SFC WG
Internet-Draft
Intended status: Informational
Expires: May 1, 2018

R. Khalili
Z. Despotovic
A. Hecker
Huawei ERC, Munich, Germany
D. Purkayastha
A. Rahman
D. Trossen
InterDigital Communications, LLC

October 28, 2017

Optimized Service Function Chaining
draft-khalili-optimized-service-function-chaining-00

Abstract

This draft investigates possibilities to use so-called 'transport-derived service function forwarders' (tSFFs) that ignore the SFC encapsulation, using instead existing transport information for explicit service path information. The draft discusses two such possibilities. In the first one, the transport network is SDN-based. The second one introduces and explains a specific service request routing (SRR) function to support URL-level routing of service requests.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright and License Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1 Introduction 3
  1.1 Terminology 3
2 SFC Forwarding Solutions 5
  2.1 Edge classification and network forwarding aggregation . . . 5
    2.1.1 Example 5
  2.2 SRR 6
3 Optimized SFC Chaining 9
  3.1 Utilizing Transport-derived SFFs 9
    3.1.1 Hierarchical addressing for service chaining 9
    3.1.2 Edge classification and service chains 10
    3.1.3 Dynamic addition of service chains 11
  3.2 Pre-Warming SFP Information for SRR-based Chaining 11
4 Applicability 15
5 Discussion 16
6 Informative References 17
Authors' Addresses 17

1 Introduction

The delivery of end-to-end network services often requires steering traffic through a sequence of individual service functions. Deployment of these functions and particularly creation of a composite service from them, i.e. steering traffic through them, had traditionally been coupled to the underlying network topology and as such awkward to configure and operate [RFC7498].

To remedy the problems identified by [RFC7498], [RFC7665] defines architecture for service function chaining that is topology independent. The architecture is predicated on a service indirection layer composed of architectural elements such as service functions (SF), service function forwarders (SFF), classifiers, etc. SFFs are the key architectural element as they connect the attached SFs and thus create a service plane.

[RFC7665] proposes SFC encapsulation as a means for service plane elements to communicate. The SFC encapsulation serves essentially two purposes. It provides path identification in the service plane (which is the primary and mandatory usage of the encapsulation) and serves as a placeholder for metadata transferred among SFs. [Quinn2017] defines NSH as a particular realization of the SFC encapsulation.

Standalone SFC encapsulation such as NSH is the mainstream SFC forwarding method with the intention to work over multiple (possibly inter-domain) transport networks. However, SFC has been identified as a suitable methodology to chain services even within single transport networks or, as outlined in [Kumar2017], even in data centers. In such cases, [RFC7665] points at the possibility of utilizing so-called 'transport-derived service function forwarders' (tSFFs) that ignore the SFC encapsulation, using existing transport information for explicit service path information.

In this document, we expand on this possibility by focusing on the realization of efficient chaining over a single transport network. In our first solution, said transport network is an SDN-based one where we represent the SFP (service function path) through a vector of aggregated flow identifiers. This solution is positioned as a tSFF between two or more SFs with no need for this solution to be SFC encapsulation aware. In our second solution, we refer to [Purka2017] which uses a specific service request routing (SRR) function to support URL-level routing of service requests. Chaining more than one SRR-connected SFs can be optimized for reducing the initial request latency, while supporting at least three different tSFFs, including the flow aggregation one presented as the first solution.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2 SFC Forwarding Solutions

2.1 Edge classification and network forwarding aggregation

Assume we are free to choose network locators (routable addresses in the considered network) for edge nodes in a network. Besides, assume that routers (switches in SDN terminology) in that network can forward packets based on wildcard matching on bit-fields in the destination address. For example, a switch somewhere in the network can forward a packet by following this logic: "the packet should be sent out the port k, because the bits 15, 16 and 17 of the destination address are 1, 0 and 1, respectively." This is possible with SDN deployments compliant e.g. with OpenFlow v1.3 and higher.

One can then come up with a multi-level classification of edge nodes, which leads to an assignment of locators to the edge nodes such that for every switch of the network the following holds:

The switch has as many forwarding rules as it has ports

For switch port k, the rule takes the form: when the destination address of the incoming packet contains a bit-field of a specific form, forward the packet to port k . For example, if the packet has 1 in the bit p of the destination address, forward to port 4.

