Internet Engineering Task Force (IETF)
Internet-Draft
Intended status: Informational
Expires: October 6, 2016

M-K. Shin
ETRI
M-J. Choi
KNU
S. Lee
ETRI

April 4, 2016

Yang Data Model for Service Function Chaining Control Plane
draft-shin-sfc-control-plane-yang-01

Abstract

This document defines Yang data model for control plane management of service function chaining based on [I-D.ietf-sfc-control-plane], which describes components and requirements of SFC control plane.

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). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on October 6, 2016.

Copyright Notice

Table of Contents

1. Introduction

Service Function Chaining (SFC) consists of SFC data plane and control plane from the aspect of architecture [I-D.ietf-sfc-architecture]. The document [I-D.ietf-sfc-control-plane] describes requirements for delivering information between SFC control elements and SFC functional elements. By capturing the information conveyed via a set of control interfaces in [I-D.ietf-sfc-control-plane], this document defines Yang data model of management operations performed in a SFC control plane. Note that the base Yang data model for the SFC data plane is already covered in [I-D.penno-sfc-yang].

2. Terminology

This document uses the following terms and most of them were reproduced from [I-D.ietf-sfc-architecture] and [I-D.ietf-sfc-control-plane].

- Classification: Locally instantiated policy and customer/network/service profile matching of traffic flows for identification of appropriate outbound forwarding actions.

- Service Function Chain (SFC): A service function chain defines a set of abstract service functions and ordering constraints that must be applied to packets and/or frames selected as a result of classification. The implied order may not be a linear progression as the architecture allows for SFCs that copy to more than one branch, and also allows for ere there is flexibility in the order in which service functions need to be applied. The term service chain is often used as shorthand for service function chain.

- Service Function (SF): A function that 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. One or multiple Service Functions can be embedded in the same network element. Multiple occurrences of the Service Function can exist in the same administrative domain.

- Service Function Instance (SFI): The instantiation of Service Function that can be a virtual instance or be embedded in a physical network element. One of multiple Service Functions can be embedded in the same network element. Multiple instances of the Service Function can be enabled in the same administrative domain.

- Service Function Forwarder (SFF): A service function forwarder is responsible for delivering traffic received from the network to one or more connected service functions according to information carried in the SFC encapsulation, as well as handling traffic coming back from the SF. Additionally, a service function forwarder is responsible for delivering traffic to a classifier when needed and supported, mapping out traffic to another SFF (in the same or different type of overlay), and terminating the SFP.

- Service Function Path (SFP): The SFP provides a level of indirection between the fully abstract notion of service chain as a sequence of abstract service functions to be delivered, and the fully specified notion of exactly which SFF/SFs the packet will

visit when it actually traverses the network. By allowing the control components to specify this level of indirection, the operator may control the degree of SFF/SF selection authority that is delegated to the network.

3. Overall Module Structure

Ahead to defining separate sub-modules, overall module structure for the management of SFC control plane is examined. The overall modules consist of 8 sub-modules (groups). The overall module is devised on the basis of the requirements for conveying information between SFC control elements and SFC functional elements described in the document [I-D.ietf-sfc-control-plane].

Module: SFC-control-planes

- Path Maintenance
- Path Optimization
- Load Balancing
- SFC Topology
- Policy
- History
- Fault Handling
- Event

3.1. SFC-Control-Planes Configuration Requirements

This section covers general considerations for managing SFC control plane. The considerations are classified into topology, connectivity, performance, traffic classification, load balancing, and event handling.

First, topology discovery or maintenance is necessary in order to monitor whole or parts of SFP or SFC. Various topologies such as point-to-point and multi-points should be supported. Also, in order to show connectivity topology, the naming mechanism must be considered.

Second, a mechanism of checking connectivity to examine that the services of the given SF execute well or not. This is related to SFP maintenance and optimization.

Also, in order to support path optimization, the performance monitoring is necessary. Basically, the checking of availability and status of services included in SFC needs to be supported. Specifically, statistical data such as packet rates, bandwidth, CPU usage, etc., in accordance with SFP and SFC must be provided for path optimization and load balancing.

