ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2015

G. Bernstein
Grotto Networking
W. Roome
M. Scharf
Alcatel-Lucent
Y. Yang
Yale University
July 1, 2014

ALTO Topology Extension
draft-yang-alto-topology-01.txt

Abstract

The Application-Layer Traffic Optimization (ALTO) Service has defined Network and Cost maps to provide basic network information. In this document, we discuss an initial design to provide graph representations of network topology, motivated by a basic use case of multi-flow scheduling. The design is based on the ALTO WG discussions at IETF 89.

Requirements Language

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].

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 January 2, 2015.

Table of Contents

1. Introduction

Topology is a basic information component that a network can provide to network management tools and applications. Example tools and applications that can utilize network topology include traffic engineering, network services (e.g., VPN) provisioning, PCE, application overlays, among others [RFC5693,I-D.amante-i2rs-topology-use-cases, I-D.lee-alto-app-net-info-exchange].

A basic challenge in exposing network topology is that there can be

multiple representations of the topology of the same network infrastructure, and each representation may be better suited for its own set of deployment scenarios. For example, the current ALTO base protocol [I-D.ietf-alto-protocol] is designed for a setting of exposing network topology using the extreme "my-Internet-view"

representation, which abstracts a whole network as a single node that has a set of access ports, with each port connects to a set of endhosts. The base protocol refers to each access port as a PID. This "single-node" abstraction achieves simplicity and provides flexibility. A problem of this abstraction, however, is that the base protocol as currently defined does not provide enough information for use cases such as multiflow scheduling (see Section 2).

An opposite of the single-node representation is the complete raw topology, spanning across multiple layers, to include all details of network states such as endhosts attachment, physical links, physical switch equipment, and logical structures (e.g., LSPs) already built on top of physical infrastructural devices. A problem of the raw topology representation, however, is that its exposure may violate privacy constraints. Also, a large raw topology may be overwhelming and unnecessary for specific applications. As typical ALTO clients are not NMS systems and hence there is no need to access the full RIB.

This document specifies a new type of ALTO Information Resources, which provide graph representations of a network. We call such Information Resources ALTO Topology Maps, or Topology Maps for short. Different from the base single-node abstraction, a Topology Map includes multiple network nodes. Different from the raw topology representation that uses real network nodes, Topology Maps use abstract nodes, although they should be constructed from the real, raw topology, in order to provide grounded information. The design of this document is based on the ALTO WG discussions at IETF 89, with summary slides at http://tools.ietf.org/agenda/89/slides/slides-89-alto-2.pdf.

The organization of this document is organized as follows. We first review the ALTO base protocol in Section 2. Then in Section 3, we give the multi-flow scheduling use case as an example. In Section 4, we specify path vector as a key component to handle multi-flow

scheduling. In Section 5, we give two graph representations complete the design. Section 6 gives a framework of topology transformations to help with the understanding of deriving multiple representations of the topology of the same network infrastructure.

2. Review: the Base Single-Node Representation

We distinguish between endhosts and the network infrastructure of a network. Endhosts are sources and destinations of data that the network infrastructure carries. The network itself is neither the source nor the destination of data.

For the given network, it provides "access ports" or access points where digital signal from endhosts enter and leave the network. One should understand "access ports" in a generic sense. For example, an access port can be a physical Ethernet port connecting to a specific endhost, or it can be a port connecting to a CE which connects to a large number of endhosts. Let AP be the set of access ports (AP) that the network provides.

A high-level abstraction of a network topology is only the set AP, and one can visualize the network as a single, abstract node with the set AP of access ports attached. At each ap in AP, a set of endhosts can be reached as destinations. Let dest(ap) denote the set of endhosts reachable at ap.

There can be multiple ways to partition the set AP. Each partition is called a Network Map. Given a complete partition of AP, the ALTO base protocol introduces PID to represent each partition subset. The ALTO base protocol then conveys the pair-wise connection properties from one PID to another PID through the "single-node". This is the Cost Map.

3. The Multi-flow Scheduling Use Case

There are use cases where simple Cost Metric cannot convey enough information to the applications about pair-wise connection properties from one PID to another PID. Consider an application overlay which needs to schedule the traffic among a set of endhost source-destination pairs, say eh1 -> eh2, and eh3 -> eh4. Simple Cost Metric such as 'available bw' for eh1 -> eh2 and eh3 -> eh4 may not

reflect whether the two paths for eh1 -> eh2 and eh3 -> eh4 share a bottleneck.