When this is done, the network essentially becomes a fabric that delivers a packet arriving at one of its inports to the appropriate outport. It does that while maintaining the minimum internal state. [Khalili2017] explains details of the approach. In particular, it shows that large networks and networks with particular topologies require a large ID space. With that in mind, [Khalili2017] proposes an approximate method that trades node state for ID (address) space and shows that a small increase of the node state brings a large reduction or the address space (additional forwarding rules that don't follow the above form). It is this approximate method that we refer to in the rest of this section.

2.1.1 Example

Consider a simple network with an ingress (classifier) and an egress node, two transport switches/routers C1 and C2, and two service function forwarders, SFF1 and SFF2 (as depicted in Figure 1). Service functions SF1 and SF2 are attached to SFF1 and SF3 and SF4 are attached to SFF2. In this example, we assume that edge nodes are SFF1, SFF2, and the egress node.

The ASC algorithm proposed in [Khalili2017] assigns to an edge node in the network an ID of the form (v(1), v(2), ..., v(K)), where v(j), j\in[1, K], being 1 if there is a path crossing link j that ends in the corresponding edge node, and 0 otherwise. K is the size of IDs assigned to edge nodes and is an output of the algorithm.

Applying ASC algorithm to our example, we have IDSFF1 = (0, 1, 1, 0, 0), IDSFF2 = (1, 0, 0, 1, 0), and IDEgress = (1, 0, 0, 0, 1). Assuming that the destination-edge IDs are embedded in the header of the packets, e.g. via encapsulation, the forwarding rules at C1 and C2 can be aggregated by matching on bits of these IDs:

At C1: if 1st bit is 1, forward over port 3; if 2nd bit is 1, forward over port 2.

At C2: if 3rd bit is 1, forward over port 1, if 4th bit is 1, forward over port 2, if 5th bit is 1, forward over port 3.

Note from this example that each edge node has a unique ID and that we put no limitation on how SFCs are defined.

```
 _ _ _ _ _          _ _ _            _ _ _          _ _ _
|          |       |     |           |     |        |      |
|classifier|-----1-| C1  |-3-------1-| C2  |-3------|Egress|
|_ _ _ _ _ |       |_ _ _|           |_ _ _|        |_ _ _ |
                      |                 |
                      2                 2
                      |                 |
                     _|_               _|_
                    |    |            |    |
                    |SFF1|            |SFF2|
                    | _ _|            | _ _|
                     /  \              /  \
                    /    \            /    \
                  _/_    _\_        _/_    _\_
                 |SF1|  |SF2|      |SF3|  |SF4|
                 |_ _|  |_ _|      |_ _|  |_ _|
```

Figure 1: A simple topology with two SFFs and two transport switches/routers.

2.2 SRR

In [Purka2017], an extension to the Service Function Chaining (SFC)

concept is being proposed for a flexible chaining of service functions in an SFC environment, where a number of virtual instances for a single service function might exist. Hence, instead of explicitly (re-)chaining a given SFC in order to utilize a new virtual instance for an existing SF, a special service function called SRR (service request routing) is utilized to direct the requests via a URL-based abstraction (here, www.foo.com) for the SF address. As a first step, the work in [Purka2017] proposes to extend the notion of the service function path (SFP) to include such URLs in addition to already defined Ethernet or IP addresses. This is shown in Figure 1. Here the SFP includes the URLs of the service functions 1 to N (i.e., www.foo.com to www.fooN.com) as well as link-local IP addresses being used for forwarding at the local access (here shown as simple 192.168.x.x IP addresses). The creation of a suitable SFP is assumed to be part of an orchestration process, which is not within the scope of the SFC framework per se.

The SRR service function in Figure 2 can be further divided into sub-functions for realizing the dynamic chaining capabilities, as shown in [Purka2017]. Here, the service functions (such as clients and SF1 in Figure 2) communicate with local NAPs (network attachment points), while the latter communicate with the PCE (path computation element) to realize the IP and HTTP-level communication. In this case, the incoming NAP is denoted as the client NAP (cNAP) and the outgoing NAP as server NAP (sNAP). The Layer 2 transport is realized via the tSFF1 function (transport-derived service function forwarder). Here we assume that each service function is connected to an own NAP (via link-local IP communication) although one or more service functions could also reside at a single NAP.

```
                                        +--------+
                                        |        |
     |-------------------|-------------+  SRR   + <-------------|
     |                   |             |        |               |
     |                   |             +---/|\--+               |
     |                   |                  |                   |
 +--\|/--+   +----+   +-\|/-+   +----+   +--+--+   +----+   +--+--+
 |       |   |    |   |     |   |    |   |     |   |    |   |     |
 +Client +-->+SRR +-->+ SF1 +-->+SRR +-->+ SF2 +-->+SRR |-->| SFn |
 |       |   |    |   |     |   |    |   |     |   |    |   |     |
 +-------+   +----+   +-----+   +----+   +-----+   +----+   +-----+
```

