       Name-Based Service Function Forwarder (nSFF) component within SFC
                                framework
                   draft-trossen-sfc-name-based-sff-00

Abstract

   Many stringent requirements are imposed on today's network, such as
   low latency, high availability and reliability in order to support
   several use cases such as IoT, Gaming, Content distribution, Robotics
   etc.

   Adoption of cloud and fog technology at the edge of the network
   allows operator to deploy a single "Service Function" to multiple
   "Execution locations". The decision to steer traffic to a specific
   location may change frequently based on load, proximity etc. Under
   the current SFC framework, steering traffic dynamically to the
   different execution end points require a specific 're-chaining',
   i.e., a change in the service function path reflecting the different
   IP endpoints to be used for the new execution points. In order to
   address this, we discuss separating the logical Service Function Path
   from the specific execution end points. This can be done by
   identifying the Service Functions using a name rather than a routable
   IP endpoint (or Layer 2 address). This draft describes the necessary
   extensions to the various concepts and methods of SFC, specifically
   the SFF, in order to handle such name based relationships.

Status of This Memo

   This Internet-Draft will expire on December 29, 2018.

Copyright Notice

Table of Contents

1. Introduction

   The requirements on today's networks are very diverse, enabling
   multiple use cases such as IoT, Content Distribution, Gaming, Network
   functions such as Cloud RAN. Every use case imposes certain
   requirements on the network. These requirements vary from one
   extreme to other and often they are in a divergent direction.
   Network operator and service providers are pushing many functions

towards the edge of the network in order to be closer to the users. This reduces latency and backhaul traffic, as user request can be processed locally.

It becomes more challenging for the network when user mobility as well as non-deterministic availability of compute and storage resources are considered. The impact is felt most at the edge of the network because as users move, their point of attachment changes frequently, which results in (at least partially) relocating the service as well as the service endpoint. Furthermore, network functions are pushed more and more towards the edge, where compute and storage resources are constrained and availability is non-deterministic.

In such a dynamic network environment, the capability to dynamically compose new services from available services as well as move a service instance in response to user mobility or resource availability is desirable.

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 service functions), realized through their direct Layer 2 (e.g., 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 3 for more background on SFC.

A dynamic service environment as the one outlined above, i.e. utilizing edge-based services, requires that service end points are created and recreated frequently. Within the SFC framework, it means to reconfigure the existing chain through information based on the new relationships, causing overhead in a number of components, specifically the orchestrator that initiates the initial service function chain and any possible reconfiguration.

This document describes how such changes can be handled without involving the initiation of new and reconfigured SFCs by embedding the necessary functionality directly into the Service Function Forwarder. In other words, the SFF described here allows for keeping an existing SFC intact, as described by its service function path (SFP), while enabling the selection of an appropriate service function endpoint(s) during the traversal of packets through the SFC.

2. Example use case: 5G control plane services

   As stated above, we can formulate the problem of keeping a service
   function chain intact while providing selection of the specific
   service function endpoints that are traversed at any point in time.
   We exemplify the need for such a solution through a use case stemming
   from the current 3GPP Rel 16 work on Service Based Architecture (SBA)
   [_3GPP_SBA], [_3GPP_SBA_ENHANCEMENT]. In this work, a change in
   designing mobile network control planes is motivated 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 [_3GPP_SBA]. With this in mind,
   the exchange between, say the session management function (SMF) and
   the authentication management function (AMF) in a 5G control plane is
   being described as a set of web service like requests which are in
   turn embedded into HTTP requests. Hence, interactions in a 5G
   control plane can be modelled based on service function chains where
   the relationship is between the specific (IP-based) service function
   endpoints that implement the necessary service endpoints in the SMF
   and AMF.

   Motivating the move from a network function model (in pre-Rel 15
   systems of 3GPP) to a service-based model is the proliferation of
   data center operations for mobile network control plane services. In
   other words, typical IT-based methods to service provisioning, in
   particular that of virtualization of entire compute resources, are
   envisioned to being used in future operations of mobile networks.
   Hence, operators of such future mobile network desire to virtualize
   service function endpoints and direct (control plane) traffic to the
   most appropriate current service instance. 'Appropriate' here can be
   defined by topological or geographical proximity of the service
   initiator to the service function 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 centre centric
   operation is extended with the trend towards regionalization of load
   through a 'regional office' approach, where micro data centres
   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 're-chaining' of the SFC since the
   relationships in said SFC are defined through Layer 2 or 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).

3. Background

3.1. Basic Definitions

   We first start with few basic definitions, according to [RFC7665]:

   o A Service Function (SF) is responsible for specific treatment of
      received packets. A Service Function can act at various layers of
      a protocol stack (e.g., at the network layer or other OSI layers).
      As a logical component, a service function can be realized as a
      virtual element or be embedded in a physical network element.

   o The ordered series of Service Functions is defined through the
      Service Function Chain (SFC).

   o Service Function Path (SFP) defines the specific execution of an
      SFC in specific Service Function instances. The SFP reflects the
      mechanism used by service chaining to express the result of
      applying more granular policy and operational constraints to the
      abstract requirements of a service chain (SFC).

   o The SFP is identified by the Service Path Identifier (SPI) and the
      Service index (SI).

   o A Network Locator Map exists at each SFF that translates the SPI
      and SI information into a Next Hop information within the SFC
      domain.

