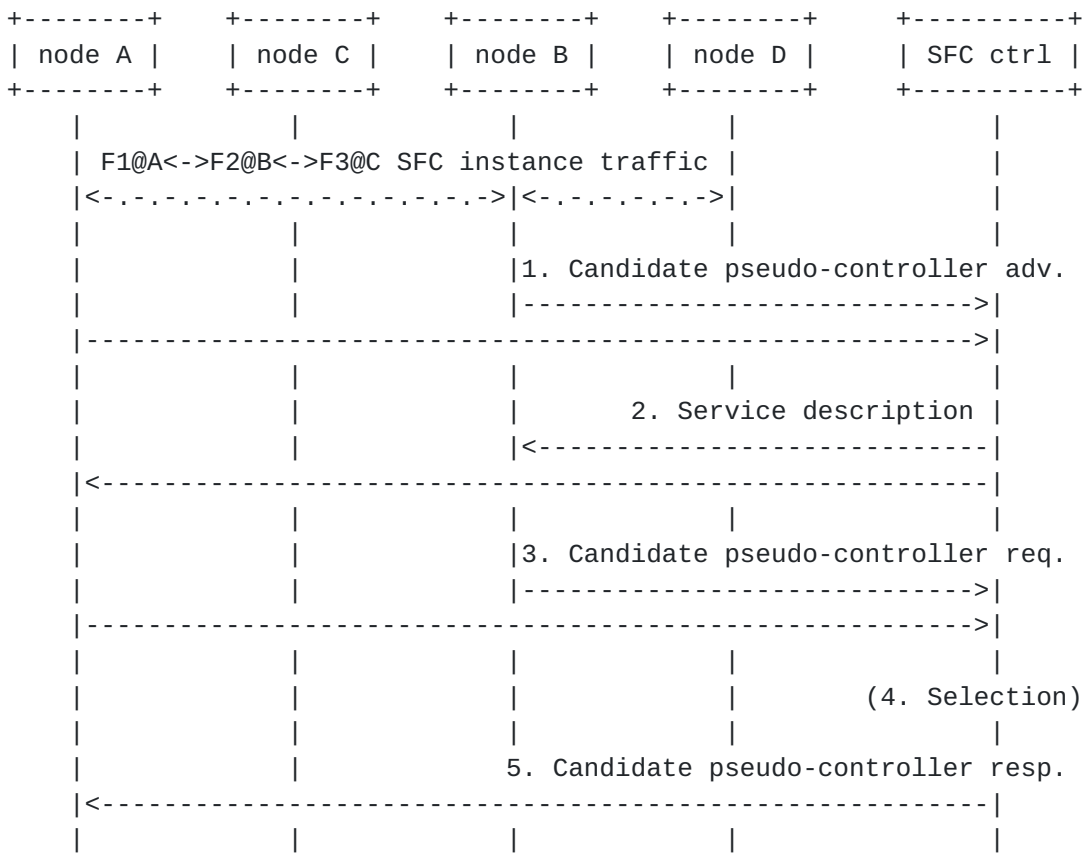

Figure 2: SFC pseudo controller initialization

A detailed message sequence chart is shown in Figure 2. The different steps are described next:

1. Among the fog nodes hosting functions of an SFC, several might be willing to play the role of SFC pseudo controller. There is an exchange of information between the SFC controller and the SFC nodes, where the SFC pseudo controller candidates will be determined by fog nodes' willingness/preference for a service. A preference value from the candidate nodes is used by the SFC controller as primary value to decide which is the selected SFC pseudo controller, but other information can also be used in this decision, such as, but not limited to: local policies, volatility of the node, known capabilities of the node (monitoring, connectivity, available resources, etc) and so on. Those nodes willing to play the role of SFC pseudo controller advertise their presence to the SFC controller, through which SFC controller can recognize the willingness or readiness of the nodes and will enable to inform proper nodes of the detailed service description such as NSD and OAMD. This message includes:

* The service ID, identifying the service (note that multiple SFCs might be running in parallel, even with some nodes participating simultaneously in more than one).

* An ID of the candidate SFC pseudo controller. This ID uniquely identifies the pseudo controller instance. It might be generated using a unique identifier of the node.

2. The SFC controller shares information about the service (e.g., service requirement, desired monitoring configuration) with the nodes, through which information the nodes can assess the own availability as a pseudo controller for the specific service and decides some preferences. This includes:

   * The service ID, identifying the specific service.