Figure 2: Dynamic Chaining SFC, as proposed in [Purka2017]. SFP: 192.168.x.x -> www.foo.com -> 192.168.x.x -> www.foo2.com -> ... -> www.fooN.com

As presented in [Purka2017], the hierarchical addressing presented in

Section 3.1.1 can be utilized for the realization of said tSFF1, while other realizations could utilize SDN-based transport networks or a BIER routing layer [Wijnands2017]. With this, the SRR service function is placed in-between specific tSFFs (the three aforementioned ones) and general service functions to be chained.

3 Optimized SFC Chaining

3.1 Utilizing Transport-derived SFFs

Our model retains the architectural behavior of the SFC architecture of [RFC7665]. Yet, the SFC and the transport encapsulation are merged into the transport header. Thus everything, both transport and service plane forwarding, is happening based on transport encapsulation bits. The model builds on the edge node classification presented in Section 2.1 and comes in two flavors. The first one (Section 3.1.1) treats SFFs as edge nodes. The second one (Section 3.1.2) assigns fictitious edge nodes to entire service chains. In both cases, the key points are how we identify the service chains, and related to that, how we embed these identifiers into the available address space.

3.1.1 Hierarchical addressing for service chaining

This approach treats SFFs as edge nodes. The set of SFFs, as points of attachment of SFs, is normally static, known in advance in a network. In that sense, SFFs do not impose any stronger requirements than edge nodes, so the approach presented next looks viable.

The hierarchical service chain addressing works with the address structure (IDSFF.IDSF.IDChain), in which IDSFF identifies an SFF in the network, IDSF identifies an SF attached to that SFF, while IDChain refers to a specific chain handled by that SF. (Note that SFs can be attached to multiple SFFs, i.e. the approach is not limiting in this sense. It is rather obvious that multiple SFs can be attached to a single SFF.)

```
 _ _ _ _ _        _ _       _ _        _ _       _ _        _ _
|          |     /   \     |    |     /   \     |    |     /   \
|classifier|----| net |----|SFF1|----| net |----|SFF2|----| net |
|_ _ _ _ _ |     \_ _/     |_ _ |     \_ _/     |_ _ |     \_ _/
                            /  \                 /  \
                           /    \               /    \
                         _/_    _\_           _/_    _\_
                        |SF1|  |SF2|         |SF3|  |SF4|
                        |_ _|  |_ _|         |_ _|  |_ _|
```

Figure 3: a service chain of SF1-SF2-SF3 is considered in this example.

See Figure 2 for an example. Assume that the edge nodes in the shown

network are SFF1 and SFF2 (with possibly many other nodes which are not shown), while the service functions SF1, ..., SF4 are considered end nodes, i.e. they are not edge nodes as such do not underlie classification. (Note that this will be changed in Section 3.1.2.). Assume that the classification yields the locators (IDs) IDSFF1 and IDSFF2 for SFF1, respectively SFF2, and that a service chain SF1-SF2-SF3 has the same identifier IDChain1 at each SF (SF1, SF2, SF3). This is just for simplicity, these IDs can be different at different SFs. The service chain SF1-SF2-SF3 can operate as follows.

The classifier first adds the outer transport header with the destination address (IDSFF1.IDSF1.IDChain1). The network uses the IDSFF1 bitfield to route the packet to SFF1. SFF1 uses the middle part of the address, IDSF1, to deliver the packet to SF1. SF1, being SFC-aware, strips off the transport header and saves it, then processes the packet and, after restoring the saved transport header, sends it back to SFF1. SFF1 changes the transport header destination address to (IDSFF1.IDSF2.IDchain1) and forwards the packet to SF2. SF2 performs similar steps as SF1 and returns the packet to SFF1. SFF1 changes the transport header to (IDSFF2.IDSF3.IDChain1) and sends the packet towards SFF2. (In an SDN network, switches can manipulate with the headers by means of suitable flow rules, which should match on the (IDSF.IDChain) fraction of the destination address. A second pass through the SDN processing stack will select the appropriate port to send the packet towards SFF2.) SFF2 performs the very same sequence of steps to deliver the packet to the correct SF and then further to the network.