More concretely, assume that the network has 7 switches (sw1 to sw7) forming a dumb-bell topology. Switches sw1/sw3 provide access on one side, s2/s4 provide access on the other side, and sw5-sw7 form the backbone. Endhosts eh1 to eh4 are connected to access switches sw1 to sw4 respectively. Assume that the bandwidth of each link is 100 Mbps. Assume that the network is abstracted with 4 PIDs, with each representing the hosts at one access switch.

Now, consider a Cost Map providing end-to-end available bandwidth. There can be two possible interpretations on the semantics of the value of PIDi -> PIDj reported by the Cost Map: (1) it represents reserved bandwidth from PIDi -> PIDj, or (2) it represents possible bandwidth for PIDi -> PIDj, if no other applications use shared resources. The common understanding is (2), just as when we look at the number of available seats on a flight.

Assume that the application receives from the Cost Map that both PID1 -> PID2 and PID3 -> PID4 have bandwidth 100 Mbps. It cannot determine that if it schedules the two flows together, whether it will obtain a total of 100 Mbps or 200 Mbps. This depends on whether the flows share a bottleneck:

- If PID1 -> PID2 and PID3 -> PID4 use different paths, for example, when the first uses sw1 -> sw5 -> sw7 -> sw2, and the second uses sw3 -> sw5 -> sw6 -> sw7 -> sw4. Then the application will obtain 200 Mbps.

- If PID1 -> PID2 and PID3 -> PID4 share the bottleneck, for example, when both use the direct link sw5 -> sw7, then the application will obtain only 100 Mbps.

To distinguish the two possible cases, the network needs to provide more details.

4. Path-Vector as Cost Metric Representation

A key component to address the problem in the preceding section is to introduce path vectors as a Cost Metric, which is a set of path

vectors from a source PID to a destination PID, where each path vector is a sequence (array) of network elements.

A schema for introducing path vectors in Cost Maps is the following extension of Section 11.2.3.6 of [I-D.ietf-alto-protocol]:

```
object {
  cost-map.DstCosts.JSONValue -> JSONString<0,*>;
  meta.cost-mode = "path-vector";
} InfoResourcePVCostMap : InfoResourceCostMap;
```

Specifically, the preceding specifies that InfoResourcePVCostMap extends InfoResourceCostMap. The body specifies that the first extension is achieved by changing the type of JSONValue defined in DstCosts of cost-map to be an array of JSONString; the second extension is that the cost-mode of meta MUST be "path-vector".

An example Cost Map using path-vector is the following:


```
GET /costmap/pv HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,application/alto-error+json
```

```
HTTP/1.1 200 OK
Content-Length: TDB
Content-Type: application/alto-costmap+json

{
  "meta" : {
    "dependent-vtags" : [
      { "resource-id": "my-default-network-map",
```

```
            "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
          },
          {"resource-id": "my-topology-map", // See below
           "tag": "4xee2cb7e8d63d9fab71b9b34cbf76443631554de"
          }
        ],
        "cost-type" : {"cost-mode"  : "path-vector"
        }
      },

      "cost-map" : {
        "PID1": { "PID1":[],
                  "PID2":["ne56", "ne67"],
                  "PID3":[],
                  "PID4":["ne57"]
        },
        "PID2": { "PID1":["ne75"],
                  "PID2":[],
                  "PID3":["ne75"],
                  "PID4":[]
        },
        "PID3": { "PID1":[],
                  "PID2":["ne57"],
                  "PID3":[],
                  "PID4":["ne57"]
        },
        "PID4": { "PID1":["ne75"],
                  "PID2":[],
                  "PID3":["ne75"],
                  "PID4":[]}
      }
    }
```

The example illustrates that there are two key extensions to the ALTO base protocol:

- It introduces a new "cost-mode" named path vector.

- To indicate the resource that provides information on the elements

of path vectors (e.g., ["ne5", "ne67"] for the path vector from PID1 to PID2, it introduces a new dependency. In the example, we indicate that it is a resource named "my-topology-map".

5. Topology using Opaque Network Elements

A missing piece to complete the preceding design is how to represent the information resource that provides information on the elements of the path vectors. One simple approach is to introduce simple network element property maps, which provide a mapping from a network element to its properties such as bandwidth or shared risk link group (srlg).