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

Figure 1: Network Locator Map in SFC

o Network Service Header (NSH) [RFC8300] is a distinct identifiable
   plane that can be used across all transports to create a service
   chain and exchange metadata along the chain. A Network Service
   Header (NSH) contains service path information and optionally
   metadata that are added to a packet or frame and used to create a
   service plane. A control plane is required in order to exchange
   NSH values with participating nodes, and to provision the same
   nodes with requisite information such as service path ID to
   overlay mapping

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Service Path Identifier (SPI)        | Service Index |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
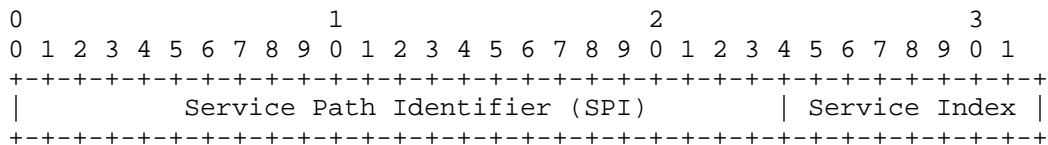
Figure 2: NSH Service Path Header

o Service Classifier / Service Classification is responsible for
   directing incoming packets from a user or another network towards
   an identified SFP, therefore forwarding the packet to the
   respective service functions specified in the SFP. For this, the
   service classifier adds the NSH header, which contains the
   respective (SPI, SI) to identify the Service Function Path.
   Additional context information such as UserID, ApplicationID maybe
   added. Finally, the service classifier determines the suitable
   transport and adds transport encapsulation for delivery to the
   SFF.

   o Service Function Forwarder (SFF at the router) executes the
      following functions:

     * Remove Transport encapsulation

     * Inspect NSH header to find SPI/SI, determine the next SF1,
       which is the Firewall function in our example above.

     * Forward the Packet and NSH encapsulation to appropriate SF,
       here SF1 in our example

3.2. Chaining of Service Functions

When a traffic flow is forwarded over a service chain, packets in the
traffic flow are processed by the various service function instances,
with each service function instance applying a service function
(e.g., firewall, network access translation (NAT), deep packet
inspection (DPI), etc.) 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 an Underlay network, possibly IP
and any Layer 2 technology. At the service layer, Service Functions
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 service function is running.

As an example, a user's request to access a video server from his
home over a fixed network, may flow through several service functions
over a service chain, as deployed by network operator. The CPE
accepts the request, pass it on through a firewall, video optimizer
and video server. These service functions define a Service Function
Chain. The following diagram in Figure 3 shows the example Service
Function Chain described here.

```
+------+   +------+    +----------+    +-----------------+
| User |-->| CPE  |--->| Firewall |--->| Video Optimizer |--->
+------+   +------+    +----------+    +-----------------+
```
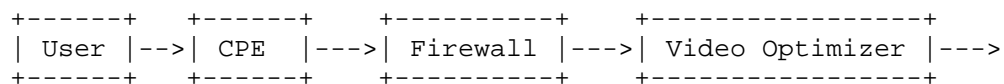
Figure 3: Mapping SFC onto Service Function Execution Points along a
Service Function Path

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

3.3. Operations for Traversing SFCs

   The following diagram in Figure 4 fits the architectural concepts and
   operation of SFC to the use case described above. How packets are
   encapsulated and de-capsulated, as it traverses through the service
   functions, is also shown.

   The CPE may act as a Service Classifier. The Service Function
   Forwarder (SFF) is the router or gateways in the network. SF1
   represents the Firewall function. SF2 represents Video Optimization
   function and Video server function.

```
             +------------+   +------+     +-----+
             | Controller |   | SF1  |     | SF2 |
             +----|-------+   +--/\--+     +-/\--+
    Configuration|--------------||-------     ||
                 |        \     ||       \    ||
       +-----------+   \+------+      \+------+
------>| Classifier |--->| vTOR |----->| vTOR |------>
Packet +------------+ |  | SFF  | |    | SFF  | Packet
                    |  +------+ |    +------+
             +--------+   +--------+
             | L2 HDR |   | L2 HDR |
             +--------+   +--------+
             | NSH HDR|   | NSH HDR|
             +--------+   +--------+
             | Packet |   | Packet |
             +--------+   +--------+
```
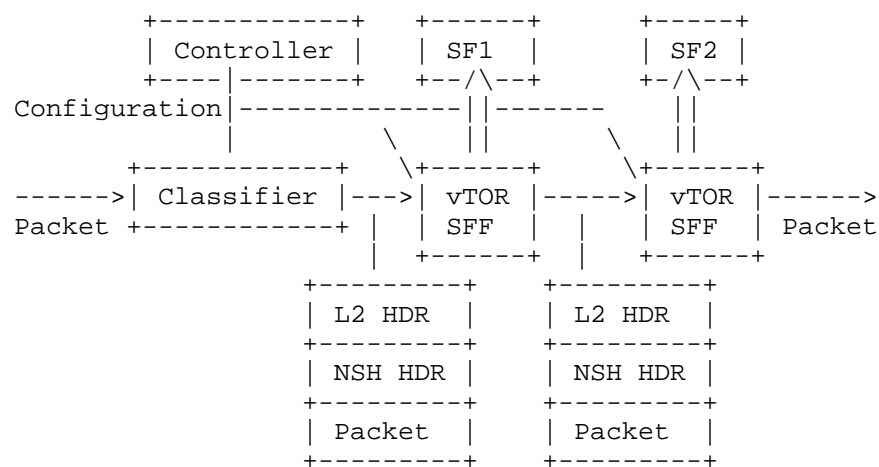
           Figure 4: Processing of an SFC Packet along an SFP