Next, traffic classification, reclassification and forwarding must be provided in SF level. This mechanism is performed based on policies.

In order to minimize the effects from faults, the load balancing or load sharing needs to be provided. The load balancing among multiple instances of the same SF. Fault handling, and event handling mechanism should be supported.

Finally, historical data of performance and faults must be stored in order to reliable SFC services.

3.2. SFC-Control-Planes Configuration Model

```
<CODE BEGINS> file "sfc-control-planes@2015-10-15.yang"
module sfc-control-planes {
    namespace "urn:etri:params:xml:ns:yang:sfc-cp";
    prefix sfc-cp;
    include path-maintenance {
        revision-date 2015-10-15;
    }
    include path-optimization {
        revision-date 2015-10-15;
    }
    include load-balancing {
        revision-date 2015-10-15;
    }
    include sfc-topology {
        revision-date 2015-10-15;
    }
    include policy {
        revision-date 2015-10-15;
    }
    include history {
        revision-date 2015-10-15;
    }
    include fault-handling {
        revision-date 2015-10-15;
    }
    include event {
        revision-date 2015-10-15;
    }
    organization "ETRI.";
    contact
    "M-K. Shin mkshin@etri.re.kr
     M-J Choi mjchoi@kangwon.ac.kr";
    description
     "This Module define the management information of
      the control plane as a total of nine modules.";

    revision 2015-10-15{
        reference
         "draft-shin-sfc-control-plane-yang-00 -
          Yang Data Model for Service Function Chaining Control Plane";
    }
}
<CODE ENDS>
```

4. Path Maintenance

This module checks the aliveness of a SFP. This receives SFP name and returns aliveness result which is true or false value.

- SFP name
- Aliveness of SFP: True or False
- RPC: check-path-aliveness (input: SFP name, output: aliveness)

4.1. Path Maintenance Configuration Model

```
<CODE BEGINS> file "path-maintenance@2015-10-15.yang"
submodule path-maintenance {
    belongs-to sfc-control-planes {
        prefix sfc-control-planes;
    }
    organization "ETRI.";
    contact
     "M-K. Shin mkshin@etri.re.kr
      M-J Choi mjchoi@kangwon.ac.kr";

    description
    "This module checks the aliveness of a SFP.
    This receive SFP name and returns aliveness
    result which is true or false value.";
    revision 2015-10-15{
        reference
        "draft-shin-sfc-control-plane-yang-00 -
        Yang Data Model for Service Function Chaining Control Plane";
    }
    container path-maintenance {
        description
        "This container checks the aliveness of a SFP.
        This receive SFP name and returns aliveness result
        which is true or false value.";

        leaf sfp-name {
            type string;
            description
            "The name of service function path.";
        }
        leaf aliveness-of-sfp {
            type boolean;
            description
            "Aliveness flag of the service function.";
        }
```

```
        }
        rpc check-path-aliveness {
            description "Check a path aliveness.";
            input  {
                leaf sfc-name {
                    type string;
                    description
                    "The name of service function path.";
                }
            }
            output {
                leaf Aliveness {
                    type boolean;
                    description
                    "Aliveness flag of the service function.";
                }
            }
        }
    }
    <CODE ENDS>
```

5. Path Optimization

This module constructs and maintains a SFP with a low stretch considering the topological locations and properties (e.g., latency, bandwidth) of SFI

- SFP name
- Optimized SFP
- SFP availability
- Load status
- RPC: Find-optimized-SFP (input: SFP name, output: optimized SFP, SFP availability, Load status)