   * The Network Service Descriptor (NSD) of the service, which includes the description of the chain in terms of composing functions and logical links connecting them, as well as associated requirements (e.g.: compute requirements, connectivity, affinity, etc).

   * The information required to perform an efficient monitoring of the service: the Operations, Administration and Maintenance Descriptor (OAMD). Examples of this information are: latency constraints for all logical links of the service chain, bandwidth requirements for all logical links, maximum tolerated packet losses for each of the logical links and the whole end-to-end service, tolerated jitter, minimum required availability for a given instantiation (i.e., the minimum time the functions need to remain instantiated on their hosting resources to ensure a minimum consistency for the service), battery status/lifetime, periodicity on which each of the parameters needs to be monitored, etc.

   * Security capabilities required to manage the service, which include the set of security mechanisms that need to be supported for an (pseudo-)orchestrator to be able to manage the service. This set might include a set of alternatives, so different solutions can be used. Examples of these capabilities are: authentication algorithms, encryption algorithms, supported certificate authorities, etc.

3. The candidate SFC pseudo controllers respond to the SFC controller, including the following information:

   * The service ID, identifying the specific service.

* An ID, identifying the SFC pseudo controller instance.

* A preference value for the candidate node to play the role of active pseudo controller. This preference is represented with an integer that indicates the willingness of the candidate node to become an "active" SFC pseudo-controller for the running service. This preference is locally selected by the node based on the specifics of the service. Note that the nodes have all the information about the service (contained in the NSD) and its monitoring (the nodes also have the OAMD, which specifies which aspects need to be monitored, at both service and resource level). Each node decides the local preference value based on this service information, as well as its own capabilities (e.g., whether it is capable of performing the associated control and monitoring tasks and if it is willing to assume the associated cost, for example in terms of energy consumption -- if the node is battery-powered).

* A list of other known candidate SFC pseudo-controllers. A node might know this based on local broadcasts from those candidate pseudo-controllers (advertising its presence). This allows the SFC controller to discover if there are other potential candidate pseudo-controllers available.

* A list of the supported security mechanisms.

4. With the information received from the candidate SFC pseudo controllers, the (centralized) SFC controller decides which node becomes the selected SFC pseudo-controller. The selection is based on the preference value indicated by the candidate pseudo controllers. Note that multiple nodes could include the same preference value, and in case of a tie, the SFC controller might use a policy to select one, or simply use a tie-breaking rule (for example selecting the one with a lowest/highest ID). In this example, A is selected to be the SFC pseudo controller.

5. The SFC controller responds to the selected SFC pseudo controller with a message that includes:

* The service ID, identifying the specific service.

* The ID of the selected SFC pseudo controller instance.

* Security material. The centralized SFC controller and the SFC nodes have pre-established security credentials, allowing the controller to perform the required orchestration tasks (they have a secure signaling channel). Since this security

relationship does not exist between any pair of the nodes, the SFC controller acts as a security "anchor", by providing the SFC pseudo controller -- using the secure channel -- with security material allowing to securely control all the nodes part of the SFC. One example of approach is to generate a certificate, signed by the SFC controller, that can then be used by the SFC pseudo controller. Other approaches are possible.

* Profiles of all involved nodes in the service. This includes information about the resources of the involved nodes (plus additional also considered by the SFC controller in case function migration is needed). Note that once a SFC pseudo controller is selected, it could also query and request information about the nodes that are part of the SFC. The information included in the profile of a node may contain: Virtual machine specification, computation properties (RAM size, disk size, memory page size, number of CPUs, number of cores per CPU, number of threads per core), storage requirements, Scale out/scale in limits, network and connectivity properties (number and type of interfaces), etc.

* The list of other candidate SFC pseudo-controllers. This allows local synchronization among candidate pseudo controllers if needed.

The new signaling messages described below can be implemented as either new protocol messages, e.g., via REST API, or as extensions of either inband or outband protocols. Examples of those include: NSH (for inband signaling among SFC nodes), IPv6 (for outband via new extension headers). Details and examples of signaling will be added in future revisions of this draft.

5. IANA Considerations

N/A.

6. Security Considerations

TBD.

7. Acknowledgments

The work in this draft has been developed under the framework of the H2020 5G-DIVE project (Grant 859881).

8. Informative References

[I-D.bernardos-sfc-fog-ran]
Bernardos, C., Rahman, A., and A. Mourad, "Service Function Chaining Use Cases in Fog RAN", draft-bernardos-sfc-fog-ran-07 (work in progress), March 2020.

[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
Leganes, Madrid 28911
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/