ALTO Topology Extensions: Node-Link Graphs
draft-yang-alto-topology-06.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 designs to provide abstracted (node-link) graph
   representations of network topology.

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 September 10, 2015.

Copyright Notice

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 [RFC7285] 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 called endpoints.
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 sufficient information for use
cases such as the multi-flow scheduling use case (see [draft-yang-
alto-path-vector]).

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 the 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. Since the target of ALTO
is general applications which do not want or need to understand
detailed routing protocols or raw topology collected in routing
information bases (RIB), raw topology does not appear to be a good
fit for ALTO.

A main objective of this document is to specify a new type of ALTO
Information Resources, which provide abstracted graph (node-link)
representations of a network to provide only enough information for
applications. 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, a topology map may use abstract nodes, although they will 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. In Section 3, we give a
node-link representation.

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 a given network, it provides "access ports" (interfaces, or
access points) where data signal from endhosts enter and leave the

network infrastructure. 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, as Figure 1, the network as a single, abstract
node with the set AP of access ports attached. At each ap in AP, a
set of endhosts are attached to send or receive information from the
network. Let attach(ap) denote the set of endhosts attached to ap.

```
                 +---------------------+
    ap_1      |                     |
      +------+                     +------+
             |                     |
             |                     |
      +------+                     +------+
             |                     |

             |                     |
      +------+                     +------+
             |                     |
             |                     |
      +------+                     +------+
             |                     |    ap_n
                 +---------------------+
```
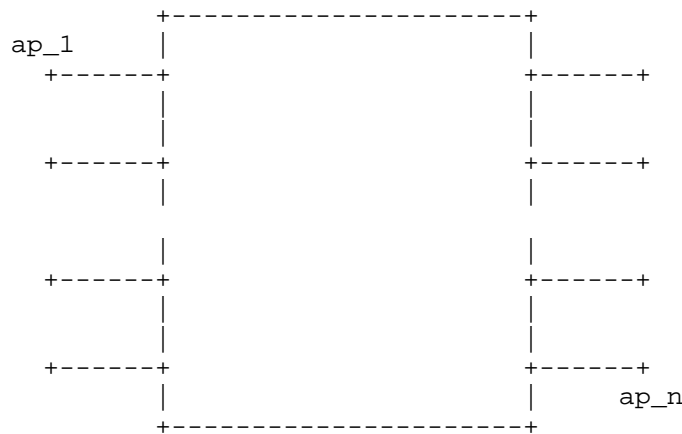
Figure 1: Base Single-Node Topology Abstraction.

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
between one PID and another PID through the "single-node". This is
the cost map.

3. Topology using a Graph (Node-Link) Representation

3.1. Use Cases

[draft-yang-alto-path-vector] proposes path vectors to extend the
preceding topology to expose network elements. A potential problem
of the path vector representation, however, 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, and hence can be a much
smaller value than N. For such settings, in particular, when privacy
protection is not an issue (e.g., in the same-trust domain setting),
a node-link representation can be more compact.

Another setting where a node-link graph approach is beneficial is
application guided path selection. With a topology graph, an
application can compute maximum flows to discover the desired paths
and signal (out the scope of this document) to the network to set up
the paths. The computation can be done by the application itself, or
through a third entity such as a PCE server. The recent development
of SDN makes this use case more possible. A requirement of realizing
this use case is that the path computed by the application is
realizable, in particular, when the topology is an abstract topology.
By realizable, we mean that a path computed on the abstract topology
can be converted to configurations on network devices to achieve the
properties in the abstract topology.

3.2. A Node-Link Schema

A schema for the graph (node-link) 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;
```

In particular, the schema distinguishes two types of links: edge-
attach, and core, where the former is for a link between a network
element and a group of endhosts (PID), and the later is between two
network elements.

An example using the schema is:

```
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" : {
      "dependent-vtags" : [
         { "resource-id": "my-default-network-map",
           "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
         }
      ],
      "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": "edge-attach",
                 "costs" : [
                    {"cost-metric" : "availbw", "value" : 100
                    },
                    {"cost-metric" : "srlg", value : [1, 3]}

                 ]
         },
         "e2" : {"src" : "PID2",
                 "dst" : "sw2",
                 "type": "edge-attach",
```