3.4. Operations inside an SFF

   The following diagram in Figure 5 describes the functions inside an
   SFF.

```
                          +----------+
                     ----|   SF     |/----
                     |   +----------+\---|
                     |           |       |
      +--------------|-----------------|-------------+
      | SFF          |                 |             |
      |              |                 |             |
      | +-----------+ \/ +-------------+ |  +------+  |
  ------>| Remove prev|--->| Process nsh |----->| Add  |------>
  TE+NSH | TE        |   | identify next|     | TE   |Packet
  +PKT   | +---------+   | SF          |     |      |+NSH+TE
      |                  +-------------+     +------+  |
      |          nsh+pkt              nsh+pkt          |
      +----------------------------------------------+
```
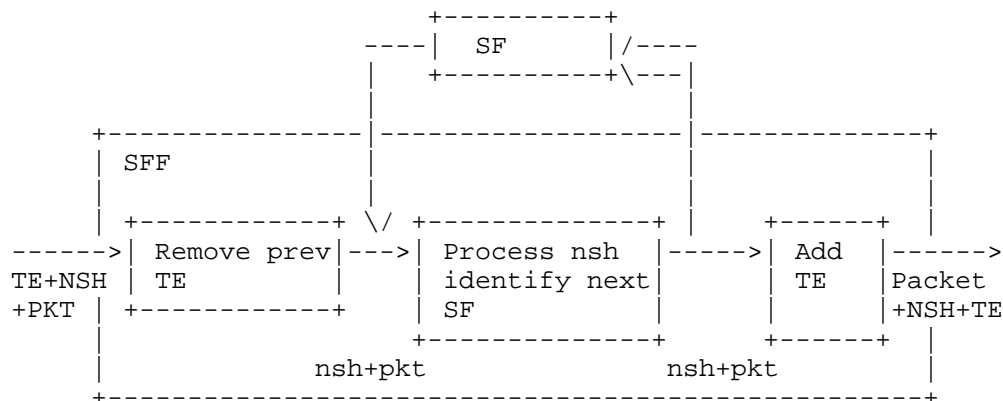
              Figure 5: Processing an incoming SFC Packet in an SFF

   Service Function Forwarder (SFF at the router) executes the following
   functions:

   o Remove TE (RTE): SFF first removes the Transport Encapsulation
      (TE) and forwards the Packet with NSH header to the next sub-
      function

   o Process NSH (PNSH):

      * Based on the NSH header, a possibly local service function is
         identified as next SF. If it is reachable from the same SFF,
         the SFF then forwards the packet. The packet is processed by
         service function. It is returned to SFF with updated NSH
         header. SFF processes the NSH header again to identify next SF
         (e.g., the Video Optimization and Delivery function in
         Figure 3. This SF is not reachable by the same SFF, it needs
         to be sent to the next SFF.

      * The "Process NSH (PNSH)" sub-function may process the following
         information: NSH header, Service Function Graph, Network
         locator map available at SFF

      * The SPI (Service Path ID) and SI (Service Index) from NSH
         header may be used to identify the current position in the
         Service Function Graph [RFC7665]. From that, PNSH function
         determines the next Service Function, which needs to be
         reached.

* As an example, let us examine the following Service Function
  Graphs available at the SFF node. Each graph may be
  identified/associated with a SPI.

```
+-------------------------------------------------------+
| ,-+-.          ,---.          ,---.          ,---.     |
| /    \        /    \        /    \        /    \      |
|(  1  )+--->(  2  )+---->(  6  )+---->(  8  )  |
| \    /        \    /        \    /        \    /      |
| `---'        `---' SPI = 10 `---'        `---'       |
+-------------------------------------------------------+
  ,-+-.          ,---.          ,---.          ,---.          ,---.
 /    \        /    \        /    \        /    \        /    \
(  1  )+--->(  2  )+---->(  3  )+---->(  7  )+---->(  9  )
 \    /        \    /        \    /        \    /        \    /
  `---'        `---'        `---'        `---'        `---'

  ,-+-.          ,---.          ,---.          ,---.          ,---.
 /    \        /    \        /    \        /    \        /    \
(  1  )+--->(  7  )+---->(  8  )+---->(  4  )+---->(  7  )
 \    /        \    /        \    /        \    /        \    /
  `---'        `---'        `---'        `---'        `---'
```
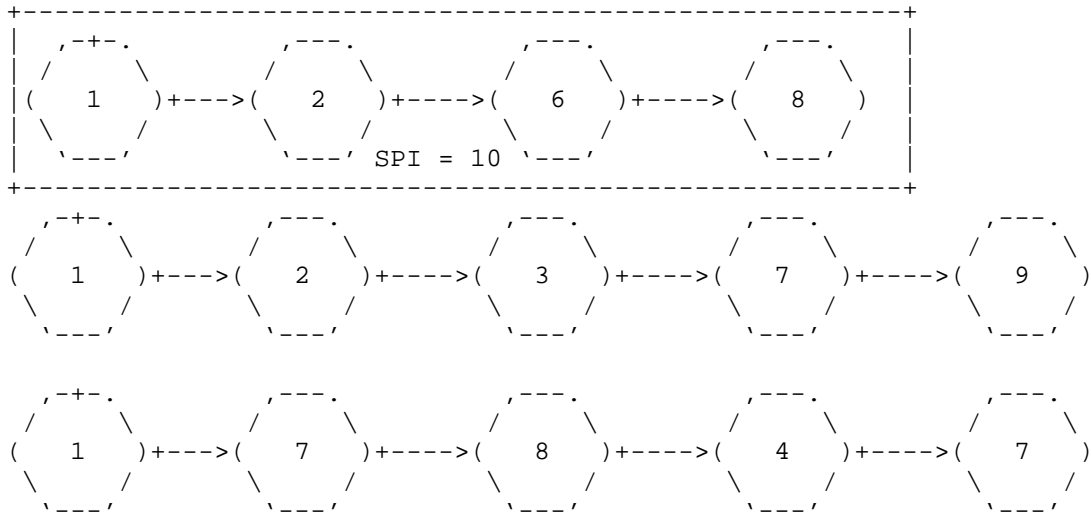
Figure 6: Example of mapping SFCs onto SPI/SI Information

* Let us assume that PNSH determines from NSH header that SPI =
  10 and SI = 254. SPI value of 10 indicates that the first
  graph (inside dotted line) in the above diagram is referenced.
  The SI value indicates the location in the graph. Generally at
  the beginning of the graph, SI is set to 255 and decremented by
  1 as the packet traverses each node. In this case, e.g. a
  value of 254 indicates that 1 nodes (e.g. Firewall function)
  have been traversed and the next SF to reach is 2 (e.g. the
  Video Optimizer / Video delivery function).

