Workgroup: SFC WG
Internet-Draft:
draft-bernardos-sfc-distributed-control-operation-04
Published: 21 March 2022
Intended Status: Experimental
Expires: 22 September 2022
Authors: CJ. Bernardos A. Mourad
UC3M InterDigital

Distributed SFC control operation

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 describes a general framework for distributed SFC operation.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction
2. Terminology
3. Problem statement
4. Distributed SFC control operation
  4.1. P-CTRL taking over C-CTRL
    4.1.1. P-CTRL taking over C-CTRL due to a local monitoring event
    4.1.2. P-CTRL taking over C-CTRL due to a C-CTRL failure
    4.1.3. C-CTRL gaining back control
  4.2. Inter P-CTRL seamless handover
5. IANA Considerations
6. Security Considerations
7. Acknowledgments
8. References
  8.1. Normative References
  8.2. Informative References
Authors' Addresses

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 introduces a framework for local distributed SFC operation, by allowing P-CTRLs to temporarily substitute the central controller (C-CTRL) in its task to carry out the global lifecycle management of the given 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.

The following terms are defined and 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. Problem statement

Mobile network architectures are evolving to support network virtualization and service function orchestration. Current Service

Function Chain (SFC) architectures [RFC7665] rely on a centralized controller/orchestrator (C-CTRL) which shall be connected to all the hosts participating in a given SFC. This poses issues and inefficiencies in fog computing environments especially because of the mobility and volatility of some hosts, as well as the associated signaling overhead.

These problems can be alleviated by enabling autonomous SFC self-orchestration (SOC), based on the concept of SFC pseudo controller (P-CTRL) introduced in [I-D.bernardos-sfc-distributed-control]. A pseudo controller is capable of substituting (at least temporarily and partially) the centralized SFC controller in situations where the centralized controller may not be able to perform its functions (e.g., when the connectivity with some hosts is broken).

[I-D.bernardos-sfc-distributed-control] introduces the role of the SFC pseudo controller and describes mechanisms to select and initialize a service-specific SFC pseudo controller among host nodes which are participating in the SFC. This document specifies mechanisms to enable an SFC pseudo controller trigger and control NS lifecycle management operations, such as migration of NS functions, chains or parts of a chain.

Figure 1 shows an exemplary scenario where a host UE makes use of an NS composed of the chain of SFs F1-F2-F3. These functions may be application functions -- using 3GPP jargon -- network functions or over-the-top functions. Non-limiting examples of these functions are: load balancers, traffic steering, performance enhancement proxies (PEPs), video transcoders, firewalls, etc. In this example, F1 instance runs on a first UE (node A), F2 instance runs on a second UE (node B), and F3 instance runs on a gNB (node D). SFC pseudo controller instances are assumed running on UE node A and D (which is a gNB). Node A and B are connected via D2D communications. If all the UEs move out of the coverage of the gNB node D, the service chain will then need to be reconfigured to maintain service continuity as gNB node D is hosting one function (F3) of the chain and would become disconnected. Since gNB node D is also providing the UEs with connectivity to the network infrastructure where the SFC central controller is hosted, this type of event cannot be resolved by the SFC controller, as the nodes hosting the functions would be disconnected from the central controller. Similar problems arise in highly mobile/volatile and/or latency-demanding scenarios, where centralized lifecycle management becomes unsuitable.

```
                             o
                      node B |
                    +--------|-+    F1+-·--·-+F2+-·--·-+F3 SFC
                    | ········ |
             <====  | |P-CTRL| |
                    | ········ |
                  +-·--·-+F2   |
         o       /  +---+------+                 ________
         |      .       .                      _(        )_
+--------|-+   /       /                     _( +--------+ )_
|          |  ·       ·                     (_  | C-CTRL |  _)
|          | /       /                        (_+--------+_)
|          |·       |                           (________)
|     +-·-·/        ·
|    F1    |        |         ( (oo) )
+----------+        ·  o         /\  ········
   node A           |  |        /\/\ |P-CTRL|
              +-----·--|-+     /\/\/\········
              |     |    |    /\/  \/\ (F3)
              |     ·    |      node D
       <====  |     |    |
              |     +    |
              |    F3    |
              +----------+
                 node C
```

Figure 1: Example scenario: disrupted SFC due to mobility

In these scenarios, an SFC pseudo controller can substitute (at least temporarily and partially) the centralized SFC controller when the latter is not available or able to perform a given task. This document proposes solutions addressing the following problem: How to enable SFC pseudo controllers to perform NS lifecycle management operations, such as migration of functions, service chains or parts of a chain? This requires solving the sub-problems listed below.