A schema for the Network Element Property Map can be:

```
object-map {
  JSONString -> NetworkElementProperties; // name to properties
} NetworkElementMapData;

object-map {
  JSONString bw;
  JSONString srlg<0,*>;
} NetworkElementProperties;
```

An example Network Element Property Map:

```
GET /nepmap HTTP/1.1
Host: alto.example.com
Accept: application/alto-nepmap+json,application/alto-error+json
```

```
HTTP/1.1 200 OK
Content-Length: TBD
Content-Type: application/alto-nepmap+json

{
  "meta" : {
    "vtag": {
      "resource-id": "my-topology-map",
      "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
    }
  },
  "nep-map" : {
    "ne57" : {"bw" : 100, "srlg" : [1, 3]}, // link sw5->sw7
    "ne75" : {"bw" : 100, "srlg" : [1, 3]}, // link sw7->sw5
    "ne56" : {"bw" : 100, "srlg" : [1]},    // link sw5->sw6
    "ne65" : {"bw" : 100, "srlg" : [1]},    // link sw6->sw5
    "ne67" : {"bw" : 100, "srlg" : [3]},    // link sw6->sw7
    "ne76" : {"bw" : 100, "srlg" : [3]},    // link sw7->sw6
  }
}
```

An advantage of the representation is that it does not need to distinguish network nodes vs network links.

6. Topology using a Graph Representation

A potential problem of the path vector representation is its lacking of compactness. For example, suppose a network has N PIDs, then it will need to represent N * (N-1) paths, if each source-destination pair has one path computed using a shortest-path algorithm. On the other hand, the underlying graph may have only O(F * N) elements, where F is the average degree of the topology.

A graph representation of the example topology in the preceding section has three components:

1. PIDs: {AP1, AP2, AP3}, which is already defined by base ALTO;

2. Nodes: {s1, s2, s3}; and

3. Links: {AP1 -- s1, AP2 -- s2, AP3 -- s3, s1 -- s2, s2 -- s3, s1 -- s3}.

A schema for the graph representation, based on the types already defined in the base ALTO protocol, is the following:

```
object {
  TopologyMapData topology-map;
} InfoResourceTopologyMap : ResponseEntityBase;

object {
  NodeMapData nodes;
  LinkMapData links;
} TopologyMapData;

object-map {
  JSONString -> NodeProperties; // node name to properties
} NodeMapData;

object {
  JSONString type;
  ...
} NodeProperties;


object-map {
  JSONString -> LinkProperties; // link name to properties
} LinkMapData;

object {
  JSONString src;
  JSONString dst;
  JSONString type;
  CostValue costs<0,*>;
} LinkProperties;

object {
  CostMetric metric;
  JSONValue value; // value type depends on metric type
} CostValue;
```

One complexity of the graph representation is multicast/broadcast links. Assume that the link from sw5 -> sw7 is actually a wireless

link and the application may benefit in knowing that sw5 -> sw7 and sw5 -> sw6 can be active simultaneously. In other words, sw5 -> [sw6, sw7] is a broadcast link. Knowing such links can be beneficial in settings such as wireless opportunistic routing.

An example using the schema:


```
GET /topologymap HTTP/1.1
Host: alto.example.com
Accept: application/alto-topologymap+json,application/alto-error+json
```

```
HTTP/1.1 200 OK
Content-Length: TBD
Content-Type: application/alto-topologymap+json

{
  "meta" : {
    "vtag": {
      "resource-id": "my-topology-map",
      "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
    }
  },
  "topology-map" : {
    "nodes" : {
      "sw1" : {"type" : "switch"},
      "sw2" : {"type" : "switch"},
      "sw3" : {"type" : "switch"},
      "sw4" : {"type" : "switch"},
      "sw5" : {"type" : "switch"},
      "sw6" : {"type" : "switch"},
      "sw7" : {"type" : "switch"}
    },
    "links" : {
      "e1" : {"src" : PID1,
              "dst" : "sw1",
```

```
                "type": "bidirected";
                "costs" : [
                  {"cost-metric" : "availbw", "value" : 100
                  },
                  {"cost-metric" : "srlg", value : [1, 3]}

                ]
        },
        "e2" : {"src" : PID2,
                "dst" : "sw2",
                ...
        },
        "e3" : {"src" : PID3,
                "dst" : "sw3",
                ...
        },
```



```
        "e4" : {"src" : PID4,
                "dst" : "sw4",
                ...
        },
        "e15" : {"src" : "sw1",
                 "dst" : "sw5",
                 ...
        },
        "e35" : {"src" : "sw3",
                 "dst" : "sw5",
                 ...
        },
        "e27" : {"src" : "sw2",
                 "dst" : "sw7",
                 ...
        },
        "e47" : {"src" : "sw4",
                 "dst" : "sw7",
                 ...
        },
        "e57" : {"src" : "sw5",
                 "dst" : "sw7",
                 ...
        },
        "e56" : {"src" : "sw5",
```

```
                    "dst" : "sw6",
                    ...
            },
            "e67" : {"src" : "sw6",
                    "dst" : "sw7",
                    ...
            }
          }
        }
      }
```

Note that the preceding Graph Representation does not provide path informatino. Hence, it should be used with the path-vector representation, or in constraint-light settings where networks use simple, public algorithms to compute routing and hence no need to provide the path vectors explicitly.