* Next SPI may be used as an index in the Network Locator Map
  [RFC8300] (see Figure 1) to identify the next hop .

* SPI = 10 and SI = 254 indicates next hop is 192.51.100.10. It
  indicates to reach the next SF, the packet needs to reach the
  host with the address and use GRE as Layer 2 encapsulation.

o Add TE (ATE): The third sub-function is to identify the Transport
   encapsulation (TE) for the Packet and NSH. The transport
   encapsulation is added to the packet and NSH. SFF puts the
   encapsulated packet in the network to be forwarded to the next SF.

Service functions (SF, SF1, ...SFn) act on an incoming packet based
on the NSH context information. E.g. SF1, i.e. Firewall function
apply policy based on UserID and decides if the packet should be
forwarded or not. The SF then decrements SI, add the NSH back into
packet and forwards to SFF. The local SFF then looks into NSH and
forwards in a similar way to next service function.

The SFC architecture [RFC7665] does not mandate how the SFP is
compiled or how SFCs are defined in general. For the compilation of
SFC, it is envisioned that the network operator identifies the
Service Functions (SFs), that a packet need to traverse, to compose
or offer a specific service. Other alternatives are the
specification of SFCs through standardization bodies, reflecting a
standardized exchange that represent a standardized service that can
be deployed in operator network.

It is envisioned that the network operator or the service provider
defines the Service Function Path for a given SFC. The specific SFP
can be decided based on policy, network deployment, services to be
offered, how a packet related to an user or a service needs to be
processed or served. Through the selection of the specific SFP for a
given SFC, the network operator has full control over how to realize
the given SFC.

For the distribution of the SFC forwarding information, i.e., the
Network Locator Map, a controller is envisioned to provide the
necessary information to the SFC nodes (i.e., the classifier and
SFF). The configuration information and provisioning may be done
using Netconf, CLI and it includes "Network Locator Map". This
configuration information may be provided to the SFC nodes during
initialization, periodic refresh or when required. Realizations of
such 'controller' could be a Northbound SDN controller application.

It is this management level distribution of configuration
information, such as the Next Hop information to the SFF nodes, that
impede flexibility by requiring any change to, for instance, Next Hop
information to be distributed to the suitable SFF node in cases of
changing the specific Service Function instance which should be used
for the next transactions traversing this SFF node. In the following
chapters, we formalize this problem and provide a solution for it.

4. Challenges with current framework

As outlined in Section 3, the Service Function Path defines an
ordered sequence of specific Service Functions instances being used
for the interaction between initiator and service functions along the
SFP. These service functions are addressed by IP addresses and

defined as next hop information in the network locator maps of
traversing SFF nodes.

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

Following the original model of SFC, any change in a specific
execution points for a specific Service Function along the SFP will
require a change of the SFP information (since the new service
function execution points 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 service function instances
is dynamic (e.g., through the use of container-based virtualization
techniques), the current model and realization of SFC reduces the
flexibility of the service providers and increases 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.

The next section outlines a solution to address the issue, allowing
for keeping SFC information (represented in its SFP) intact while
addressing the flexibility desire 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 a Service Function Path onto the level of 'name-based
interactions' rather than limiting SFPs to Layer 3 or Layer 2
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 [RFC7665]. In the following sections, we outline these
two components of our solution.

5.2. Name-Based Service Function Path (nSFP)

In the existing SFC framework [2], as outlined in Section 3, the SFP
information encapsulates forwarding information based on Layer 2 or 3
information, i.e., MAC or IP addresses. 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.
The realization of "Name based SFP" through extended SFF operations

(see Section 5.4) is illustrated using HTTP transactions. Here, URIs
are being used to name for a Service Function along the nSFP. It is
to be noted that the Name based SFP 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 . With this, our
notion of the nSFP goes beyond the initial proposals made in [7],
which limited the notion of a 'name' to a URL (uniform resource
locator), commonly used in the addressing of HTTP requests. In other
words, IP addresses as well as fully qualified domain names forming
complex URIs (uniform resource identifiers), such as www.foo.com/
service_name1, are all captured by the notion of 'name' in this
draft.

Generally, nSFPs are defined as an ordered sequence of the "name" of
Service Functions (SF) and a typical name-based Service Function Path
may look like: 192.168.x.x -> www.foo.com -> www.foo2.com/service1 ->
www.foo2.com/service2

Our use case in Section 2 can then be represented as an ordered named
sequence. An example for a session initiation that involves an
authentication procedure, this could look like 192.168.x.x ->
smf.3gpp.org/session_initiate -> amf.3gpp.org/auth -> smf.3gpp.org/
session_complete -> 192.168.x.x