5.1. Path Optimization Configuration Model

```
<CODE BEGINS> file "path-optimization@2015-10-15.yang"
   submodule path-optimization {
     belongs-to sfc-control-planes {
       prefix sfc-po;
     }
     organization "ETRI.";
     contact
     "M-K. Shin mkshin@etri.re.kr
```

```
         M-J Choi mjchoi@kangwon.ac.kr";
        description
        "This module constructs and maintains a SFP with a low stretch
         considering the topological locations and properties (e.g.,
         latency, bandwidth) of SF.";
        revision 2015-10-15 {
       reference
       "draft-shin-sfc-control-plane-yang-00 -
        Yang Data Model for Service Function Chaining Control Plane";
        }
        container path-optimization {
          description
          "This container constructs and maintains a SFP with
           a low stretch considering the topological locations and
           properties (e.g., latency, bandwidth) of SF.";
          leaf sfp-name {
            type string;
            description
            "The name of service function path to be changed for
             optimization.";
          }
          leaf optimized-sfp {
            type string;
            description
            "The name of optimized service function path.";
          }
          leaf sfp-availability {
            type boolean;
            description
            "The availability of the optimized service function path:
             true or false.";
          }
          leaf load-status {
            type uint8;
            description
            "A percentage value of load status.";
          }
        }
        rpc find-optimized-sfp {
          description
          "Find an optimized service function path.";
          input  {
            leaf sfp-name {
              type string;
              description
              "The name of service function path to be changed for
               optimization.";
            }
```

```
          }
          output {
            leaf optimized-sfp {
              type string;
              description
              "The name of optimized service function path.";
            }
            leaf sfp-availability {
              type string;
              description
              "The availability of the optimized service function path:
               true or false.";
            }
          }
      }
   }
   <CODE ENDS>
```

6. Load Balancing

This module constructs and maintains SFPs to localize the traffic in the network considering load and administrative domain of SFIs.

- Load type: traffic/CPU/memory
- Source SFP name
- Target SFP name
- RPC: perform-load-balance (input: Load type, source SFP name, output: target SFP name)

6.1. Load Balancing Configuration Model

```
<CODE BEGINS> file "load-balancing@2015-10-15.yang"
   submodule load-balancing {
     belongs-to sfc-control-planes{
       prefix sfc-lb;
     }
     organization "ETRI.";
     contact
     "M-K. Shin mkshin@etri.re.kr
      M-J Choi mjchoi@kangwon.ac.kr";
     description
     "This module constructs and maintains SFPs to localize the
      traffic in the network considering load and administrative
      domain of SFIs.";
     revision 2015-10-15 {
```

```
       reference
       "draft-shin-sfc-control-plane-yang-00 -
        Yang Data Model for Service Function Chaining Control Plane";
        }
        container load-balancing {
          description
          "This container constructs and maintains SFPs
           to localize the traffic in the network considering load
           and administrative domain of SFIs.";
          leaf load-type {
            type enumeration {
              enum traffic{
                 description
                 "Traffic";
              }
              enum cpu{
                 description
                 "CPU";
              }
              enum memory{
                 description
                 "Memory";
              }
            }
            description
            "A resource type for load balancing.";
          }
          leaf source-sfp-name {
            type string;
            description
            "The name of service function path with heavy load.";
          }
          leaf target-sfp-name {
            type string;
            description
            "Another name of service function path for load
             distribution.";
          }
        }
        rpc perform-load-balance {
          description
          "Perform a load balancing.";
          input {
            leaf load-type {
              type enumeration {
                enum traffic{
                   description
                   "Traffic";
```

```
                }

                enum cpu{
                   description
                   "CPU";
                }

                enum memory{
                   description
                   "Memory";
                }
              }
              description
              "A resource type for load balancing.";
            }
            leaf source-sfp-name {
              type string;
              description
              "The name of service function path with heavy load.";
            }
          }
          output  {
            leaf target-sfp-name {
              type string;
              description
              "Another name of service function path for load
               distribution.";
            }
        }
    }
}
```

<CODE ENDS>

7. SFC Topology

This module shows the connectivity map of currently working SFCs, SFPs and SFIs.