* When a SFC centralized controller (C-CTRL) is in charge, what mechanisms and exchanges are needed between the C-CTRL and a SFC pseudo-controller (P-CTRL), and the nodes, so as to facilitate a seamless transition from C-CTRL to P-CTRL (due to an event, e.g. failure, of the C-CTRL)?

* When P-CTRL takes over from the C-CTRL, what actions shall it take towards other P-CTRLs, so as to sustain seamless transition if a first P-CTRL fails (e.g., moving from P-CTRL A to P-CTRL B)?

* When C-CTRL is back into the picture, how shall the control transfer seamlessly back from P-CTRL to C-CTRL?

4. Distributed SFC control operation

A key concept is to allow a P-CTRL to take over temporarily and partially from the C-CTRL to perform NS lifecycle management decisions. The definition, selection and initialization of a P-CTRL is covered in [I-D.bernardos-sfc-distributed-control].

Using Figure 1, we can think of an example where a function (F3) is migrated from node D to node C, triggered by the move of nodes A and B hosting F1 and F2 away from the coverage of node D hosting F3 (nodes A nd B are UEs within coverage of node D which is a gNB). The P-CTRL in node B performs OAM operations locally and monitors the NS-specific SLAs. Upon detecting or predicting that the NS-specific SLAs may not be met in the near future, P-CTRL A takes actions to temporary and partially substitute C-CTRL, and starts performing local NS lifecycle management operations (e.g., instantiating F3 on node C, since current hosting node -- node D --is predicted to become unreachable soon).

Note that, in the previous example, the prediction and local NS lifecycle management operations could have been performed by P-CTRL running at node D as well. We have assumed that the active (designated) P-CTRL is running at node B, but could have been at node D as well, which would imply the need to also migrate the active P-CTRL role to node B.

Thanks to enabling P-CTRL B to perform local NS lifecycle management decisions, service continuity will be guaranteed when C-CTRL fails or is out of reach.

The "activation" of P-CTRL operation only occurs when C-CTRL cannot properly operate (e.g., it is disconnected from the SFC or it is not reacting fast enough to the local changing conditions). For example, P-CTRL can send a scaling command to a given node, in order to adapt the resources to the current NS demands. P-CTRL would also notify this to C-CTRL, as soon as the connection to C-CTRL is recovered so that both are synchronized.

In order to support the operation of P-CTRLs complementing or replacing the operation of the C-CTRL, the following operations are needed:

* When an NS is onboarded, the C-CTRL receives an NS descriptor (NSD), together with the VNF descriptors (VNFDs) of the functions composing the service, and the Operations, Administration and Maintenance Descriptor (OAMD), which includes the information of what needs to be monitored to ensure a given SLA.

* When the NS is instantiated, in addition to the regular orchestration decisions (e.g., placement of functions on

available resources, etc.), the C-CTRL, based on its knowledge of existing P-CTRLs, decides how monitoring is going to be performed. This includes: (i) What is monitored by each P-CTRL node (e.g., vCPU load, bandwidth of a certain link, etc.) and how (e.g., active, passive or hybrid monitoring); (ii) Which orchestrators are in charge of collecting and processing the measurements.

Currently defined mechanisms assume a semi-static environment and the standardized message flows do not support dynamic migration of the SFC controller role to other entities. Therefore, new signaling flows need to be defined between C-CTRL and P-CTRLs and in-between P-CTRLs: (i) allowing prediction of events via local monitoring and faster reaction, (ii) enabling orchestration when C-CTRL is temporarily unreachable, and (iii) supporting migrating CTRL role to P-CTRLs.

4.1. P-CTRL taking over C-CTRL

There are two main triggers for a P-CTRL to take over from the C-CTRL: a local monitoring event or a C-CTRL failure. We specify next the procedures for each of these two triggers.