In accordance with our use case in Section 2, any of these named
services can potentially be realized through more than one replicated
SF instances. This leads to make dynamic decision on where to send
packets along the SAME service function path 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 describes this name-based SFP concept and the
resulting mapping of those named interactions onto (possibly)
replicated instances.

```
+-----------------------------------------------------------------+
|SERVICE LAYER                                                    |
| 192.168.x.x --> www.foo.com --> www.foo2.com --> www.fooN.com   |
|                         ||                 ||                    |
+-------------------------||-----------------||-------------------+
                          ||                 ||
                          ||                 ||
+-------------------------||-----------------||-------------------+
| Underlay network      \/                \/                      |
|              +--+ +--+ +--+     +--+ +--+ +--+                   |
|              |  | |  | |  |     |  | |  | |  |                   |
|              +--+ +--+ +--+     +--+ +--+ +--+                   |
|                  Compute nodes     Compute nodes                |
+-----------------------------------------------------------------+
```
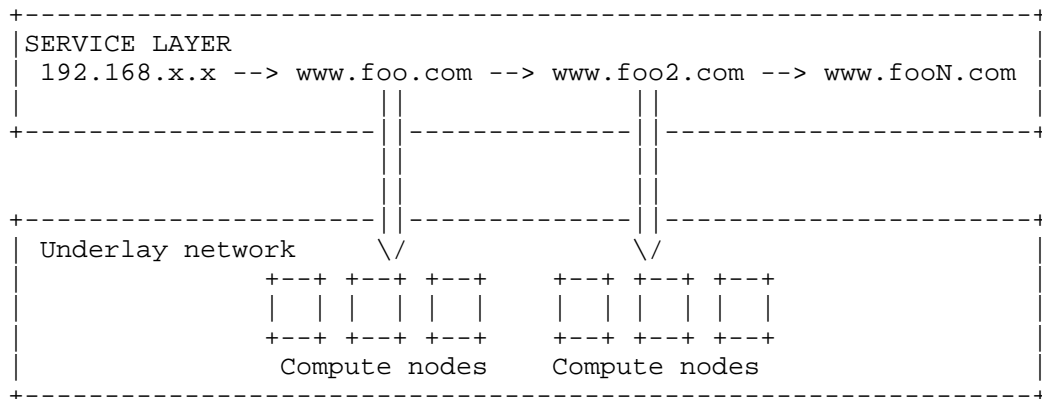
   Figure 7: Mapping SFC onto Service Function Execution Points along a
   Service Function Path based on Virtualized Service Function Instance

5.3. Name Based Network Locator Map (nNLM)

   In order to forward a name-based SFC, we need to extend the network
   locator map as originally defined in [RFC8300] with the ability to
   consider name relations based on URIs as well as high-level transport
   protocols such as HTTP for mean of packet forwarding.

   The extended Network Locator Map or name-based Network Locator Map
   (nNLM) is shown in Figure 8 as an example for www.foo.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 |
+------+------+-------------------+------------------------+
| 10   | 255  | 192.0.2.1         | VXLAN-gpe              |
|      |      |                   |                        |
| 10   | 254  | 198.51.100.10     | GRE                    |
|      |      |                   |                        |
| 10   | 253  | www.foo.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                   |
+------+------+-------------------+------------------------+
```

Figure 8: Name-based Network Locator Map

Alternatively, the extended network locator map may be defined with
implicit name information rather than explicit URIs as in Figure 8.
In the example of Figure 9 below, the next hop is represented as a
generic HTTP service without a specific URI being identified in the
extended network locator map. 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 |
+------+------+-------------------+------------------------+
| 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                  |
+------+------+-------------------+------------------------+
```

        Figure 9: Name-based Network Locator Map with Implicit Name
                              information

5.4. Name-based Service Function Forwarder (nSFF)

   While [I-D.purkayastha-sfc-service-indirection] outlined the
   realization of forwarding packets in URL-based interaction through
   HTTP via a specific function (called Service Request Routing in
   [I-D.purkayastha-sfc-service-indirection], it is desirable to extend
   the SFF of the SFC underlay in order to handle nSFPs transparently
   and without the need to insert a special (SRR) service function into
   the nSFP. Such extended name-based SFF would then be responsible for
   forwarding a packet in the SFC domain as per the definition of the
   (extended) nSFP.

   In our exemplary realization for an extended SFF, the solution
   described in this document uses HTTP transactions, i.e., HTTP as an
   example bearer protocol for the next hop information. The URI in the
   HTTP transaction are the names in our nSFP information, which will be
   used for name based forwarding, while the HTTP requests are encoded
   in plain text, e.g.,

   GET / HTTP/1.1 Host: www.foo.com Accept-Language: en

   Following our reasoning so far, HTTP requests (and more specifically
   the plain text encoded requests above) are the equivalent of Packets
   that enter the SFC domain . The handling of any intermediate layers
   such as TCP, 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 5.4.2, finalized by a layered view on the
realization of the nSFF in Section 5.4.3.

5.4.1. Lifecycle of a Packet

   With reference to the SFC architecture, the lifecycle of packet is
   shown in Figure 10 below, following the steps outlined in Section 3:

```
                                           +----------------+
                                           | NSH | HTTP GET |
           OUTSIDE           |             +----------------+
           SFC DOMAIN        |                 /\   SF
                             |                 ||
 +----------+       |   +----------+----------+ ||    +----+----------+
 | HTTP GET |       _   |          |          | _     |    |          |
 | Host:    |--|_|->| TE + NSH | HTTP GET |---|_|-->| TE | HTTP GET |
 | Accept   |       |          | HOST:    |  SFF    | +  | HOST:    |
 | Lang: en | Class |          | Acc :    |         | NSH| Acc :    |
 +----------+ ifier +----------+----------+         +---||+----------+
                                                        ||
                                                        \/
                                                    SFF _
                                                    ---|_|-->

                   --------------       ||
                   OUTSIDE SFC  +-----\/---+
                   DOMAIN       | HTTP GET |
                                | Host:    |
                                | Accept   |
                                | Lang: en |
                                +----------+
```
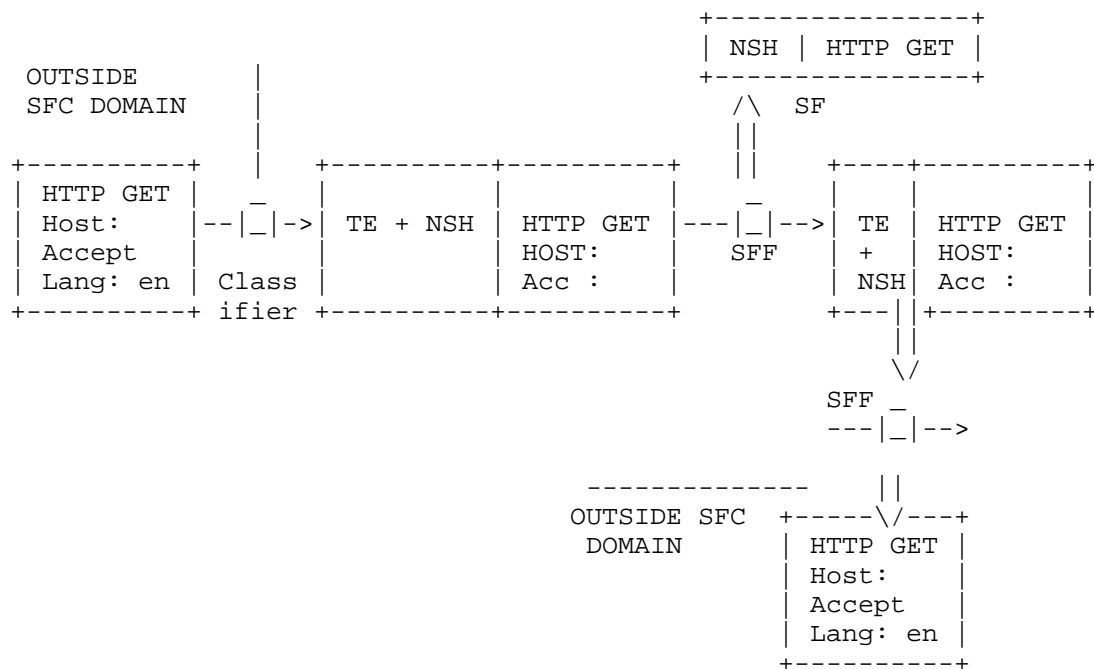
         Figure 10: Lifecycle of an SFC Packet traversing SFC Domain

