Independent Submission
Request for Comments: 8677
Category: Informational
ISSN: 2070-1721

D. Trossen
InterDigital Europe, Ltd
D. Purkayastha
A. Rahman
InterDigital Communications, LLC

November 2019

Name-Based Service Function Forwarder (nSFF) Component within a Service Function Chaining (SFC) Framework

Abstract

Adoption of cloud and fog technology allows operators to deploy a single "Service Function" (SF) to multiple "execution locations". The decision to steer traffic to a specific location may change frequently based on load, proximity, etc. Under the current Service Function Chaining (SFC) framework, steering traffic dynamically to the different execution endpoints requires a specific "rechaining", i.e., a change in the service function path reflecting the different IP endpoints to be used for the new execution points. This procedure may be complex and take time. In order to simplify rechaining and reduce the time to complete the procedure, we discuss separating the logical Service Function Path (SFP) from the specific execution endpoints. This can be done by identifying the SFs using a name rather than a routable IP endpoint (or Layer 2 address). This document describes the necessary extensions, additional functions, and protocol details in the Service Function Forwarder (SFF) to handle name-based relationships.

This document presents InterDigital's approach to name-based SFC. It does not represent IETF consensus and is presented here so that the SFC community may benefit from considering this mechanism and the possibility of its use in the edge data centers.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The requirements on today's networks are very diverse, enabling multiple use cases such as the Internet of Things (IoT), Content Distribution, Gaming, and Network functions such as Cloud Radio Access Network (RAN) and 5G control planes based on a Service-Based Architecture (SBA). These services are deployed, provisioned, and managed using Cloud-based techniques as seen in the IT world. Virtualization of compute and storage resources is at the heart of providing (often web) services to end users with the ability to quickly provision virtualized service endpoints through, e.g., container-based techniques. This creates the ability to dynamically compose new services from existing services. It also allows an operator to move a service instance in response to user mobility or to change resource availability. When moving from a purely "distant cloud" model to one of localized micro data centers with regional, metro, or even street level, often called "edge" data centers, such virtualized service instances can be instantiated in topologically different locations with the overall "distant" data center now being transformed into a network of distributed ones. The reaction of content providers, like Facebook, Google, NetFlix, and others, is not just to rely on deploying content servers at the ingress of the customer network. Instead, the trend is towards deploying multiple Point of Presences (POPs) within the customer network, those POPs being connected through proprietary mechanisms [Schlinker2017] to push content.

The Service Function Chaining (SFC) framework [RFC7665] allows network operators as well as service providers to compose new services by chaining individual "service functions". Such chains are expressed through explicit relationships of functional components (the SFs) realized through their direct Layer 2 (e.g., Media Access Control (MAC) address) or Layer 3 (e.g., IP address) relationship as defined through next-hop information that is being defined by the network operator. See Section 4 for more background on SFC.

In a dynamic service environment of distributed data centers such as the one outlined above, with the ability to create and recreate service endpoints frequently, the SFC framework requires reconfiguring the existing chain through information based on the new relationships, causing overhead in a number of components, specifically the orchestrator that initiates the initial SFC and any possible reconfiguration.

This document describes how such changes can be handled without involving the initiation of new and reconfigured SFCs. This is accomplished by lifting the chaining relationship from Layer 2 and Layer 3 information to that of SF "names", which can, for instance, be expressed as URIs. In order to transparently support such named relationships, we propose to embed the necessary functionality directly into the Service Function Forwarder (SFF) as described in [RFC7665]. With that, the SFF described in this document allows for keeping an existing SFC intact, as described by its Service Function Path (SFP), while enabling the selection of appropriate service function endpoint(s) during the traversal of packets through the SFC. This document is an Independent Submission to the RFC Editor. It is not an output of the IETF SFC WG.

2. Terminology

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.

3. Example Use Case: 5G Control-Plane Services

We exemplify the need for chaining SFs at the level of a service name through a use case stemming from the current 3GPP Release 16 work on Service Based Architecture (SBA) [SDO-3GPP-SBA], [SDO-3GPP-SBA-ENHANCEMENT]. In this work, mobile network control planes are proposed to be realized by replacing the traditional network function interfaces with a fully service-based one. HTTP was chosen as the application-layer protocol for exchanging suitable service requests [SDO-3GPP-SBA]. With this in mind, the exchange between, for example, the 3GPP-defined (Rel. 15) Session Management Function (SMF) and the Access and Mobility Management Function (AMF) in a 5G control plane is being described as a set of web-service-like requests that are, in turn, embedded into HTTP requests. Hence, interactions in a 5G control plane can be modeled based on SFCs where the relationship is between the specific (IP-based) SF endpoints that implement the necessary service endpoints in the SMF and AMF. The SFs are exposed through URIs with work ongoing to define the used naming conventions for such URIs.

This move from a network function model (in pre-Release 15 systems of 3GPP) to a service-based model is motivated through the proliferation of data-center operations for mobile network control-plane services. In other words, typical IT-based methods to service provisioning, particularly that of virtualization of entire compute resources, are envisioned to being used in future operations of mobile networks. Hence, operators of such future mobile networks desire to virtualize SF endpoints and direct (control-plane) traffic to the most appropriate current service instance in the most appropriate (local) data center. Such a data center is envisioned as being interconnected through a software-defined wide area network (SD-WAN). "Appropriate" here can be defined by topological or geographical proximity of the service initiator to the SF endpoint. Alternatively, network or service instance compute load can be used to direct a request to a more appropriate (in this case less loaded) instance to reduce possible latency of the overall request. Such data-center-centric operation is extended with the trend towards regionalization of load through a "regional office" approach, where micro data centers provide virtualizable resources that can be used in the service execution, creating a larger degree of freedom when choosing the "most appropriate" service endpoint for a particular incoming service request.