4.1.1. P-CTRL taking over C-CTRL due to a local monitoring event

In this case, the C-CTRL has delegated some monitoring actions to the P-CTRL, as indicated in the OAMD sent by the C-CTRL to the P-CTRL.

```
+---------+  +----+  +---------+ +---------+ +----------+   +------+
| node A  |  | C  |  | node B  | | node D  | |   3GPP   |   | SFC  |
|P-CTRL F1|  | F3 |  |P-CTRL F2| |P-CTRL F3| |ctrl plane|   |C-CTRL|
+--+----+-+  +----+  +--+----+-+ +--+----+-+ +----------+   +------+
   |    |       |       |    |      |    |         |            |
   |  F1@A<->F2@B<->F3@D SFC network service       |            |
   |    |<-.-.-.-.-.-.-.-.-.>|<-.-.-.-.->|         |A.Serv. OAM |
   |    |       |       |    |      |    |         |<-.-.-.-.-.>|
   |    |       |       |    |B.Serv. OAM monitor. |            |
   |    |       |       |    |-.-.-.-.-.-.-.-.-.-.>|            |
   |    |       |       |    |<-.-.-.-.-.-.-.-.-.-.|            |
   |    |       |       |    |      |    |         |            |
   |    |       |       |    |C.Serv. OAM monitor. |            |
   |    |       |       |    |<-.-.-|    |         |            |
   |    |       |       |    |-.-.-.-.-.-.-.-.-.-.>|            |
   |    |       |       |    |<-.-.-.-.-.-.-.-.-.-.|            |
   |    |       |       |    |-.-.->|    |         |            |
   |    | D.Serv. OAM monitor.      |    |         |            |
   |    |       |<-.-.->|    |      |    |         |            |
   |    |       |       |<-.-.-.-.->|    |         |            |
   |<-.>|<-.-.-.-.-.-.->|    |      |    |         |            |
   |<-.-.-.-.-.-.-.-.-.>|    |      |    |         |            |
   |    |       |       |    |      |    |         |            |
   |    |    P-CTRL@B detects that it's  |         |            |
   |    |   loosing connectivity with D  |         |            |
   |    |       |       |    |      |    |         |            |
   |    | Orch. signal. |    |      |    |         |            |
   |    |       |<-.-.->|    |      |    |         |            |
   |    |<-.-.-.-.-.-.->|    |      |    |         |            |
   |    |       |       |    |  Sync. with C-CTRL  |            |
   |    |       |       |    |<.-.-.-.-.-.-.-.-.-.>|            |
   |    |       |       |    |      |    |         |            |
```

Figure 2: P-CTRL taking over C-CTRL due to a local monitoring event

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

* (We assume that the network service has been instantiated and there is traffic: F1@A--F2@B--F3@D).

* C-CTRL runs overall OAM monitoring of the network services. This can be done for example by contacting the 3GPP network to obtain different network analytics via NEF. This is shown in the figure as A.

* P-CTRLs are running service specific OAM monitoring actions, as indicated in the OAMD sent by the C-CTRL in the network service

instantiation procedure. This requires signaling procedures. Various non-limiting example options are possible:

- The P-CTRL may directly obtain information metrics from different network functions through the network exposure function (NEF). This is shown in the figure as B.

- The P-CTRL may indirectly obtain the information metrics through the local AF or NF hosted on the UE, which interacts with any other entity inside or outside 3GPP (e.g., AF to AF, NF to AF, or NF to NF) and then parse these on the interface to the P-CTRL. If the function hosted on the UE is an NF, and the info is about 3GPP network data analytics, then the NF will go get some data from NWDAF and locally at the UE, expose these to the P-CTRL. This is shown in the figure as C.

- In addition to the former (mutually exclusive approaches), it is also possible to perform local OAM monitoring via standalone procedures, such as local OAM monitoring and the use of SFC OAM. This is shown in the figure as D.

The interface between the P-CTRL and the SFC functions running on the UE to obtain OAM metrics may be a local API, or standard interface like IETF SFC OAM, or like the interface between 3GPP NWDAF and an NWDAF service consumer.

* At a certain point in time, a local monitoring event, which cannot be detected by the C-CTRL, triggers the whole process. In the example of the figure P-CTRL@node B detects that B is losing connectivity with node D.