   It is also possible that within an SFC domain there may be mix of
   routing decisions at the SFF. Conceptually, all routing is done by
   interpreting the SFP information as 'names'. The routing decisions
   themselves are either based on well-known, e.g., IP routing methods,
   or they utilize methods such as those outlined in [EuCNC] (for IP-
   addressed SFC packets) or [H2020POINT] (for URI-addressed SFC
   packets).

   In the proposed solution, the respective operations shown in Figure 5
   of Section 3.4 are realized by the name-based SFF, called nSFF in the
   following. The following diagram in Figure 11 describes the
   traversal between two service function instances of an nSFP-based

transactions in an example chain of: 192.168.x.x -> SF1 (www.foo.com)
-> SF2 (www.foo2.com) -> SF3 -> ...

According to the lifecycle in Figure 10 and based on our example
chain above, the following figure shows an example traffic flow in an
nSFP with nSFF and nNLM.

```
+----------+
| SF1      |                     +--------+                   +------+
| instance |\                    |   NR   |                   | SF2  |
+----------+ \                   +---/\---+                   +------+
              \                      ||                          ||
+-----------+  \ +-------+   +---------+   +---------+   +-------+
| Classifier |-->| nSFF1 |-->|Forwarder|-->|Forwarder|-->| nSFF2 |
+-----------+    +-------+   +---------+   +---------+   +-------+
                                                            ||
                                                       +----------+
                                                       | Boundary |
                                                       |  node    |
                                                       +----------+
```
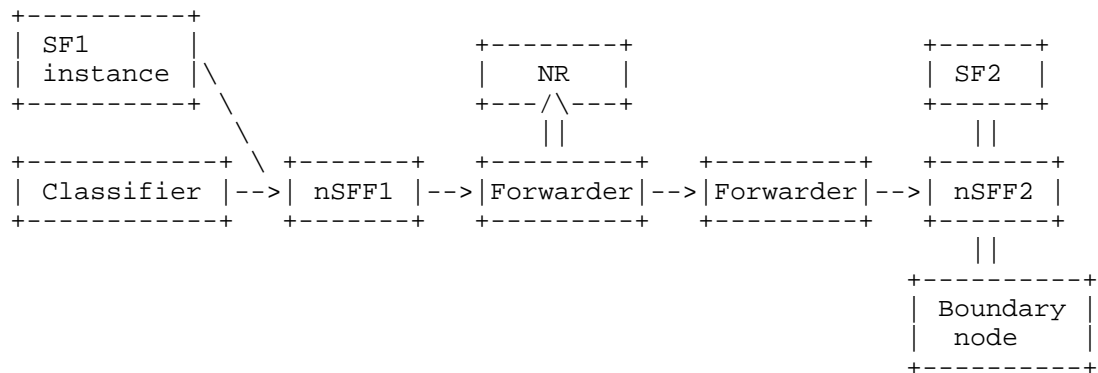
Figure 11: Operations at extended SFF

Traffic originates from a Classifier or another SFF on the left.
Traffic is processed by the incoming nSFF1 (on the left side) through
the following steps. The operations described in Figure 4 are
extended and the traffic exits at nSFF2:

o Step 1: At nSFF1 the nNLM in Figure 12 is assumed

```
+------+------+-------------------+------------------------+
| SPI  | SI   | Next Hop(s)       | Transport Encapsulation |
+------+------+-------------------+------------------------+
| 10   | 255  | 192.0.2.1         | VXLAN-gpe              |
|      |      |                   |                        |
| 10   | 254  | 198.51.100.10     | GRE                    |
|      |      |                   |                        |
| 10   | 253  | www.foo.com       | HTTP                   |
|      |      |                   |                        |
| 10   | 252  | www.foo2.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                   |
+------+------+-------------------+------------------------+
```

Figure 12: nNLM at SFF1

o 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).

o Step 3: nSFF1 then processes the NSH information to identify the
   next SF at the nSFP level by mapping the NSH information to the
   appropriate entry in its nNLM (see Figure 12) based on the
   provided SPI/SI information in the NSH (see Section 3) 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.foo.com and HTTP as the
   transport. Given the next hop resides locally, the SFC packet is
   forwarded to the SF1 instance of www.foo.com (see Figure 11).