While the move to a service-based model aligns well with the framework of SFC, choosing the most appropriate service instance at runtime requires so-called "rechaining" of the SFC since the relationships in said SFC are defined through Layer 2 or Layer 3 identifiers, which, in turn, are likely to be different if the chosen service instances reside in different parts of the network (e.g., in a regional data center).

Hence, when a traffic flow is forwarded over a service chain expressed as an SFC-compliant SFP, packets in the traffic flow are processed by the various SF instances, with each SF instance applying an SF prior to forwarding the packets to the next network node. It is a service-layer concept and can possibly work over any Virtual network layer and corresponding underlay network. The underlay network can be IP or alternatively any Layer 2 technology. At the service layer, SFs are identified using a path identifier and an index. Eventually, this index is translated to an IP address (or MAC address) of the host where the SF is running. Because of this, any change-of-service function instance is likely to require a change of the path information since either the IP address (in the case of changing the execution from one data center to another) or MAC address will change due to the newly selected SF instance.

Returning to our 5G control-plane example, a user's connection request to access an application server in the Internet may start with signaling in the control plane to set up user-plane bearers. The connection request may flow through SFs over a service chain in the control plane, as deployed by a network operator. Typical SFs in a 5G control plane may include "RAN termination / processing", "Slice Selection Function", "AMF", and "SMF". A "Network Slice" is a complete logical network including Radio Access Network (RAN) and Core Network (CN). Distinct RAN and CN Slices may exist. A device may access multiple Network Slices simultaneously through a single RAN. The device may provide Network Slice Selection Assistance Information (NSSAI) parameters to the network to help it select a RAN and a Core Network part of a slice instance. Part of the control plane, the Common Control Network Function (CCNF), includes the Network Slice Selection Function (NSSF), which is in charge of selecting core Network Slice instances. The classifier, as described in SFC architecture, may reside in the user terminal or at the Evolved Node B (eNB). These SFs can be configured to be part of an SFC. We can also say that some of the configurations of the SFP may change at the execution time. For example, the SMF may be relocated as the user moves and a new SMF may be included in the SFP based on user location. Figure 1 shows the example SFC described here.

```
+------+   +---------+  +-----+   +-----+
| User |   | Slice   |  |     |   |     |
| App  |-->| Control |->| AMF |-->| SMF |-->
| Fn   |   | Function|  |     |   |     |
+------+   +---------+  +-----+   +-----+
```

Figure 1: Mapping SFC onto Service Function Execution Points along a Service Function Path

4. Background

[RFC7665] describes an architecture for the specification, creation, and ongoing maintenance of SFCs. It includes architectural concepts, principles, and components used in the construction of composite services through deployment of SFCs. In the following, we outline the parts of this SFC architecture relevant for our proposed extension, followed by the challenges with this current framework in the light of our example use case.

4.1. Relevant Part of SFC Architecture

The SFC architecture, as defined in [RFC7665], describes architectural components such as SF, classifier, and SFF. It describes the SFP as the logical path of an SFC. Forwarding traffic along such an SFP is the responsibility of the SFF. For this, the SFFs in a network maintain the requisite SFP forwarding information. Such SFP forwarding information is associated with a service path identifier (SPI) that is used to uniquely identify an SFP. The service forwarding state is represented by the Service Index (SI) and enables an SFF to identify which SFs of a given SFP should be applied, and in what order. The SFF also has information that allows it to forward packets to the next SFF after applying local SFs.

The operational steps to forward traffic are then as follows: Traffic arrives at an SFF from the network. The SFF determines the appropriate SF the traffic should be forwarded to via information contained in the SFC encapsulation. After SF processing, the traffic is returned to the SFF and, if needed, is forwarded to another SF associated with that SFF. If there is another non-local hop (i.e., to an SF with a different SFF) in the SFP, the SFF further encapsulates the traffic in the appropriate network transport protocol and delivers it to the network for delivery to the next SFF along the path. Related to this forwarding responsibility, an SFF should be able to interact with metadata.

4.2. Challenges with Current Framework

As outlined in previous sections, the SFP defines an ordered sequence of specific SF instances being used for the interaction between initiator and SFs along the SFP. These SFs are addressed by IP (or any L2/MAC) addresses and defined as next-hop information in the network locator maps of traversing SFF nodes.

As outlined in our use case, however, the service provider may want to provision SFC nodes based on dynamically spun-up SF instances so that these (now virtualized) SFs can be reached in the SFC domain using the SFC underlay layer.

Following the original model of SFC, any change in a specific execution point for a specific SF along the SFP will require a change of the SFP information (since the new SF execution point likely carries different IP or L2 address information) and possibly even the next-hop information in SFFs along the SFP. In case the availability of new SF instances is rather dynamic (e.g., through the use of container-based virtualization techniques), the current model and realization of SFC could lead to reducing the flexibility of service providers and increasing the management complexity incurred by the frequent changes of (service) forwarding information in the respective SFF nodes. This is because any change of the SFP (and possibly next-hop info) will need to go through suitable management cycles.

To address these challenges through a suitable solution, we identify the following requirements:

* Relations between Service Execution Points MUST be abstracted so that, from an SFP point of view, the Logical Path never changes.

* Deriving the Service Execution Points from the abstract SFP SHOULD be fast and incur minimum delay.

* Identification of the Service Execution Points SHOULD NOT use a combination of Layer 2 or Layer 3 mechanisms.

The next section outlines a solution to address the issue, allowing for keeping SFC information (represented in its SFP) intact while addressing the desired flexibility of the service provider.

5. Name-Based Operation in SFF

5.1. General Idea

The general idea is two pronged. Firstly, we elevate the definition of an SFP onto the level of "name-based interactions" rather than limiting SFPs to Layer 2 or Layer 3 information only. Secondly, we extend the operations of the SFF to allow for forwarding decisions that take into account such name-based interaction while remaining backward compatible to the current SFC architecture as defined in [RFC7665]. In the following sections, we outline these two components of our solution.