* The P-CTRL takes an orchestration decision based on its local knowledge and signals it to the involved nodes. The decision consists in migrating/moving F3 from node D to C. New extensions to MIPv6 are used, as described in TBD. Alternatively, extensions to IETF SFC NSH can also be used, as described in TBD.

* The P-CTRL also informs the C-CTRL to keep it synchronized. Similarly, the C-CTRL can then update other P-CTRLs if needed. New extensions to NSH are used (NS lifecycle management), as described in TBD.

4.1.2. P-CTRL taking over C-CTRL due to a C-CTRL failure

In this case, the P-CTRL detects/predicts a C-CTRL failure (e.g., it becomes unreachable).

```
+---------+  +----+  +---------+ +---------+ +----------+   +------+
| node A  |  | C  |  | node B  | | node D  | |   3GPP   |   | SFC  |
|P-CTRL F1|  | F3 |  |P-CTRL F2| |P-CTRL F3| |ctrl plane|   |C-CTRL|
+--+----+-+  +----+  +--+----+-+ +--+----+-+ +----------+   +------+
   |    |       |       |    |      |    |         |            |
   |  F1@A<->F2@B<->F3@D SFC  network service      |            |
   |    |<-.-.-.-.-.-.-.-.-.>|<-.-.-.-.->|         |            |
   |    |       |       |    |      |    |         |            |
   |    |       |       |    |   C-CTRL connectivity failure    |
   |    |       |       |    |<.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.->|
   |     P-CTRL@B detects that connectivity        |            |
   |    with C-CTRL is lost (or about to be)       |            |
   |    |       |       |    |      |    |         |            |
   |    |       |       |    |      Sync with C-CTRL (if        |
   |    |       |       |    |      failure is predicted)       |
   |    |       |       |    |<.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.->|
   |    |     P-CTRL@B becomes the  |    |         |            |
   |    |     CTRL for the service  |    |         |            |
   |    |       |       |    |      |    |         |            |
   | P-CTRL activation  |    |      |    |         |            |
   |<-.-.-.-.-.-.-.-.-.>|    |      |    |         |            |
   |    |       |       |    |      |    |         |            |
   |    | Orch. signal. |    |      |    |         |            |
   |    |       |<-.-.->|    |      |    |         |            |
   |    |<-.-.-.-.-.-.->|    |      |    |         |            |
   |    |       |       |    |      |    |         |            |
```

Figure 3: P-CTRL taking over C-CTRL due to a C-CTRL failure

A detailed message sequence chart is shown in Figure 3. The different steps are described next:

*(We assume that the network service has been instantiated and there is traffic: F1@A--F2@B--F3@D).

*A failure can be detected by a P-CTRL via multiple mechanisms, such as: (i) Sending periodic keep-alive messages to the C-CTRL; (ii) Transport-layer mechanisms that allow detecting connectivity failures; (iii) Observing a lack of action from the C-CTRL upon an event that requires an orchestration action.

*A failure can also be predicted by a P-CTRL, by using local monitoring information.

*When a C-CTRL failure is detected, the designated backup P-CTRL takes over the orchestration of the network service, by:

-Notifying other P-CTRLs, as well as selecting a new designated backup P-CTRL. This involves the synchronization of relevant information (orchestration DBs, descriptors, etc.) with C-CTRL (if the failure is predicted). This is done through new IETF SFC NSH extensions (NS lifecycle management), as described in TBD.

-The P-CTRL becoming the SFC controller/orchestrator. From this point of time on, the P-CTRL can send orchestration signaling. Extensions to IETF NSH or MIPv6 can be used, as described in TBD and TBD.

