        Service Function Chaining: Chain to Path Reduction
                    draft-parker-sfc-chain-to-path-00

Status of this Memo

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

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF), 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/ietf/1id-abstracts.txt

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

   This Internet-Draft will expire on May 8, 2014.

Copyright Notice

Abstract

   The service function chaining concept differentiates between a
   service function chain and a service function path.   A chain is a
   sequence of service function types whereas a path is a sequence of
   service function instances that realize each type of service
   function.

   This document describes the architectural approach to reducing the
   abstract service function chain to the concrete service function
   path.

Table of Contents

1. **Introduction**

1.1. **Service Function Chaining Concepts**

   The service function chain is an abstract sequenced set of service
   functions, as described in [I-D.boucadair-sfc-framework]. Service
   functions that participate in SFC are assumed to be transparent
   "midbox" service functions, where transparent in this usage denotes
   only that the service function is not explicitly addressed by either
   communications endpoint.

   The service function locator, also described in [I-D.boucadair-sfc-
   framework], is the means to bind the abstract service functions to
   physical or virtual instantiations.   The completed mapping of a
   service function chain through the locators of its constituent
   service functions is called a service function path, also described
   in [I-D.boucadair-sfc-framework].

   A service function may have a single locator, or multiple locators.
   When multiple locators are specified, a load balancing operation is
   required to select which one of the instances shall be used for any
   particular service function path. This document describes an
   approach where the service function path is rendered in real time,
   on a packet by packet basis, from the service function chain.

1.2. **Scope**

   This document defines a technique whereby service function chains
   may be rendered in real time as service function paths.   This
   document does not make any assumptions regarding the behavior of any
   particular service function or the appropriate technique (i.e.,
   stateless, stateful, etc.) to load balance for a given service
   function.

   Load balancing across multiple locators may involve liveness
   checking or load checking.   No assumptions are made regarding the
   appropriate technique to determine the liveness or load for a
   particular service function.   Such load balancing may also involve
   other policies, such as location of the service function instance
   relative to the location of one or both communication endpoints.
   No assumptions are made regarding the use or type of policy to
   perform load balancing for a service function.

1.3. Objectives

. SFC classifier (PDP) actions identical for SFC-aware and non-SFC-
  aware service functions
. SFC classifier (PDP)_actions identical for singly-located and
  multiply-located service functions
. SFC classifier (PDP) actions identical for any service function
. Preceding service function actions identical for SFC-aware and
  non-SFC-aware succeeding service functions
. Preceding service function actions identical for singly-located
  and multiply-located succeeding service functions
. Preceding service function actions identical for any succeeding
  service function
. SFC does not specify or limit mechanisms and behaviors for load
  balancing within a service function

1.4. Rationale

SFC proxy functionality for non-SFC aware service functions and load
balancing for multiply-located service functions (SFC-aware and non-
SFC-aware) belongs to the service function domain.   It is not
possible nor is it desirable to specify, in a practical manner,
either proxy functionality or load balancing for any arbitrary
service function.

2. Conventions used in this document

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].

In this document, these words will appear with that interpretation
only when in ALL CAPS. Lower case uses of these words are not to be
interpreted as carrying RFC-2119 significance.

In this document, the characters ">>" preceding an indented line(s)
indicates a compliance requirement statement using the key words
listed above. This convention aids reviewers in quickly identifying
or finding the explicit compliance requirements of this RFC.

An abstract service function (i.e., part of a service chain
definition) is denoted as '<F>'.

A concrete service function instance (i.e. a service function that
has been located) is denoted as '<F>.<n>'.

An SFC service function proxy, as will be defined later in this document, is denoted as '<F>.p'.

A service function load balancer is denoted as '<F>.lb'.

A combined service function load balancer and proxy, as will be defined later in this document, is denoted as '<F>.plb'.

Unless otherwise specified, the term "Service Function Chain" shall be more broadly interpreted as "Service Function Graph".   It is understood that any service function may include a reclassifier (i.e., Policy Decision Point) in order to make dynamic branching decisions along the service function graph.

[3](#). **New SFC Functional Entities**

The following SFC functional entities are in addition to those identified in [[I-D.boucadair-sfc-framework](#)].

[3.1](#). **SFC Proxy**

An SFC proxy front-ends a legacy service function that is not SFC-aware.

[3.1.1](#). **SFC Proxy for Non-last Service Function**

When receiving packets from the preceding service function (or the PDP if this service function is first), the SFC proxy terminates both the transport and SFC service headers. The resulting packets are forwarded to the legacy service function.   The legacy service function must be provisioned to return the packets back to the SFC proxy.