If the next-hop information in the Network Locator Map (NLM) is described using an L2/L3 identifier, the name-based SFF (nSFF) may operate as described for (traditional) SFF, as defined in [RFC7665]. On the other hand, if the next-hop information in the NLM is described as a name, then the nSFF operates as described in the following sections.

In the following sections, we outline the two components of our solution.

5.2. Name-Based Service Function Path (nSFP)

The existing SFC framework is defined in [RFC7665]. Section 4 outlines that the SFP information is representing path information based on Layer 2 or Layer 3 information, i.e., MAC or IP addresses, causing the aforementioned frequent adaptations in cases of execution-point changes. Instead, we introduce the notion of a "name-based Service Function Path (nSFP)".

In today's networking terms, any identifier can be treated as a name, but we will illustrate the realization of a "Name-based SFP" through extended SFF operations (see Section 6) based on URIs as names and HTTP as the protocol of exchanging information. Here, URIs are being used to name for an SF along the nSFP. Note that the nSFP approach is not restricted to HTTP (as the protocol) and URIs (as next-hop identifier within the SFP). Other identifiers such as an IP address itself can also be used and are interpreted as a "name" in the nSFP. IP addresses as well as fully qualified domain names forming complex URIs (uniform resource identifiers), such as www.example.com/service_name1, are all captured by the notion of "name" in this document.

Generally, nSFPs are defined as an ordered sequence of the "name" of SFs, and a typical nSFP may look like: 192.0.x.x -> www.example.com -> www.example2.com/service1 -> www.example2.com/service2.

Our use case in Section 3 can then be represented as an ordered named sequence. An example for a session initiation that involves an authentication procedure, this could look like 192.0.x.x -> smf.example.org/session_initiate -> amf.example.org/auth -> smf.example.org/session_complete -> 192.0.x.x. (Note that this example is only a conceptual one since the exact nature of any future

SBA-based exchange of 5G control-plane functions is yet to be defined by standardization bodies such as 3GPP).

In accordance with our use case in Section 3, any of these named services can potentially be realized through more than one replicated SF instance. This leads to making dynamic decisions on where to send packets along the SAME SFP information, being provided during the execution of the SFC. Through elevating the SFP onto the notion of name-based interactions, the SFP will remain the same even if those specific execution points change for a specific service interaction.

The following diagram in Figure 2 describes this nSFP concept and the resulting mapping of those named interactions onto (possibly) replicated instances.

```
+---------------------------------------------------------------+
|Service Layer                                                  |
| 192.0.x.x --> www.example.com --> www.example2.com -->        |
|                       ||               ||                     |
+----------------------||--------------||-----------------------+
                       ||               ||
                       ||               ||
+----------------------||--------------||-----------------------+
|Underlay Network      \/               \/                      |
|               +--+ +--+ +--+    +--+ +--+ +--+                 |
|               |  | |  | |  |    |  | |  | |  |                 |
|               +--+ +--+ +--+    +--+ +--+ +--+                 |
|               Compute and       Compute and                    |
|               storage nodes     storage nodes                  |
+---------------------------------------------------------------+
```

Figure 2: Mapping SFC onto Service Function Execution Points along a Service Function Path Based on Virtualized Service Function Instance

5.3. Name-Based Network Locator Map (nNLM)

In order to forward a packet within an nSFP, we need to extend the NLM as defined in [RFC8300] with the ability to consider name relations based on URIs as well as high-level transport protocols such as HTTP for means of SFC packet forwarding. Another example for SFC packet forwarding could be that of Constrained Application Protocol (CoAP).

The extended NLM or name-based Network Locator Map (nNLM) is shown in Table 1 as an example for www.example.com being part of the nSFP. Such extended nNLM is stored at each SFF throughout the SFC domain with suitable information populated to the nNLM during the configuration phase.

SPI	SI	Next Hop(s)	Transport Encapsulation (TE)
10	255	192.0.2.1	VXLAN-gpe
10	254	198.51.100.10	GRE

10	253	www.example.com	HTTP
40	251	198.51.100.15	GRE
50	200	01:23:45:67:89:ab	Ethernet
15	212	Null (end of path)	None

Table 1: Name-Based Network Locator Map

Alternatively, the extended NLM may be defined with implicit name information rather than explicit URIs as in Table 1. In the example of Table 2, the next hop is represented as a generic HTTP service without a specific URI being identified in the extended NLM. In this scenario, the SFF forwards the packet based on parsing the HTTP request in order to identify the host name or URI. It retrieves the URI and may apply policy information to determine the destination host/service.

SPI	SI	Next Hop(s)	Transport Encapsulation (TE)
10	255	192.0.2.1	VXLAN-gpe
10	254	198.51.100.10	GRE
10	253	HTTP Service	HTTP
40	251	198.51.100.15	GRE
50	200	01:23:45:67:89:ab	Ethernet
15	212	Null (end of path)	None

Table 2: Name-Based Network Locator Map with Implicit Name Information

5.4. Name-Based Service Function Forwarder (nSFF)

It is desirable to extend the SFF of the SFC underlay to handle nSFPs transparently and without the need to insert any SF into the nSFP. Such extended nSFFs would then be responsible for forwarding a packet in the SFC domain as per the definition of the (extended) nSFP.

In our example realization for an extended SFF, the solution described in this document uses HTTP as the protocol of forwarding SFC packets to the next (name-based) hop in the nSFP. The URI in the HTTP transaction is the name in our nSFP information, which will be used for name-based forwarding.

Following our reasoning so far, HTTP requests (and more specifically, the plaintext-encoded requests above) are the equivalent of packets that enter the SFC domain. In the existing SFC framework, an IP payload is typically assumed to be a packet entering the SFC domain. This packet is forwarded to destination nodes using the L2 encapsulation. Any layer 2 network can be used as an underlay network. This notion is now extended to packets being possibly part of an entire higher-layer application such as HTTP requests. The handling of any intermediate layers, such as TCP and IP, is left to the realization of the (extended) SFF operations towards the next (named) hop. For this, we will first outline the general lifecycle of an SFC packet in the following subsection, followed by two examples for determining next-hop information in Section 6.2.3, finished up by a layered view on the realization of the nSFF in Section 6.2.4.