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

o Step 5: nSSF1 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 Figure 12, to
   be www.foo2.com. Due to this SF not being locally available, the
   nSFF consults any locally available information regarding the
   routing/forwarding information towards a suitable nSFF that can
   serve this next hop.

o 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.

o Step 7: If such information does not exist, nSFF1 consults the
   Name Resolver (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 to contact
   the Name Resolver for every SFC packet forwarding. The packet is
   now marked to be sent via the network described in next step 8.

o Step 8: Utilizing the forwarding information determined in steps 6
   or 7, nSFF1 adds the suitable transport encapsulation (TE) for the
   SFC packet before forwarding via the forwarders in the network
   towards the next nSFF i.e. nSFF2.

o 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 Figure 13 is assumed. Based on
   the nNLM in Figure 13 and NSH information where SPI = 10 and SI =
   252, nSFF2 identifies the next SF as www.foo2.com.

```
+------+------+--------------------+------------------------+
| SPI  | SI   | Next Hop(s)        | Transport Encapsulation |
+------+------+--------------------+------------------------+
|      |      |                    |                        |
| 10   | 252  | www.foo2.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                   |
+------+------+--------------------+------------------------+
```

Figure 13: nNLM at SFF2

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

o Step 11: Otherwise, the outgoing nSFF adds a new TE information to
   the packet and forwards the packet (+NSH+TE) to the next SFF or
   boundary node, as shown in Figure 11.

5.4.2. Determining Suitable Forwarding Information

   The extended operations of the nSFF rely on determining suitable
   forwarding information in step 3, 5, 6 and 7 as well as adding this
   forwarding information as traffic encapsulation in steps 8 and 10 of
   the previous section. We describe details of the extended nSFF
   operations in the following sections.

5.4.2.1. DNS-based Resolution

   The SFF-Name Resolver (NR) communication in Figure 11, i.e., step 7,
   may be realized through a DNS lookup towards the next hop information
   derived from the (extended) network locator map.

   Hence, the NR is the first DNS resolver configured at the nSFF for
   any such lookup. The NR will return the suitable IP address of the
   SF instance provided in the lookup. The NR may also propagate any
   received lookup in case a NR-local setup cannot be done. Examples
   for the protocol used between nSFF and NR are the available DNS
   protocol.

   Upon receiving the NR response, the nSFF uses IP-based communication
   to send the SFC packet to the next hop, i.e., IP is being used for
   the traffic encapsulation of steps 8 and 10, while the forwarders in
   Figure 11 are standard IP routers.

   For registration of the local SF instances at each nSFF, methods such
   as management consoles (mapping IP addresses onto DNS names) or DNS
   registrations (with nSFF intercepting such registrations) can be used
   to build an nSFF-local knowledge of all locally available service
   functions at the level of a named function.

   One problem arises, however, with caching NR (i.e., Name Resolver)
   responses in step 5 since this requires dealing with stale DNS
   entries when such DNS entries might change due to new instances of
   service functions becoming available

5.4.2.2. IP/HTTP-over-ICN

   The methods in [EuCNC], [H2020POINT] describe steps to interpret IP
   or HTTP level communication from an incoming service instance
   (residing on, for instance, an IP-based device such as an UE or a
   server or a network component) as ICN (information-centric
   networking) based communication.

   Here, the NR in Figure 11 is implemented by the functions of
   Rendezvous (RV) and Topology Management (TM) in [EuCNC],

[H2020POINT]. The nSFF in Figure 11 is implemented by the methods in
the Network Attachment Point (NAP) in [EuCNC], [H2020POINT].

Hence, after having removed any transport encapsulation of the
incoming SFC packet (in case of receiving the packet from a
classifier) and the determination of an SFC packet as being sent to
another SFF (see steps of the nSFF described earlier), the nSFF
publishes a suitable ICN packet with the HTTP or IP packet (depending
on the next hop information in the nNLM) as well as the NSH as
payload.

If no suitable local forwarding information exists at the nSFF for
this publication, a path computation request is sent to the NR (i.e.,
the combination of RV and TM in [EuCNC], [H2020POINT]) and suitable
forwarding information is returned to the nSFF. This forwarding
information is used to publish the SFC packet (i.e., either an IP
packet or an HTTP plain text request, depending on the next hop
information in the nNLM) to the suitable nNLM by using this
forwarding information as traffic encapsulation towards the next
nSFF. As the next nSFF for the specific next hop information, said
nSFF has previously subscribed to the named SFP information, e.g.,
www.foo.com or the next hop IP address. It will therefore receive
the forwarded SFC packet + NSH, published in the previous step.

For the forwarding information provided by the NR in the initial
publication, 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. The forwarders in Figure 11 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] . 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 11) 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 a suitable TE information to
forward the information to another nSFF or boundary node.

Alternatively, the solution in [I-D.purkayastha-bier-multicast-http]
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 (plus the NSH) header with
BIER-based traffic encapsulation into the BIER underlay with each of
the forwarders in Figure 11 being realized as a so-called Bit-
Forwarding Router (BFR) [RFC8279].

Various registration methods can be used to bind next hop information
to a specific nSFF, such as a management console, DNS interception,
or a local registration interface. All of those methods eventually
result in a suitable subscription of the outgoing nSFF to the next
hop information, this subscription being sent to the NR, which in
turns allows the NR to determine suitable forwarding from an incoming
nSFF to said outgoing (next hop) nSFF.

5.4.3. Layered View of Realizing nSFF

In Figure 14 below, we present a layered view on realizing the nSFF
with a focus on the layers that are present at the incoming and
outgoing nSFF based on well known protocols being used but not
limiting to those only. For the sake of simplicity, we here assume
SDN being used as a Layer 2 technology although the implementation
approach equally applies to a BIER-based approach. We also show only
a sub-chain from one SF to another, leaving out the reception from
another SFF at the incoming SFF.