When receiving returned packets from the legacy service function, the SFC proxy generates appropriate transport and SFC service headers towards the succeeding service function.

It is permissible for an SFC Proxy to also act as a reclassifier, thereby making a branching decision on a service function graph.

[3.1.2](#). **SFC Proxy for Last Service Function**

It shall be an implementation decision as to whether a non-SFC-aware service function that is the last service function in the chain returns the packets to its proxy or send the packets beyond the end of the chain directly.

[3.2](#). **Service Function Load Balancer**

A service function load balancer manages all load balancing duties for its constituent SFC-aware service function instances.   How this is accomplished for the given service function is completely outside the scope of this document.

The service function load balancer appears to preceding services, or the PDP if the service function is the first in the chain, as a single entity that receives all traffic for that service function within the particular chain.   Thus, preceding services and the PDP, itself, never concern themselves with the fact that the succeeding service may have multiple locators.

   Since the service function load balancer is managing SFC-aware
   service function instances, it is not, itself eligible to be a
   reclassifier (Policy Decision Point).

[3.2.1](). Proxy Service Function Load Balancer

   The proxy service function load balancer is a combination of the
   service function proxy and the SFC-aware load balancer. Since it
   terminates the transport and SFC service headers, the proxy service
   function load balancer is eligible to be a reclassifier (Policy
   Decision Point).

[4](). Service Function Path Topologies

   This section provides a number of example topologies that
   demonstrate the real-time reduction of the service function chain to
   a specific service function path.

[4.1](). Chain Comprised of Singly-located Service Functions

   This topology is the simplest since the chain and path have a 1:1
   correspondence. An example service function path comprised solely
   of singly-located, SFC-aware service functions is depicted below.

```
      +-----+       +-----+       +-----+       +-----+
 ----->| PDP |----->| A.1 |----->| B.1 |----->| C.1 |----->
      +-----+       +-----+       +-----+       +-----+
```

[4.1.1](). With Non-SFC-aware Service Functions

   In this use case, the service functions require an SFC-proxy to act
   as front-end.

```
      +-----+       +-----+       +-----+       +-----+
 ----->| PDP |----->| A.p |----->| B.p |----->| C.p |----->
      +-----+       +-----+       +-----+       +-----+
                     | ^           | ^           | ^
                     | |           | |           | |
                    +-----+       +-----+       +-----+
                    | A.1 |       | B.1 |       | C.1 |
                    +-----+       +-----+       +-----+
```

[4.2]. Chain Comprised of Multiply-located Service Functions

This use case supports load balancing across multiple service
function instances. For clarity, each load balancer is shown as
having picked a particular service instance. The means by which
this selection is made is outside the scope of this document.

```
    +-----+       +------+           +------+           +------+
--->| PDP |----->| A.lb |      +-->| B.lb |      +-->| C.lb |
    +-----+       +------+      |    +------+      |   +------+
                     |          |   |             |    |
                     |   +-----+ |   |   +-----+ |    |   +-----+
                     +-->| A.1 |-+   +   | B.1 | |    +   | C.1 |
                     |   +-----+      |   +-----+ |    |   +-----+
                     |                |          |    |
                     |   +-----+      |   +-----+ |    |   +-----+
                     +   | A.2 |      +-->| B.2 |-+   +   | C.2 |
                         +-----+          +-----+      |   +-----+
                                                       |
                                                       |   +-----+
                                                       +-->| C.3 |--->
                                                           +-----+
```