- Name: SFC name or SFP name or SFI name
- Lists of SF node
- RPC: show-SFC-topology (input: SFC name, output: list of SF nodes)

7.1. SFC Topology Configuration Model

```
<CODE BEGINS> file "sfc-topology@2015-10-15.yang"
   submodule sfc-topology {
     belongs-to sfc-control-planes {
       prefix sfc-st;
     }
     organization "ETRI.";
     contact
     "M-K. Shin mkshin@etri.re.kr
      M-J Choi mjchoi@kangwon.ac.kr";
     description
     "This module shows the connectivity map of currently working
      SFCs, SFPs and SFIs.";
     revision 2015-10-15 {
        reference
        "draft-shin-sfc-control-plane-yang-00 -
        Yang Data Model for Service Function Chaining Control Plane";
     }
     container sfc-topology {
       description
       "This container shows the connectivity map of
        currently working SFCs, SFPs and SFIs. ";
       leaf name {
         type string;
         description
         "The name of service function chain or service function
          path or service function instances.";
       }
       list lists-of-sf-node {
         key name;
         description
         "A list of service functions that compose the service
          chain.";
         leaf name {
           type string;
           description
           "A list of service functions that compose the service
            chain.";

         }
         leaf order {
           type uint8;
           description
           "The order of the service functions.";

         }
       }
```

```
        }
        rpc show-sfc-topology {
          description
          "Display the topology of service function chain.";
          input {
            leaf sfc-name {
              type string;
              description
              "The name of service function chain.";
            }
          }
          output {
            list lists-of-sf-node {
              key name;
              description
              "A list of service functions that compose the service
               chain.";
              leaf name {
                type string;
                description
                "A list of service functions that compose the service
                 chain.";
              }
              leaf order {
                type uint8;
                description
                "The order of the service functions.";
               }
            }
         }
      }
   }
```

<CODE ENDS>

8. Policy

Policies are used to bind an incoming flow to an appropriate SFP.

o Classification Policy Table: One SFC is selected by a classification function according to the defined policy. The classification means service profile matching of traffic flows for identification of appropriate outbound forwarding actions.

  * Flow identifier

  * Matching condition

* Priority

* Mapping SFC name

o Forwarding Policy Table: This policy is used to select SFIs defined in SFC and connect them. A SFF is responsible for delivering traffic received from the network to one or more connected service functions according to the policy table.

* SFF name

* Condition

* SFC name

* SFP name

8.1. Policy Configuration Model

```
<CODE BEGINS> file "policy@2015-10-15.yang"
   submodule policy {
     belongs-to sfc-control-planes {
       prefix sfc-p;
     }
     organization "ETRI.";
     contact
     "M-K. Shin mkshin@etri.re.kr
      M-J Choi mjchoi@kangwon.ac.kr";
     description
     "Policies are used to bind an incoming flow to an appropriate
      SFP.";
     revision 2015-10-15 {
        reference
        "draft-shin-sfc-control-plane-yang-00 -
        Yang Data Model for Service Function Chaining Control Plane";
     }
     container classification-policy-table {
       description
       "One SFC is selected by the classification function according to
       the defined policy. The classification means service profile
       matching of traffic flows for identification of appropriate
       outbound forwarding actions.";
       list classification-policy-table {
         key flow-identifier;
         description
         "To classify the flow, flow identification is necessary.";
         leaf flow-identifier {
           type string;
```

```
            description
            "The flow identifier.";
          }
          leaf matching-condition {
            type string;
            description
            "The flow matching condition.";
          }
          leaf priority {
            type int32;
            description
            "Policy priority. The low value is high priority.";
          }
          leaf mapping-sfc-name {
            type string;
            description
            "The mapped service function name of the flow.";
          }
        }
      }
      container forwarding-policy-table {
        description
        "This policy is used to select SFIs defined in SFC and
         connect them. This policy is used in Service Function
         Forwarder (SFF). A SFF is responsible for delivering traffic
         received from the network to one or more connected service
         functions according to the policy table.";
        list Forwarding-Policy-Table {
          key index;
          description
          "The index of forwarding policy table.";
          leaf index {
            type int32;
            description
            "The index of forwarding policy table.";
          }
          leaf sff-name {
            type string;
                 description "Name of Service Function Forwarder ";
          }
       }
    }
 }

 <CODE ENDS>
```

9. History

This module shows statistical information related performance and faults in accordance with SFCs and SFPs.