Here we assume the use of HTTP as the bearer protocol over the
operator (transport) network. As indicated in Section 5.2, one could
also realize named services that would normally run over a TCP-based
connection only, such as a BitTorrent type of protocol with named
chunks of content.

As also shown in the figure, IP-level SFCs can also be realized via
this implementation approach, utilizing methods such as those in
[EuCNC] by interpreting IP addresses as names. With this in mind,
the HTTP, TCP or IP level transactions originating from SF1 on the
left of Figure 12 are terminated at the incoming nSFF and then mapped
onto the Layer 2 exchange over the transport network, while the right
nSFF restores the suitable transaction and forwards the SFC packet to
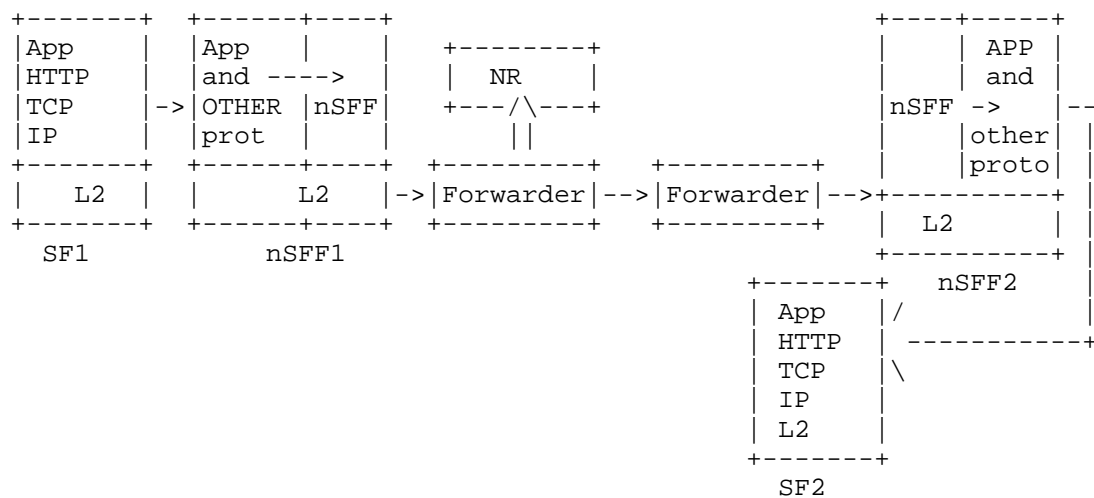SF2.

```
+-------+ +------+----+    +--------+              +----+-----+
|App    | |App   |    |    | NR     |              |    | APP  |
|HTTP   | |and ----> |    +---/\---+              |    | and  |
|TCP    |->|OTHER |nSFF|       ||                 |nSFF ->    |--
|IP     | |prot  |    |       ||                 |    |other | |
+-------+ +------+----+  +---------+  +---------+  |    |proto | |
|  L2   | |    L2   |->|Forwarder|-->|Forwarder|-->+----------+ |
+-------+ +------+----+  +---------+  +---------+  | L2       | |
   SF1          nSFF1                             +----------+ |
                                       +-------+   nSFF2       |
                                       | App   |/              |
                                       | HTTP  | -----------+
                                       | TCP   |\
                                       | IP    |
                                       | L2    |
                                       +-------+
                                          SF2
```

        Figure 14: Layered View of Realizing a Service Router as Part of
                              extended SFF

6. IANA Considerations

    This document requests no IANA actions.

7. Security Considerations

    TBD.

8. Informative References

    [_3GPP_SBA]
             3GPP, "Technical Realization of Service Based
             Architecture", 3GPP TS 29.500 0.4.0, January 2018,
             <http://www.3gpp.org/ftp/Specs/html-info/29500.htm>.

    [_3GPP_SBA_ENHANCEMENT]
             3GPP, "New SID for Enhancements to the Service-Based 5G
             System Architecture", 3GPP S2-182904 , February 2018, <htt
             p://www.3gpp.org/ftp/tsg_sa/WG2_Arch/TSGS2_126_Montreal/
             Docs/S2-182904.zip>.

    [EuCNC] Trossen, D. and et al., "POINT: IP Over ICN - The Better
             IP?", European Conference on Networks and Communications
             (EuCNC), , 2015.

   [H2020POINT]
              EU, "H2020 POINT project, First Platform Design", D 3.1,
              2015, <https://www.point-h2020.eu/wp-
              content/uploads/2015/12/
              POINT_D3.1-First_Platform_Design_v1.01.pdf>.

   [I-D.purkayastha-bier-multicast-http]
              Purkayastha, D., Rahman, A., and D. Trossen, "Multicast
              HTTP using BIER", draft-purkayastha-bier-multicast-http-00
              (work in progress), March 2018.

   [I-D.purkayastha-sfc-service-indirection]
              Purkayastha, D., Rahman, A., Trossen, D., Despotovic, Z.,
              and R. Khalili, "Alternative Handling of Dynamic Chaining
              and Service Indirection", draft-purkayastha-sfc-service-
              indirection-02 (work in progress), March 2018.

   [Reed2016]
              Reed, M., Al-Naday, M., Thomas, N., Trossen, D., and S.
              Spirou, "Stateless multicast switching in software defined
              networks", ICC 2016, 2016.

   [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>.

   [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>.

Authors' Addresses

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

   Email: Dirk.Trossen@InterDigital.com
   URI: http://www.InterDigital.com/

Debashish Purkayastha
InterDigital Communications, LLC
Conshohocken
USA

Email: Debashish.Purkayastha@InterDigital.com

Akbar Rahman
InterDigital Communications, LLC
Montreal
Canada

Email: Akbar.Rahman@InterDigital.com