5.5. High-Level Architecture

```
+----------+
| SF1      |                 +---------+                   +-------+
| instance |\                |   NR    |                   | SF2   |
+----------+ \               +---------+                   +-------+
              \                  ||                           ||
+------------+ \ +-------+   +---------+   +---------+   +-------+
| Classifier |---| nSFF1 |---|Forwarder|---|Forwarder|---| nSFF2 |
+------------+   +-------+   +---------+   +---------+   +-------+
                                                              ||
                                                          +----------+
                                                          | Boundary |
                                                          |  node    |
                                                          +----------+
```

Figure 3: High-Level Architecture

The high-level architecture for name-based operation shown in Figure 3 is very similar to the SFC architecture as described in [RFC7665]. Two new functions are introduced, as shown in the above diagram: namely, the nSFF and the Name Resolver (NR).

The nSFF is an extension of the existing SFF and is capable of processing SFC packets based on nNLM information, determining the next SF where the packet should be forwarded, and the required transport encapsulation (TE). Like standard SFF operation, it adds TE to the SFC packet and forwards it.

The NR is a new functional component, capable of identifying the execution endpoints, where a "named SF" is running, triggered by suitable resolution requests sent by the nSFF. Though this is similar to DNS function, it is not same. It does not use DNS protocols or data records. A new procedure to determine the suitable routing/forwarding information towards the nSFF serving the next hop of the SFP is used. The details are described later.

The other functional components, such as classifier and SF, are the same as described in SFC architecture, as defined in [RFC7665], while the Forwarders shown in the above diagram are traditional Layer 2 switches.

5.6. Operational Steps

In the proposed solution, the operations are realized by the name-based SFF, called "nSFF". We utilize the high-level architecture in Figure 3 to describe the traversal between two SF instances of an nSFP-based transaction in an example chain of: 192.0.x.x -> SF1 (www.example.com) -> SF2 (www.example2.com) -> SF3 -> ...

Service Function 3 (SF3) is assumed to be a classical SF; hence, existing SFC mechanisms can be used to reach it and will not be considered in this example.

According to the SFC lifecycle, as defined in [RFC7665], based on our example chain above, the traffic originates from a classifier or another SFF on the left. The traffic is processed by the incoming nSFF1 (on the left side) through the following steps. The traffic exits at nSFF2.

Step 1: At nSFF1, the following nNLM is assumed:

SPI	SI	Next Hop(s)	Transport Encapsulation (TE)
10	255	192.0.2.1	VXLAN-gpe
10	254	198.51.100.10	GRE
10	253	www.example.com	HTTP
10	252	www.example2.com	HTTP
40	251	198.51.100.15	GRE
50	200	01:23:45:67:89:ab	Ethernet
15	212	Null (end of path)	None

Table 3: nNLM at nSFF1

Step 2: nSFF1 removes the previous transport encapsulation (TE) for any traffic originating from another SFF or classifier (traffic from an SF instance does not carry any TE and is therefore directly processed at the nSFF).

Step 3: nSFF1 then processes the Network Service Header (NSH) information, as defined in [RFC8300], to identify the next SF at the nSFP level by mapping the NSH information to the appropriate entry in its nNLM (see Table 3) based on the provided SPI/SI information in the NSH (see Section 4) in order to determine the name-based identifier of the next-hop SF. With such nNLM in mind, the nSFF searches the map for SPI = 10 and SI = 253. It identifies the next hop as = www.example.com and HTTP as the protocol to be used. Given that the next hop resides locally, the SFC packet is forwarded to the SF1 instance of www.example.com. Note that the next hop could also be identified from the provided HTTP request, if the next-hop information was identified as a generic HTTP service, as defined in Section 5.3.

Step 4: The SF1 instance then processes the received SFC packet according to its service semantics and modifies the NSH by setting SPI = 10 and SI = 252 for forwarding the packet along the SFP. It then forwards the SFC packet to its local nSFF, i.e., nSFF1.

Step 5: nSFF1 processes the NSH of the SFC packet again, now with the NSH modified (SPI = 10, SI = 252) by the SF1 instance. It retrieves the next-hop information from its nNLM in Table 3 to be www.example2.com. Due to this SF not being locally available, the nSFF consults any locally available information regarding routing/forwarding towards a suitable nSFF that can serve this next hop.

Step 6: If such information exists, the Packet (plus the NSH information) is marked to be sent towards the nSFF serving the next hop based on such information in Step 8.

Step 7: If such information does not exist, nSFF1 consults the NR to determine the suitable routing/forwarding information towards the identified nSFF serving the next hop of the SFP. For future SFC packets towards this next hop, such resolved information may be locally cached, avoiding contacting the NR for every SFC packet forwarding. The packet is now marked to be sent via the network in Step 8.

Step 8: Utilizing the forwarding information determined in Steps 6 or 7, nSFF1 adds the suitable TE for the SFC packet before forwarding via the forwarders in the network towards the next nSFF22.

Step 9: When the Packet (+NSH+TE) arrives at the outgoing nSFF2, i.e., the nSFF serving the identified next hop of the SFP, it removes the TE and processes the NSH to identify the next-hop information. At nSFF2 the nNLM in Table 4 is assumed. Based on this nNLM and NSH information where SPI = 10 and SI = 252, nSFF2 identifies the next SF as www.example2.com.

SPI	SI	Next Hop(s)	Transport Encapsulation (TE)
10	252	www.example2.com	HTTP
40	251	198.51.100.15	GRE
50	200	01:23:45:67:89:ab	Ethernet
15	212	Null (end of path)	None

Table 4: nNLM at SFF2

Step 10: If the next hop is locally registered at the nSFF, it forwards the packet (+NSH) to the SF instance using suitable IP/MAC methods for doing so.