7. Graph Transformations to Build Topology/Overlays

The preceding sections give a top-down derivation. In this section we give a graph transformation framework to build the schema from a

raw topology G(0). The network conducts transformations on G(0) to obtain other topologies, with the following objectives:

1. Simplification: G(0) may have too many details that are unnecessary for the receiving app (assume intradomain); and

2. Preservation of privacy: there are details that the receiving app should not be allowed to see; and

3. Conveying of logical structure (e.g., MPLS paths already computed); and

4. Conveying of capability constraints (the network can have limitations, e.g., it uses only shortest path routing); and

5. Allow modular composition: path from one point to another point is delegated to another app.

The transformation of G(0) is to achieve/encode the preceding. For conceptual clarity, we assume that the network uses a given set of operators. Hence, given a sequence of operations and starting from G(0), the network builds G(1), to G(2), ...

Below is a list of basic operators that the network may use to transform from G(n-1) to G(n):

- O1: Deletion of a switch/port/link from G(n-1);
- O2: Switch aggregation: a set Vs of switches are merged as one new (logical) switch, links/ports connected to switches in Vs are now connected to the new logical switch, and then all switches in Vs are deleted;
- O3: Path representation: For a given extra path from A to R1 to R2 ... to B in G(n-1), a new (logical) link A -> B is added; if the constraint is that A -> must use the path, it will be put into the Overlay;
- O4: Switch split: A switch s in G(n-1) becomes two (logical) switches s1 and s2. The links connected to s1 is a subset of the original links connected to s; so is s2.

8. Operations on Exported Topology

Going beyond the basic topology exposure from the network and applications/tools, we anticipate that applications and tools can derive results and feed to topology. In particular, we consider the following operations:

- Instantiation of app guidance in real network: The details of instantiation will be outside the scope of this document. Example protocols include PCEP Extensions for Stateful PCE [I-D.ietf-pce-stateful-pce], RSVP LSP's and their associated characteristics, (i.e.: head and tail-end LSR's, bandwidth, priority, preemption, etc.). The reason that we choose the preceding operator set is that they are "implementable".
- We also anticipate topology guided mapping of other data: to allow applications to subscribe to statistics and link status from the derived topology.

9. Security Considerations

This document has not conducted its security analysis.

10. IANA Considerations

This document does not specified its IANA considerations, yet.

11. Acknowledgments

The author thanks discussions with Erran Li, Tianyuan Liu, Andreas Voellmy, Haibin Song, and Yan Luo.

12. References

12.1. Normative References

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

12.2. Informative References

[I-D.amante-i2rs-topology-use-cases]
Amante, S., Medved, J., Previdi, S., and T. Nadeau, "Topology API Use Cases", draft-amante-i2rs-topology-use-cases-00 (work in progress), February 2013.

[I-D.ietf-alto-protocol]
Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol", draft-ietf-alto-protocol-17 (work in progress), July 2013.

[I-D.lee-alto-app-net-info-exchange]
Lee, Y., Bernstein, G., Choi, T., and D. Dhody, "ALTO Extensions to Support Application and Network Resource Information Exchange for High Bandwidth Applications",

draft-lee-alto-app-net-info-exchange-02 (work in progress), July 2013.

[RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.

Authors' Addresses

Greg Bernstein
Grotto Networking
Fremont, CA
USA

Email: gregb@grotto-networking.com

Wendy Roome
Alcatel-Lucent Technologies/Bell Labs
600 Mountain Ave, Rm 3B-324
Murray Hill, NJ 07974
USA

Phone: +1-908-582-7974
Email: w.roome@alcatel-lucent.com

Michael Scharf
Alcatel-Lucent Technologies
Germany

Email: michael.scharf@alcatel-lucent.com

Y. Richard Yang
Yale University
51 Prospect St
New Haven CT
USA

Email: yry@cs.yale.edu