As can be seen from this diagram, the preceding node (PDP or service
function) always propagates packets to the responsible load balancer
for the next service function in the chain.   The load balancer
makes a service function appropriate selection for the packet and
forwards to one of the service function instances.   Since the
service function instances are SFC-aware, they directly forward the
packet to the next service function load balancer, or out of the
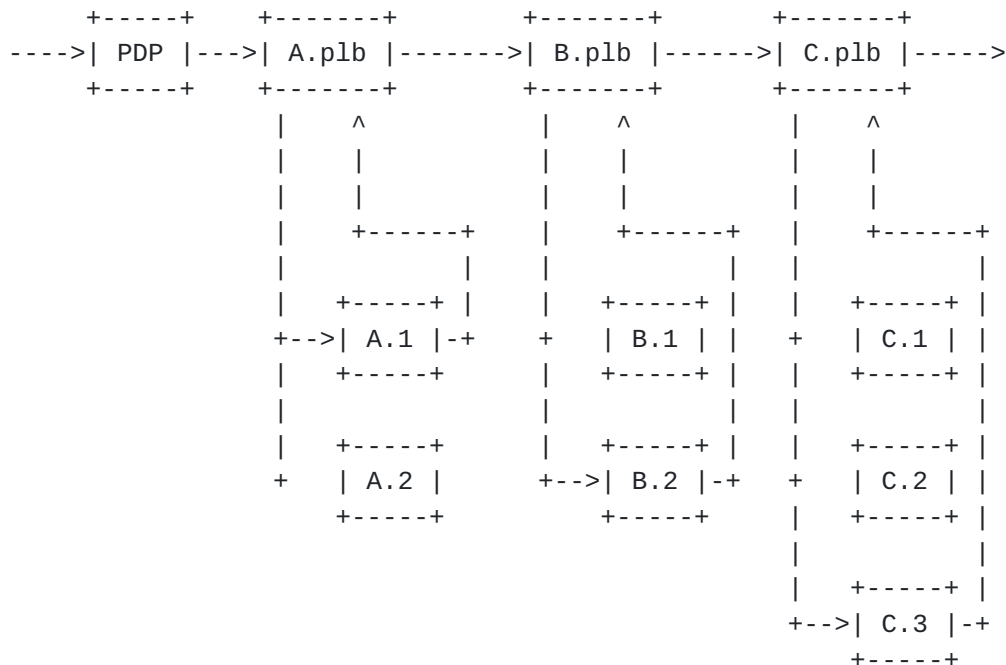chain entirely, if the last service function in the chain.

4.2.1. With Non-SFC-aware Service Functions

This use case could be achieved by front-ending each service
instance with an SFC proxy.   Such an approach is entirely valid.
However, an alternative approach is also supported by use of the
proxy service function load balancer.

```
     +-----+    +-------+         +-------+         +-------+
---->| PDP |--->| A.plb |------->| B.plb |------>| C.plb |----->
     +-----+    +-------+         +-------+         +-------+
                  |    ^            |    ^            |    ^
                  |    |            |    |            |    |
                  |    |            |    |            |    |
                  |    +------+     |    +------+     |    +------+
                  |         |     |         |     |         |
                  |   +-----+ |     |   +-----+ |     |   +-----+ |
                  +-->| A.1 |-+     +   | B.1 | |     +   | C.1 | |
                  |   +-----+       |   +-----+ |     |   +-----+ |
                  |                 |         | |     |         |
                  |   +-----+       |   +-----+ |     |   +-----+ |
                  +   | A.2 |       +-->| B.2 |-+     +   | C.2 | |
                      +-----+           +-----+       |   +-----+ |
                                                      |         |
                                                      |   +-----+ |
                                                      +-->| C.3 |-+
                                                          +-----+
```

It can be seen that this diagram differs from the previous in that
the service functions must return the packets to their respective
proxy/load balancer for appropriate transport and SFC service header
encapsulation.

4.3. Hybrid Chains

A chain of service functions may support any combination of service
functions that are singly-located SFC-aware, load-balanced SFC-
aware, singly-located proxied, and load-balanced proxied.

[5](). Security Considerations

All SFC entities must be protected against DDoS attacks, malformed
packets, and poorly configured (deliberately or not) service chains
and service function locator tables.

[6](). IANA Considerations

There are no IANA considerations for this document.

[7](). Conclusions

By strictly layering SFC chaining, SFC proxy, and load balancing, a
simpler more resilient end-to-end service function chaining
architecture is achieved.   Handling of failed service function
instances as well as static and dynamic expansion and contraction of
the service function is localized to an entity that has been
designed specifically for that type of service function.

Both preceding service functions (or PDP) and succeeding service
functions are completely unaware of whether a service function is
SFC-aware, multiply instantiated, both, or neither.

This architecture provides a unified approach for incorporation of
multiple instances of SFC-aware service functions, for incorporation
of non-SFC-aware legacy service functions, and for incorporation of
multiple instances of non-SFC aware legacy service functions.

8. References

8.1. Normative References

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
             Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

   [I-D.boucadair-sfc-framework]
           M. Bouadair, C. Jacquenet, R. Parker, D. Lopez, J. Guichard,
           C. Pignataro, "Service Function Chaining: Framework &
           Architecture", http://tools.ietf.org/html/draft-boucadair-
           sfc-framework-00 (work in progress), October 3, 2013

   [I-D.quinn-sfc-nsh]
           P. Quinn, et. al., "Network Service Header",
           http://tools.ietf.org/html/draft-quinn-sfc-nsh-00 (work in
           progress), October 7, 2013

9. Acknowledgments

   This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

   Ron Parker
   Affirmed Networks
   Acton, MA 01720
   USA

   Email: ron_parker@affirmednetworks.com