Step 11: If the next hop is not locally registered at the nSFF, the outgoing nSFF adds new TE information to the packet and forwards the packet (+NSH+TE) to the next SFF or boundary node, as shown in Table 4.

6. nSFF Forwarding Operations

This section outlines the realization of various nSFF forwarding operations in Section 5.6. Although the operations in Section 5 utilize the notion of name-based transactions in general, we exemplify the operations here in Section 5 specifically for HTTP-based transactions to ground our description into a specific protocol for such name-based transaction. We will refer to the various steps in each of the following subsections.

6.1. nSFF Protocol Layers

Figure 4 shows the protocol layers based on the high-level architecture in Figure 3.

```
   +-------+  +------+----+                              +----+-----+
   |App    |  |      |    |   +--------+                 |    |     |
   |HTTP   |  |-------->  |   |  NR    |                 |nSFF----->|--
   |TCP    |->| TCP  |nSFF|   +---/\---+                 |    | TCP | |
   |IP     |  | IP   |    |       ||                     |    | IP  | |
   +-------+  +------+----+  +---------+   +---------+   +----------+ |
   |   L2  |  |      L2   |->|Forwarder|-->|Forwarder|-->|   L2     | |
   +-------+  +------+----+  +---------+   +---------+   +----------+ |
     SF1           nSFF1                                     nSFF2    |
                                                 +-------+            |
                                                 | App   |/           |
                                                 | HTTP  | -----------+
                                                 | TCP   |\
                                                 | IP    |
                                                 | L2    |
                                                 +-------+
                                                   SF2
```

Figure 4: Protocol Layers

The nSFF component here is shown as implementing a full incoming/outgoing TCP/IP protocol stack towards the local SFs, while implementing the nSFF-NR and nSFF-nSFF protocols based on the descriptions in Section 6.2.3.

For the exchange of HTTP-based SF transactions, the nSFF terminates incoming TCP connections as well as outgoing TCP connections to local SFs, e.g., the TCP connection from SF1 terminates at nSFF1, and nSFF1 may store the connection information such as socket information. It also maintains the mapping information for the HTTP request such as originating SF, destination SF, and socket ID. nSFF1 may implement sending keep-alive messages over the socket to maintain the connection to SF1. Upon arrival of an HTTP request from SF1, nSFF1 extracts the HTTP Request and forwards it towards the next node as outlined in Section 6.2. Any returning response is mapped onto the suitable open socket (for the original request) and sent towards SF1.

At the outgoing nSFF2, the destination SF2/Host is identified from the HTTP request message. If no TCP connection exists to the SF2, a new TCP connection is opened towards the destination SF2 and the HTTP request is sent over said TCP connection. The nSFF2 may also save the TCP connection information (such as socket information) and maintain the mapping of the socket information to the destination SF2. When an HTTP response is received from SF2 over the TCP connection, nSFF2 extracts the HTTP response, which is forwarded to the next node. nSFF2 may maintain the TCP connection through keep-alive messages.

6.2. nSFF Operations

In this section, we present three key aspects of operations for the realization of the steps in Section 5.6, namely, (i) the registration of local SFs (for Step 3 in Section 5.6), (ii) the forwarding of SFC packets to and from local SFs (for Steps 3, 4, and 10 in Section 5.6), (iii) the forwarding to a remote SF (for Steps 5, 6, and 7 in Section 5.6) and to the NR as well as (iv) for the lookup of a suitable remote SF (for Step 7 in Section 5.6). We also cover aspects of maintaining local lookup information for reducing lookup latency and other issues.

6.2.1. Forwarding between nSFFs and nSFF-NRs

Forwarding between the distributed nSFFs as well as between nSFFs and NRs is realized over the operator network via a path-based approach. A path-based approach utilizes path information provided by the source of the packet for forwarding said packet in the network. This is similar to segment routing albeit differing in the type of information provided for such source-based forwarding as described in this section. In this approach, the forwarding information to a remote nSFF or the NR is defined as a "path identifier" (pathID) of a defined length where said length field indicates the full pathID length. The payload of the packet is defined by the various operations outlined in the following subsections, resulting in an overall packet being transmitted. With this, the generic forwarding format (GFF) for transport over the operator network is defined in Figure 5 with the length field defining the length of the pathID provided.

```
+---------+-----------------+------------------------//------------+
|         |                 |                         //           |
| Length  | Path ID         |  Payload               //            |
|(12 bits)|                 |                       //             |
+---------+-----------------+---------------------//---------------+
```

Figure 5: Generic Forwarding Format (GFF)

* Length (12 bits): Defines the length of the pathID, i.e., up to 4096 bits

* Path ID: Variable-length bit field derived from IPv6 source and destination address

For the pathID information, solutions such as those in [Reed2016] can be used. Here, the IPv6 source and destination addresses are used to realize a so-called path-based forwarding from the incoming to the outgoing nSFF or the NR. The forwarders in Figure 4 are realized via SDN (software-defined networking) switches, implementing an AND/CMP operation based on arbitrary wildcard matching over the IPv6 source and destination addresses as outlined in [Reed2016]. Note that in the case of using IPv6 address information for path-based forwarding, the step of removing the TE at the outgoing nSFF in Figure 4 is realized by utilizing the provided (existing) IP header (which was used for the purpose of the path-based forwarding in [Reed2016]) for the purpose of next-hop forwarding such as that of IP-based routing. As described in Step 8 of the extended nSFF operations, this forwarding information is used as traffic encapsulation. With the forwarding information utilizing existing IPv6 information, IP headers are utilized as TE in this case. The next-hop nSFF (see Figure 4) will restore the IP header of the packet with the relevant IP information used to forward the SFC packet to SF2, or it will create suitable TE information to forward the information to another nSFF or boundary node. Forwarding operations at the intermediary forwarders, i.e., SDN switches, examine the pathID information through a flow-matching rule in which a specific switch-local output port is represented through the specific assigned bit position in the pathID. Upon a positive match in said rule, the packet is forwarded on said output port.