Note the role of the (IDChain) part of the address. That tag serves to differentiate between different service chains that pass a single SF. For example, if in addition to SF1-SF3 there is a service chain SF1-SF5, where SF5 is attached to SFF3, SFF1 will use the (IDChain) to forward packets coming from SF1.

3.1.2 Edge classification and service chains

Continuing with the classification discussion from Section 3.1.1, let us assign a fictitious edge node to a service chain under consideration. More precisely, let us assign one such node to every subsequence of the chain that starts at each possible position in the chain and goes until its end. For example, for a chain SF1-SF2-SF3, define three such nodes for sub-chains SF1-SF2-SF3, SF2-SF3 and SF3. Let locators of these fictitious edge nodes be the SFs that start the corresponding sub-chains. So, in the example, the locators are SF1, SF2 and SF3. If we had another chain that goes over SF1, then we would simply add another node, say SF1', and attach it to SFF1, next to SF1. This is to indicate that we need a distinct locator for each chain that goes over SF1. So we now have the starting network and

additional imaginary edge nodes which topologically coincide with existing service functions but require additional, separate classification vectors.

Assume that, after the classification as described in Section 3.1.1, we generate locators (classification vectors) IDSF1, IDSF2 and IDSF3 for the chain SF1-SF2-SF3 and setup rules (e.g. OpenFlow compliant) that:

At SFF1:

Forward to SF1 packets with destination IDSF1, that come from the network.

Replace IDSF1 with IDSF2 and then forward to SF2 packet that arrive from SF1.

Replace IDSF2 with IDSF3 and then forward to SFF2.

At SFF2:

Forward to SF3 packets with destination IDSF3 that come from the network.

We can distinguish between the packets that are received from the network and those received from SF1 by using the inport information.

3.1.3 Dynamic addition of service chains

The method just described assumes that all service chains are known in advance, before the classification. That assumption is not realistic, i.e. presents a strong, undesired constraint. This section will in a future version discuss how we relax that assumption, how we handle dynamic additions of service chains, etc.

3.2 Pre-Warming SFP Information for SRR-based Chaining

One issue when chaining service functions utilizing the SRR function is the initial delay incurred through the necessary path computation for a new service segment along the overall service function path. For instance, when the service function 'client' residing at the first SRR in Figure 1 issues a request to foo.com, i.e., the URL for the second service function, the NAP sub-function will trigger a PCE request for path resolution within the Layer 2 transport network. Such PCE request incurs said delay for the initial request while all subsequent requests along the same path are likely going to use locally cached information at the SRR function (we here assume but do

not detail suitable path information update procedures being implemented by the SRR sub-functions in case of path changes to another service function).

It is reasonable to assume that SFPs can be established across the realm of more than one PCE, e.g., each administering one administrative domain. However, in the case of a single PCE across a number of SRR functions, Figure 2 can be redrawn as follows.

```
                                          +--------+
                                 -----------| SRR    |-------------
                                 |          +--------+             |
                                 |             |                   |
                                 |             |                   |
                                 |             |                   |
                              +------+     +------+    +-----+   +-----+
                              | SF1  |     | SF2  |----| SRR |---| SFn |
                              +------+     +------+    +-----+   +-----+
                                 |             |
             +--------------------|-------------|---------+
             |                   |             |         |
  +-------+  |  +-------+    +--------+     +--------+   |
  |Client |-----| NAP1  |    | NAP2   |     | NAP3   |   |
  +-------+  |  +-------+    +--------+     +--------+   |
             |     \              |          /           |
             |      \             |         /            |
             |       \        +-------+    /             |
             |        \-------| tSFF1 |----              |
             |                +-------+                  |
             |                    |                      |
             |                    |                      |
             |                +-------+                  |
             |                | PCE   |                  |
             |                +-------+                  |
             +-------------------------------------------+
```

Figure 4. Decomposed Dynamic Chaining SFC across two or more SFCs.

Here, two SRR functions utilize the same PCE, e.g., within a single transport network. In this case, we propose to reduce such initial

chaining delay by virtue of a 'pre-warming' of the SRR sub-functions, specifically the incoming NAP at the suitable SRR along the SFP. For this, we require a communication of the NSH and therefore the SFP information to the PCE - such communication is subject to a standardized protocol based on a trigger that led to the formation of said SFP, as shown in Figure 4. Once such SFP information has been received by the PCE, it then executes the following procedure.