- Performance: it monitors and stores time series data related performance
  - Name: SFC name or SFP name
  - Time
  - Aliveness
  - Resource utilization: packet-rate, bandwidth, CPU usage, memory usage, available memory, RIB, FIB, SF-ports bandwidth
- Fault: it monitors and stores time series data related to faults and events.
  - Fault name
  - Fault occurrence time
  - Fault type
  - Fault occurrence location
  - Fault handling action

9.1. History Configuration Model

```
<CODE BEGINS> file "history@2015-10-15.yang"
   submodule history {
     belongs-to sfc-control-planes {
       prefix sfc-h;
     }
     import ietf-yang-types {
       prefix yang;
       revision-date 2013-07-15;
     }
     organization "ETRI.";
     contact
     "M-K. Shin mkshin@etri.re.kr
      M-J Choi mjchoi@kangwon.ac.kr";
     description
     "This module shows statistical information related performance
      and faults in accordance with SFCs and SFPs.";
```

```
       revision 2015-10-15 {
           reference
           "draft-shin-sfc-control-plane-yang-00 -
           Yang Data Model for Service Function Chaining Control Plane";
       }
       container performance {
         description
         "It monitors and stores time series data related performance."
         ;
         list Performance {
           key name;
           description
           "The name of service function.";
           leaf name {
             type string;
             description
             "The name of service function.";
           }
           leaf time {
             type yang:date-and-time;
             description
             "The performance monitoring date.";
           }
           leaf aliveness {
             type boolean;
             description
             "The aliveness flag of the service function.";
           }
           container resource-utilization {
             description
             "To be defined.";
             leaf packet-rate {
               type uint8;
               description
               "Percentage of current package rate utilization.";
             }
             leaf bandwidth {
               type uint8;
               description
               "Percentage of bandwidth utilization.";
             }
             leaf cpu-usage {
               type uint8;
               description
               "Percentage of CPU utilization.";
             }
             leaf memory-usage {
               type uint8;
```

```
            description
            "Percentage of memory utilization.";
          }
          leaf available-memory {
            type uint8;
            description
            "Available memory size of the service function in MB.";
          }
          leaf rib {
            type uint8;
            description
            "Percentage of Routing Information Table utilization.";
          }
          leaf fib {
            type uint8;
            description
            "Percentage of Forwarding Information Table
             utilization.";
          }
          leaf sf-ports-bandwidth {
            type uint8;
            description
            "Percentage of the port's supported bandwidth
             utilization.";
          }
        }
      }
    }
    container fault {
      description
      "It monitors and stores time series data related to faults
       and events." ;
      list fault {
        key fault-name;
        description
        "The name occurred fault.";
        leaf fault-name {
          type string;
          description
          "The name occurred fault.";
        }
        leaf fault-occurrence-time {
        type yang:date-and-time;
          description
          "The time of the fault occurrence.";
        }
        leaf fault-type {
          type string;
```

```
            description
            "The type of occurred fault.";
          }
          leaf fault-occurrence-location {
            type string;
            description
            "The location of fault occurrence.";
          }
          leaf fault-handling-action {
            type string;
            description
            "The fault handing action: drop, bypass, use alternate
             node.";
          }
        }
    }
  }
```

<CODE ENDS>

10. Fault Handling

This module first detects faults and handles the detected faults in accordance with the fault handling action.

- Fault name: Key
- Fault type: Node/Link/Path failures
- Fault handing action: Bypass/Use alternate node/Use alternate chain/Drop traffic
- RPC: handle-fault (input: fault name, output: fault handle result)