Alternatively, the solution in [BIER-MULTICAST] suggests using a so-called BIER (Binary Indexed Explicit Replication) underlay. Here, the nSFF would be realized at the ingress to the BIER underlay, injecting the SFC packet header (plus the Network Service Header (NSH)) with BIER-based traffic encapsulation into the BIER underlay with each of the forwarders in Figure 4 being realized as a so-called Bit-Forwarding Router (BFR) [RFC8279].

6.2.1.1. Transport Protocol Considerations

Given that the proposed solution operates at the "named-transaction" level, particularly for HTTP transactions, forwarding between nSFFs and/or NRs SHOULD be implemented via a transport protocol between nSFFs and/or NRs in order to provide reliability, segmentation of large GFF packets, and flow control, with the GFF in Figure 5 being the basic forwarding format for this.

Note that the nSFFs act as TCP proxies at ingress and egress, thus terminating incoming and initiating outgoing HTTP sessions to SFs.

Figure 6 shows the packet format being used for the transmission of data, being adapted from the TCP header. Segmentation of large transactions into single transport protocol packets is realized through maintaining a "Sequence number". A "Checksum" is calculated over a single data packet with the ones-complement TCP checksum calculation being used. The "Window Size" field indicates the current maximum number of transport packets that are allowed in-flight by the egress nSFF. A data packet is sent without a "Data" field to indicate the end of the (e.g., HTTP) transaction.

Note that, in order to support future named transactions based on other application protocols, such as Constrained Application Protocol (CoAP), future versions of the transport protocol MAY introduce a "Type" field that indicates the type of application protocol being used between SF and nSFF with "Type" 0x01 proposed for HTTP. This is being left for future study.

```
+----------------------------------------------+
|          16 bits        |        16 bits      |
+----------------------------------------------+
|               Sequence number                 |
+----------------------------------------------+
|        Checksum         |    Window Size      |
+----------------------------------------------+
|                       ...                     |
|                  Data (Optional)              |
+----------------------------------------------+
```

Figure 6: Transport Protocol Data Packet Format

Given the path-based forwarding being used between nSFFs, the transport protocol between nSFFs utilizes negative acknowledgements from the egress nSFF towards the ingress nSFF. The transport protocol negative Acknowledgment (NACK) packet carries the number of NACKs as well as the specific sequence numbers being indicated as lost in the "NACK number" field(s) as shown in Figure 7.

```
+-----------------------+----------------------+
|          16 bits        |        16 bits      |
+----------------------------------------------+
|    Number of NACKs      |                     +
+----------------------------------------------+
|                    NACK number                |
+----------------------------------------------+
+                 ... NACK number               +
+----------------------------------------------+
```

Figure 7: Transport Protocol NACK Packet Format

If the indicated number of NACKs in a received NACK packet is nonzero, the ingress nSFF will retransmit all sequence numbers signaled in the packet while decreasing its congestion window size for future transmissions.

If the indicated number of NACKs in a received NACK packet is zero, it will indicate the current congestion window as being successfully (and completely) being transmitted, increasing the congestion window size if smaller than the advertised "Window Size" in Figure 6.

The maintenance of the congestion window is subject to realization at the ingress nSFF and left for further study in nSFF realizations.

6.2.2. SF Registration

As outlined in Steps 3 and 10 of Section 5.6, the nSFF needs to determine if the SF derived from the Name-Based Network Locator (nNLM) is locally reachable or whether the packet needs to be forwarded to a remote SFF. For this, a registration mechanism is provided for such local SF with the local nSFF. Two mechanisms can be used for this:

1. SF-initiated: We assume that the SF registers its Fully Qualified Domain Name (FQDN) to the local nSFF. As local mechanisms, we foresee that either a Representational State Transfer (REST-based) interface over the link-local link or configuration of the nSFF (through configuration files or management consoles) can be utilized. Such local registration events lead to the nSFF registering the given FQDN with the NR in combination with a system-unique nSFF identifier that is being used for path-computation purposes in the NR. For the registration, the packet format in Figure 8 is used (inserted as the payload in the GFF of Figure 5 with the pathID towards the NR).

```
+---------+-------------------+----------------+
|         |                   |                |
|   R/D   |    hash(FQDN)     |    nSFF_ID     |
| (1 bit) |    (16 bits)      |    (8 bits)    |
+---------+-------------------+----------------+
```

Figure 8: Registration Packet Format

   + R/D: 1-bit length (0 for Register, 1 for Deregister)

   + hash(FQDN): 16-bit length for a hash over the FQDN of the SF

   + nSFF_ID: 8-bit length for a system-unique identifier for the SFF related to the SF

   We assume that the pathID towards the NR is known to the nSFF through configuration means.

   The NR maintains an internal table that associates the hash(FQDN), the nSFF_id information, as well as the pathID information being used for communication between nSFFs and NRs. The nSFF locally maintains a mapping of registered FQDNs to IP addresses for the latter using link-local private IP addresses.

2. Orchestration-based: In this mechanism, we assume that SFC to be orchestrated and the chain to be provided through an orchestration template with FQDN information associated to a compute/storage resource that is being deployed by the orchestrator. We also assume knowledge at the orchestrator of the resource topology. Based on this, the orchestrator can now use the same REST-based protocol defined in option 1 to instruct the NR to register the given FQDN, as provided in the

template, at the nSFF it has identified as being the locally servicing nSFF, provided as the system-unique nSFF identifier.

6.2.3. Local SF Forwarding

There are two cases of local SF forwarding, namely, the SF sending an SFC packet to the local nSFF (incoming requests) or the nSFF sending a packet to the SF (outgoing requests) as part of Steps 3 and 10 in Section 5.6. In the following, we outline the operation for HTTP as an example-named transaction.

As shown in Figure 4, incoming HTTP requests from SFs are extracted by terminating the incoming TCP connection at their local nSFFs at the TCP level. The nSFF MUST maintain a mapping of open TCP sockets to HTTP requests (utilizing the URI of the request) for HTTP response association.