```
                  ...
          },
          "e3" : {"src" : "PID3",
                  "dst" : "sw3",
                  ...
          },
          "e4" : {"src" : "PID4",
                  "dst" : "sw4",
                  "type": "edge-attach",
                  ...
          },
          "e15" : {"src" : "sw1",
                   "dst" : "sw5",
                   "type": "core",
                   ...
          },
          "e35" : {"src" : "sw3",
                   "dst" : "sw5",
                   "type": "core",
                   ...
          },
          "e27" : {"src" : "sw2",
                   "dst" : "sw7",
                   "type": "core",
                   ...
          },
          "e47" : {"src" : "sw4",
                   "dst" : "sw7",
                   "type": "core",
                   ...
          },
          "e57" : {"src" : "sw5",
                   "dst" : "sw7",
                   "type": "core",
                   ...
          },
          "e56" : {"src" : "sw5",
                   "dst" : "sw6",
                   "type": "core",
                   ...
          },
          "e67" : {"src" : "sw6",
                   "dst" : "sw7",
                   "type": "core",
                   ...
          }
        }
      }
```

```
   }
```

3.3. Discussions

   The node-link schema specified in the preceding section is still a
   standard graph representation of a network (graph). An alternative
   design, which may provide substantial benefit, is using a property
   graph design. In particular, in a property graph based design, it is
   unnecessary that a node in the property graph represents a network
   node, a link in the property graph represents a network link.
   Instead, network nodes, network links and network paths can all be
   represented as nodes in a property graph, and links represent their
   relationship. This design can be flexible in modeling settings such
   as topology abstraction (e.g., to denote, in the same graph, that a
   network link is composed of a path, through a aggregation label).
   Property-graph frameworks such as Gremlin can provide powerful and
   compact querying languages for application's usage.

   Using either the standard node-link graph in the preceding section or
   the property graph abstraction, one may not use a rigid hierarchical
   design. Consider a model that uses a strict hierarchy, and a higher
   layer node can specify a set of nodes in the lower layer as
   supporting nodes; a higher layer link can specify a set of links in
   the lower layer as supporting links [draft-clemm-i2rs-yang-network-
   topo-01]. To test the problem of that model, consider a simple
   topology. Assume that the network consists of 3 data centers (dc1,
   dc2, and dc3). dc1 has two routers dc11 and dc12; dc2 has dc21 and
   dc22; and dc3 has dc31 and dc32. The connections are that (1) two
   routers in the same data center are connected; (2) dc11, dc21 and
   dc31 are mutually connected; same for dc12, dc22, and dc32.

   The network can provide different abstract topologies: for tenants in
   dc1, they see dc11, dc12, and dc2, dc3; same for tenants in dc2, and
   dc3. In other words, each tenant in a DC sees the detailed topology
   of its DC and the other data centers are abstracted to be single
   nodes.

   This case turns out to be not doable for their pure hierarchical
   layer approach, where a top layer node/link has supporting nodes/
   links. Specifically, thee model cannot have cross-layer links such
   as dc11 -> dc2.

4. Security Considerations

   This document has not conducted its security analysis.

5. IANA Considerations

   This document does not specified its IANA considerations, yet.

6. Acknowledgments

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

7. References

7.1. Normative References

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

7.2. Informative References

   [I-D.amante-i2rs-topology-use-cases]
              Medved, J., Previdi, S., Lopez, V., and S. Amante,
              "Topology API Use Cases", draft-amante-i2rs-topology-use-
              cases-01 (work in progress), October 2013.

   [I-D.clemm-i2rs-yang-network-topo]
              Clemm, A., Medved, J., Tkacik, T., Varga, R., Bahadur, N.,
              and H. Ananthakrishnan, "A YANG Data Model for Network
              Topologies", draft-clemm-i2rs-yang-network-topo-01 (work
              in progress), October 2014.

   [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.

   [RFC7285] Alimi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S.,
              Roome, W., Shalunov, S., and R. Woundy, "Application-Layer
              Traffic Optimization (ALTO) Protocol", RFC 7285, September
              2014.

Appendix A. Graph Transformations and Operations to Build Topology
             Representation for Applications

   In this appendix, 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):

   o O1: Deletion of a switch/port/link from G(n-1);

   o 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;

   o 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;

   o 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.

Authors' Addresses

   Greg Bernstein
   Grotto Networking
   Fremont, CA
   USA

   Email: gregb@grotto-networking.com

   Young Lee
   Huawei
   TX
   USA

   Email: leeyoung@huawei.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