10.1. Fault Handling Configuration Model

<CODE BEGINS> file "fault-handling@2015-10-15.yang"

```
   submodule fault-handling {
     belongs-to sfc-control-planes {
       prefix sfc-fh;
     }
     organization "ETRI.";
     contact
     "M-K. Shin mkshin@etri.re.kr
      M-J Choi mjchoi@kangwon.ac.kr";
     description
     "This module first detects faults and handles the detected
      faults in accordance with the fault handling action. ";
```

```
       revision 2015-10-15 {
          reference
          "draft-shin-sfc-control-plane-yang-00 -
          Yang Data Model for Service Function Chaining Control Plane";
       }
       container fault-handling {
          description
           "This Container first detects faults and handles the detected
            faults in accordance with the fault handling action. ";

         list fault-handling {
           key fault-name;
           description
           "The name of occurred fault.";
           leaf fault-name {
             type string;
             description
             "The fault handing action: drop, bypass, use alternate
              node.";
           }
           leaf fault-type {
             type enumeration {
               enum node{
                  description
                  "Node";
               }
               enum link{
                  description
                  "Link";
               }
               enum path-failures{
                  description
                  "Path";
               }
             }
             description
             "The type of occurred fault.";
           }
           leaf fault-handling-action {
             type enumeration {
               enum bypass{
                  description
                  "Bypass";
               }
               enum use-alternate-node{
                  description
                  "Alternate node";
               }
```

```
              enum use-alternate-chain{
                 description
                 "Alternate chain";
              }
              enum drop-traffic{
                 description
                 "Drop the traffic";
              }
            }
            description
            "The fault handing action: drop, bypass, use alternate
             node.";
          }
       }
    }
 }
```

<CODE ENDS>

11. Event

Events such as loops detection, long unavailable forwarding path time, out of service of SFIs are defined.

11.1. Event Configuration Model

```
<CODE BEGINS> file "event@2015-10-15.yang"
   submodule event {
     belongs-to sfc-control-planes {
       prefix sfc-e;
     }
     organization "ETRI.";
     contact
     "M-K. Shin mkshin@etri.re.kr
      M-J Choi mjchoi@kangwon.ac.kr";
     description
     "Events such as loops detection, long unavailable
      forwarding path time, out of service of SFIs.";
     revision 2015-10-15 {
        reference
        "draft-shin-sfc-control-plane-yang-00 -
        Yang Data Model for Service Function Chaining Control Plane";
     }
     notification event {
       description
       "Events such as loops detection, long unavailable forwarding
        path time, out of service of SFIs.";
       leaf event-type {
         type string;
         description
         "Event type.";
       }
       leaf severity {
         type string;
         description
         "The severity of the event.";
       }
       leaf event-explanation {
         type string;
         description
         "The detailed explanation of the event.";
       }
    }
}

<CODE ENDS>
```

12. Security Considerations

TBD.

13. IANA Considerations

TBD.

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

14.2. Informative References

[I-D.ietf-sfc-architecture]
Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", draft-ietf-sfc-architecture-11 (work in progress), July 2015.

[I-D.ietf-sfc-control-plane]
Li, H., Wu, Q., Huang, O., Boucadair, M., Jacquenet, C., Haeffner, W., Lee, S., Parker, R., Dunbar, L., Malis, A., Halpern, J., Reddy, T., and P. Patil, "Service Function Chaining (SFC) Control Plane Components & Requirements", draft-ietf-sfc-control-plane-03 (work in progress), January 2016.

[I-D.lee-sfc-dynamic-instantiation]
Lee, S., Pack, S., Shin, M., and E. Paik, "SFC dynamic instantiation", draft-lee-sfc-dynamic-instantiation-01 (work in progress), October 2014.

[I-D.penno-sfc-yang]
Penno, R., Quinn, P., Zhou, D., and J. Li, "Yang Data Model for Service Function Chaining", draft-penno-sfc-yang-14 (work in progress), January 2016.

Authors' Addresses

Myung-Ki Shin
ETRI
218 Gajeong-ro Yuseung-Gu
Daejeon 305-700
Korea

Phone: +82 42 860 4847
Email: mkshin@etri.re.kr

Mi-Jung Choi
Kangwon National University
1 Kangwondaehak-gil
Chuncheon-si, Gangwon-do 24341
Korea

Phone: +82 33 250 8442
Email: mjchoi@kangwon.ac.kr

Seungik Lee
ETRI
218 Gajeong-ro Yuseung-Gu
Daejeon 305-700
Korea

Phone: +82 42 860 1483
Email: seungiklee@etri.re.kr