For outgoing HTTP requests, the nSFF utilizes the maintained mapping of locally registered FQDNs to link-local IP addresses (see Section 6.2.2, option 1). Hence, upon receiving an SFC packet from a remote nSFF (in Step 9 of Section 5.6), the nSFF determines the local existence of the SF through the registration mechanisms in Section 6.2.2. If said SF does exist locally, the HTTP (+NSH) packet, after stripping the TE, is sent to the local SF as Step 10 in Section 5.6 via a TCP-level connection. Outgoing nSFFs SHOULD keep TCP connections open to local SFs for improving SFC packet delivery in subsequent transactions.

6.2.4. Handling of HTTP Responses

When executing Steps 3 and 10 in Section 5.6, the SFC packet will be delivered to the locally registered next hop. As part of the HTTP protocol, responses to the HTTP request will need to be delivered on the return path to the originating nSFF (i.e., the previous hop). For this, the nSFF maintains a list of link-local connection information, e.g., sockets to the local SF and the pathID on which the request was received. Once receiving the response, nSFF consults the table to determine the pathID of the original request, forming a suitable GFF-based packet to be returned to the previous nSFF.

When receiving the HTTP response at the previous nSFF, the nSFF consults the table of (locally) open sockets to determine the suitable local SF connection, mapping the received HTTP response URI to the stored request URI. Utilizing the found socket, the HTTP response is forwarded to the locally registered SF.

6.2.5. Remote SF Forwarding

In Steps 5, 6, 7, and 8 of Section 5.6, an SFC packet is forwarded to a remote nSFF based on the nNLM information for the next hop of the nSFP. Section 6.2.5.1 handles the case of suitable forwarding information to the remote nSFF not existing, therefore consulting the NR to obtain suitable information. Section 6.2.5.2 describes the maintenance of forwarding information at the local nSFF. Section 6.2.5.3 describes the update of stale forwarding information. Note that the forwarding described in Section 6.2.1 is used for the actual forwarding to the various nSFF components. Ultimately, Section 6.2.5.4 describes the forwarding to the remote nSFF via the forwarder network.

6.2.5.1. Remote SF Discovery

The nSFF communicates with the NR for two purposes: namely, the registration and discovery of FQDNs. The packet format for the former was shown in Figure 8 in Section 6.2.2, while Figure 9 outlines the packet format for the discovery request.

```
+--------------+-------------+ +--------+-----------------//--------+
|              |             | |        |                //         |
|   hash(FQDN) |   nSFF_ID   | | Length | pathID        //          |
|   (16 bits)  |   (8 bits)  | |(4 bits)|              //           |
+--------------+-------------+ +--------+-------------//------------+
       Path Request                         Path Response
```

Figure 9: Discovery Packet Format

For Path Request:

* hash(FQDN): 16-bit length for a hash over the FQDN of the SF

* nSFF_ID: 8-bit length for a system-unique identifier for the SFF related to the SF

For Path Response:

* Length: 4-bit length that defines the length of the pathID

* Path ID: Variable-length bit field derived from IPv6 source and destination address

A path to a specific FQDN is requested by sending a hash of the FQDN to the NR together with its nSFF_id, receiving as a response a pathID with a length identifier. The NR SHOULD maintain a table of discovery requests that map discovered (hash of) FQDN to the nSFF_id that requested it and the pathID that is being calculated as a result of the discovery request.

The discovery request for an FQDN that has not previously been served at the nSFF (or for an FQDN whose pathID information has been flushed as a result of the update operations in Section 6.2.5.3) results in an initial latency incurred by this discovery through the NR, while any SFC packet sent over the same SFP in a subsequent transaction will utilize the nSFF-local mapping table. Such initial latency can be avoided by prepopulating the FQDN-pathID mapping proactively as part of the overall orchestration procedure, e.g., alongside the distribution of the nNLM information to the nSFF.

6.2.5.2. Maintaining Forwarding Information at Local nSFF

Each nSFF MUST maintain an internal table that maps the (hash of the) FQDN information to a suitable pathID. As outlined in Step 7 of Section 5.6, if a suitable entry does not exist for a given FQDN, the pathID information is requested with the operations in Section 6.2.5.1 and the suitable entry is locally created upon receiving a reply with the forwarding operation being executed as described in Section 6.2.1.

If such an entry does exist (i.e., Step 6 of Section 5.6), the pathID is locally retrieved and used for the forwarding operation in Section 6.2.1.

6.2.5.3. Updating Forwarding Information at nSFF

The forwarding information maintained at each nSFF (see Section 6.2.5.2) might need to be updated for three reasons:

1. An existing SF is no longer reachable: In this case, the nSFF with which the SF is locally registered deregisters the SF explicitly at the NR by sending the packet in Figure 6 with the hashed FQDN and the R/D bit set to 1 (for deregister).

2. Another SF instance has become reachable in the network (and, therefore, might provide a better alternative to the existing SF): In this case, the NR has received another packet with a format defined in Figure 7 but a different nSFF_id value.

3. Links along paths might no longer be reachable: The NR might use a suitable southbound interface to transport networks to detect link failures, which it associates to the appropriate pathID bit position.

For this purpose, the packet format in Figure 10 is sent from the NR to all affected nSFFs, using the generic format in Figure 5.

```
+---------+-----------------+--------------//----+
|         |                 |              //    |
|  Type   |      #IDs       |  IDs        //     |
| (1 bit) |    (8 bits)     |            //      |
+---------+-----------------+----------//--------+
```

Figure 10: Path Update Format

* Type: 1-bit length (0 for Nsff ID, 1 for Link ID)

* #IDs: 8-bit length for number of IDs in the list

* IDs: List of IDs (Nsff ID or Link ID)

The pathID to the affected nSFFs is computed as the binary OR over all pathIDs to those nSFF_ids affected where the pathID information to the affected nSFF_id values is determined from the NR-local table maintained in the registration/deregistration operation of Section 6.2.2.