4.1.3. C-CTRL gaining back control

We describe next, using an example, how a C-CTRL may get back the orchestration control, temporarily delegated to a P-CTRL.

```
+---------+  +----+  +---------+ +---------+ +----------+   +------+
| node A  |  | C  |  | node B  | | node D  | |   3GPP   |   | SFC  |
|P-CTRL F1|  | F3 |  |P-CTRL F2| |P-CTRL F3| |ctrl plane|   |C-CTRL|
+--+----+-+  +----+  +--+----+-+ +--+----+-+ +----------+   +------+
   |    |       |       |    |      |    |         |            |
   |    |       |       |    |  Connectivity OK notification    |
   |    |       |       |    |      |    |         |-·-·-·-·-·->|
   |    |       |       |-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·>|
   |    |       |       |    |      |    |         |            |
   |    |       |       |   Request to take over SFC control    |
   |    |       |       |<·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-|
   |    |       |       |       Orchestration state sync.       |
   |    |       |       |-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·>|
   |    |       |       |    |      |    |         |            |
   |    |       |       |    |      |    |         |         C-CTRL
   |    |       |       |    |      |    |         |         regains
   |    |       |       |    |      |    |         |         control
   |    |       |       |  Notification C-CTRL back in control  |
   |<-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-·-|
   |    |       |       |    |      |    |         |            |
```

Figure 4: C-CTRL gaining back control

A detailed message sequence chart is shown in Figure 4. The different steps are described next:

*When a C-CTRL loses connectivity with the nodes involved in a service, it enters into recovery mode, waiting for the connectivity to be recovered. C-CTRL can learn that connectivity has been regained using NSH OAM signaling or 3GPP signaling from the NEF or the P-CTRL itself.

*When connectivity is gained back to a P-CTRL, the C-CTRL signals its availability so the P-CTRL can give back the control of the service. To do so, IETF SFC NSH extensions (NS lifecycle management) can be used, as described in TBD.

* The C-CTRL becomes now the active SFC controller/orchestrator for the service.

* The C-CTRL notifies to other P-CTRLs that it is back in control of the service, using IETF SFC NSH extensions (NS lifecycle management), as described in TBD.

4.2. Inter P-CTRL seamless handover

In scenarios with no C-CTRL reachability, it might be needed to transition from one P-CTRL to another one (e.g., because of mobility of the nodes while the C-CTRL is not reachable).

Reactive transition is supported as for the case of C-CTRL failure. Proactive/seamless transition is addressed as follows.

```
+---------+  +----+  +----+ +---------+ +----------+   +------+
| node A  |  | C  |  | B  | | node D  | |   3GPP   |   | SFC  |
|P-CTRL F1|  | F3 |  | F2 | |P-CTRL F3| |ctrl plane|   |C-CTRL|
+--+----+-+  +----+  +----+ +--+----+-+ +----------+   +------+
   |    |       |       |      |    |         |            |
   |  F1@A<->F2@B<->F3@D SFC  network service |            |
   |    |<------------->|<--------->|         |            |
   |    |       |       |      |    |         |            |
   |    |       |       | P-CTRL@D can no     |            |
   |    |       |       | longer act as CTRL  |            |
   |    |       |       |      |    |         |            |
  Notification to other P-CTRLs|    |         |            |
   |<--------------------------|    |         |            |
  Request to take over control |    |         |            |
   |-------------------------->|    |         |            |
  Orchestration state sync     |    |         |            |
   |<--------------------------|    |         |            |
   |    |       |       |      |    |         |            |
P-CTRL@A becomes|       |      |    |         |            |
  the CTRL for  |       |      |    |         |            |
  the service   |       |      |    |         |            |
   |    |       |       |      |    |         |            |
```

Figure 5: Inter P-CTRL seamless handover

A detailed message sequence chart is shown in Figure 5. The different steps are described next:

* (We assume that the network service has been instantiated and there is traffic: F1@A--F2@B--F3@D).

*When the active (designated) P-CTRL detects that it might not be able to operate in the near future (lack of resources, battery, moving away, etc.), a notification is sent to other P-CTRLs using new IETF SFC NSH extensions (NS lifecycle management), as described in TBD.

*Each P-CTRL receiving the notification message that is ready to take over the role of active P-CTRL sends a message to the current P-CTRL, which selects one. New IETF SFC NSH extensions are used to convey this signaling.

*At this point, the new P-CTRL becomes the SFC controller/ orchestrator of the service.

5. IANA Considerations

N/A.

6. Security Considerations

TBD.

7. Acknowledgments

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

8. References

8.1. 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, 27 January 2022, <https://www.ietf.org/archive/id/draft-bernardos-sfc-distributed-control-05.txt>.

8.2. 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, 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/