FOR ALL SF requests routed via an SRR served by the PCE:

1. Determine the incoming NAP of the first SF request, e.g., 192.168.x.x in Figure 2.

2. Determine the outgoing NAP of the service endpoint address at the outgoing SF, e.g., www.foo.com in Figure 2.

3. Compute path between incoming NAP and outgoing NAP - path computation might include a policy constraint, such as shortest path or shortest delay.

4. Deliver path information to incoming NAP.

END FOR

Figure 5 outlines the messages being exchanged between the joint PCE and the various NAPs of the SRR function. The exact nature of the messages is subject to standardization and not shown at this stage of the draft.

```
          Trigger         PCE        NAP1        NAP2      NAP3
              |             |          |           |         |
              |             |<----a----|           |         |
              |             |<-----------a---------|         |
              |             |<------------------------a-------|
              |             |          |           |         |
        --b-->|\            |          |           |         |
              | c           |          |           |         |
              |/            |          |           |         |
              |------d----->|\         |           |         |
              |             | e        |           |         |
              |             |/         |           |         |
              |             |----f---->|           |         |
              |             |------------f-------->|         |
              |             |---------------------------f---->|
              |             |          |           |         |
```

Figure 5. Message Sequence Chart Resulting in Pre-Warming of Routing Entries. a) subscribe to pre-warming information, b)initiate service chaining based on external mgmt. trigger, c) compute SFP, d) send SFP, e) map SFP information onto paths from incoming to ongoing NAPs, f) push path information with forwarding/path identifier and URL.

4 Applicability

This draft investigates whether transport encapsulation can be used for service function chaining. The main message it delivers is that this seems possible. This was demonstrated on an example of underlying SDN network.

However, we are not normative here with respect to what transport encapsulation and which bits thereof are used for service function chaining, i.e. which existing transport encapsulations give us the needed features (e.g. said assignment of transport identifiers and their handling at transport nodes) to successfully incorporate service chaining. This will be a subject of future investigations.

5 Discussion

Transport-derived SFC forwarding is related to a number of advantages. In particular, easier deployment of service chaining, as SFs and SFFs in a transport-derived chaining do not have to be SFC encapsulation aware. Further discussion will be included in future version on specific advantages and downsides of individual investigated methods, from Section 2 and Section 3.

6 Informative References

[RFC7498] P. Quinn, et al., "Problem Statement for Service Function Chaining", RFC 7498 (INFORMATIONAL), April 2015.

[RFC7665] Joel Halpern, et al., "Service Function Chaining (SFC) Architecture", RFC 7665 (INFORMATIONAL), October 2015.

[Quinn2017] P. Quinn, et al., "Network Service Header", IETF draft, draft-ietf-sfc-nsh-27 (work in progress), October 2017.

[Kumar2017] S. Kumar, et al., "Service Function Chaining Use Cases In Data Centers", IETF draft, draft-ietf-sfc-dc-use-cases-06 (work in progress), February 2017.

[Khalili2016] R. Khalili, et al., "Reducing State of OpenFlow Switches in Mobile Core Networks by Flow Rule Aggregation" IEEE ICCCN 2016.

[Purka2017] D. Purkayastha, et al., "Use Case for Handling of Dynamic Chaining and Service Indirection", IETF draft, draft-purkayastha-sfc-service-indirection-00 (work in progress), July 2017

[Wijnands2017] IJ. Wijnands, et al., "Multicast using Bit Index Explicit Replication", IETF draft, draft-ietf-bier-architecture-08 (work in progress), September 2017

Authors' Addresses

Ramin Khalili
Huawei ERC, Munich, Germany
Email: Ramin.khalili@huawei.com

Zoran Despotovic
Huawei ERC, Munich, Germany
Email: Zoran.Despotovic@huawei.com

Artur Hecker
Huawei ERC, Munich, Germany
Email: Artur.Hecker@huawei.com

Debashish Purkayastha
InterDigital Communications, LLC

Conchoken, USA
Email: Debashish.Purkayastha@InterDigital.com

Akbar Rahman
InterDigital Communications, LLC
Montreal, Canada
Email: Akbar.Rahman@InterDigital.com

Dirk Trossen
InterDigital Communications, LLC
London, United Kingdom
Email: Dirk.Trossen@InterDigital.com