The pathID may include the type of information being updated (e.g., node identifiers of leaf nodes or link identifiers for removed links). The node identifier itself may be a special identifier to signal "ALL NODES" as being affected. The node identifier may signal changes to the network that are substantial (e.g., parallel link failures). The node identifier may trigger (e.g., recommend) purging of the entire path table (e.g., rather than the selective removal of a few nodes only).

It will include the information according to the type. The included information may also be related to the type and length information for the number of identifiers being provided.

In cases 1 and 2, the Type bit is set to 1 (type nSFF_id) and the affected nSFFs are determined by those nSFFs that have previously sent SF discovery requests, utilizing the optional table mapping previously registered FQDNs to nSFF_id values. If no table mapping the (hash of) FQDN to nSFF_id is maintained, the update is sent to all nSFFs. Upon receiving the path update at the affected nSFF, all appropriate nSFF-local mapping entries to pathIDs for the hash(FQDN) identifiers provided will be removed, leading to a new NR discovery request at the next remote nSFF forwarding to the appropriate FQDN.

In case 3, the Type bit is set to 0 (type linkID) and the affected nSFFs are determined by those nSFFs whose discovery requests have previously resulted in pathIDs that include the affected link, utilizing the optional table mapping previously registered FQDNs to pathID values (see Section 6.2.5.1). Upon receiving the node identifier information in the path update, the affected nSFF will check its internal table that maps FQDNs to pathIDs to determine those pathIDs affected by the link problems and remove path information that includes the received node identifier(s). For this, the pathID entries of said table are checked against the linkID values provided in the ID entry of the path update through a binary AND/CMP operation to check the inclusion of the link in the pathIDs to the FQDNs. If any pathID is affected, the FQDN-pathID entry is removed, leading to a new NR discovery request at the next remote nSFF forwarding to the appropriate FQDN.

6.2.5.4. Forwarding to Remote nSFF

Once Steps 5, 6, and 7 in Section 5.6 are being executed, Step 8 finally sends the SFC packet to the remote nSFF, utilizing the pathID returned in the discovery request (Section 6.2.5.1) or retrieved from the local pathID mapping table. The SFC packet is placed in the payload of the generic forwarding format in Figure 5 together with the pathID, and the nSFF eventually executes the forwarding operations in Section 6.2.1.

7. IANA Considerations

This document has no IANA actions.

8. Security Considerations

Sections 5 and 6 describe the forwarding of SFC packets between named SFs based on URIs exchanged in HTTP messages. Security is needed to protect the communications between originating node and Ssff, between one Nsff and the next Nsff, and between Nsff and destination. TLS is sufficient for this and SHOULD be used. The TLS handshake allows to determine the FQDN, which, in turn, is enough for the service routing decision. Supporting TLS also allows the possibility of HTTPS-based transactions.

It should be noted (per [RFC3986]) that what a URI resolves to is not necessarily stable. This can allow flexibility in deployment, as described in this document, but may also result in unexpected behavior and could provide an attack vector as the resolution of a URI could be "hijacked" resulting in packets being steered to the wrong place. This could be particularly important if the SFC is intended to send packets for processing at security functions. Such hijacking is a new attack surface introduced by using a separate NR.

9. References

9.1. Normative References

[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>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", RFC 8279, DOI 10.17487/RFC8279, November 2017, <https://www.rfc-editor.org/info/rfc8279>.

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

9.2. Informative References

[BIER-MULTICAST]
Trossen, D., Rahman, A., Wang, C., and T. Eckert, "Applicability of BIER Multicast Overlay for Adaptive Streaming Services", Work in Progress, Internet-Draft, draft-ietf-bier-multicast-http-response-01, 28 June 2019, <https://tools.ietf.org/html/draft-ietf-bier-multicast-http-response-01>.

[Reed2016] Reed, M.J., Al-Naday, M., Thomas, N., Trossen, D., Petropoulos, G., and S. Spirou, "Stateless multicast switching in software defined networks", IEEE ICC 2016, DOI 10.1109/ICC.2016.7511036, May 2016, <https://ieeexplore.ieee.org/document/7511036>.

[Schlinker2017]
Schlinker, B., Kim, H., Cui, T., Katz-Bassett, E., Madhyastha, H., Cunha, I., Quinn, J., Hassan, S., Lapukhov, P., and H. Zeng, "Engineering Egress with Edge Fabric, Steering Oceans of Content to the World", ACM SIGCOMM 2017, August 2017, <https://research.fb.com/wp-content/uploads/2017/08/sigcomm17-final177-2billion.pdf>.

[SDO-3GPP-SBA]
3GPP, "Technical Realization of Service Based Architecture", 3GPP TS 29.500 V15.5.0, September 2019, <https://www.3gpp.org/ftp/Specs/html-info/29500.htm>.

[SDO-3GPP-SBA-ENHANCEMENT]
3GPP, "New SID for Enhancements to the Service-Based 5G System Architecture", 3GPP S2-182904, February 2018, <https://www.3gpp.org/ftp/tsg_sa/WG2_Arch/TSGS2_126_Montreal/Docs/S2-182904.zip>.

Acknowledgements

The authors would like to thank Dirk von Hugo and Andrew Malis for their reviews and valuable comments. We would also like to thank Joel Halpern, the chair of the SFC WG, and Adrian Farrel for guiding us through the Independent Submission Editor (ISE) path.

Authors' Addresses

Dirk Trossen
InterDigital Europe, Ltd
64 Great Eastern Street, 1st Floor
London
EC2A 3QR
United Kingdom

Email: Dirk.Trossen@InterDigital.com

Debashish Purkayastha
InterDigital Communications, LLC
1001 E Hector St
Conshohocken, PA
United States of America

Email: Debashish.Purkayastha@InterDigital.com

Akbar Rahman
InterDigital Communications, LLC

1000 Sherbrooke Street West
Montreal
Canada

Email: Akbar.Rahman@InterDigital.com