ALTO WG G. Bernstein
Internet-Draft Grotto Networking
Intended status: Standards Track Y. Lee
Expires: September 9, 2015 Huawei
W. Roome
M. Scharf
Alcatel-Lucent
Y. Yang
Yale University
March 8, 2015

ALTO Topology Extension: Path Vector as a Cost Mode
draft-yang-alto-path-vector-00.txt

Abstract

The Application-Layer Traffic Optimization (ALTO) Service has defined network and cost maps to provide basic network information, where the cost maps allow only scalar (numerical or ordinal) cost mode values. This document introduces a new cost mode called path vector to allow ALTO clients to better distinguish cost information. This document starts with a non-normative use case called multi-flow scheduling to illustrate that ALTO cost maps without path vectors cannot provide sufficient information. This document then defines path vector as a new cost mode.

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].

This Internet-Draft will expire on September 9, 2015.

Copyright Notice

Table of Contents

1. Introduction

The 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. The base protocol refers to each access port as a PID. This "single-node" abstraction is simple and can support a wide range of applications already.

A problem of this abstraction, however, is that it does not provide sufficient information for use cases such as multi-flow scheduling (see Section 3), which essentially require exposure of topology information beyond the single-node abstraction, to detect sharing of resources in the underlying topology.

This document goes beyond single-node topology by introducing path vector as a new ALTO cost mode, where each path vector specifies the network elements on the routing path from a set of source endhosts to a set of destination endhosts. Since the network elements on a path vector are abstract network elements defined by ALTO servers, the new path-vector cost mode provides a mechanism to allow a network to control the level of topology exposure, and at the same time better support application traffic optimization. The design of path vector 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. Section 2 gives a non-normative use case called multi-flow scheduling to illustrate the need to introduce path vectors. Section 3 formally specifies the path vector cost mode. Sections 4 and 5 discuss security and IANA considerations.

2. The Multi-flow Scheduling Use Case

ALTO uses a simple single-node network abstraction. Specifically, each network map in ALTO defines an abstract, single node network, where endhosts are partitioned to a set of access ports, with each access port called a PID. For a given network map, a cost map of a given cost metric provides a scalar (numerical or ranking) cost value for each pair of source and destination PIDs.

Although simple, the single-node, simple scalar cost maps may not convey enough information to the applications about pair-wise connection properties between one PID and another PID. See [I-D.bernstein-alto-topo] for a survey of use-cases where extended network topology information is needed.

This document uses a simple use case to illustrate the issue. Consider a network as shown in Figure 1. 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.

```
                               +------+
                               |      |
                             --+ sw6  +--
                           /   |      |  \
     PID1 +-----+         /    +------+   \          +-----+  PID2
     eh1__|     |_       /                 \     ____|     |__eh2
          | sw1 | \   +--+---+         +---+--+ /    | sw2 |
          +-----+  \  |      |         |      |/     +-----+
                    \_| sw5  +---------+ sw7  |
     PID3 +-----+   / |      |         |      |\     +-----+  PID4
     eh3__|     |__/  +------+         +------+ \____|     |__eh4
          | sw3 |                                    | sw4 |
          +-----+                                    +-----+
```

Figure 1: Raw Network Topology.

The single-node ALTO topology abstraction of the network is shown in Figure 2.

```
                    +----------------------+
           {eh1}    |                      |     {eh2}
           PID1     |                      |     PID2
             +------+                      +------+
                    |                      |
                    |                      |
           {eh3}    |                      |     {eh4}
           PID3     |                      |     PID4
             +------+                      +------+
                    |                      |
                    +----------------------+
```

Figure 2: Base Single-Node Topology Abstraction.

Consider an application overlay (e.g., a large data analysis system) which needs to schedule the traffic among a set of endhost source-destination pairs, say eh1 -> eh2, and eh3 -> eh4. The application can request a cost map providing end-to-end available bandwidth, using 'available bw' as cost-metric and 'numerical' as cost-mode, where the 'available bw' between two PIDs represents possible bandwidth for PIDi -> PIDj, if no other applications use shared resources.

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 routing of the two flows shares a bottleneck in the underlying topology:

o Case 1: 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.

o Case 2: 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 allow applications to distinguish the two possible cases, the network needs to provide more details. This document introduces path vector to resolve the issue.

3. Path-Vector as a new Cost Mode

An extension supporting the path-vector cost-mode MUST support the following following extension of Section 11.2.3.6 of [RFC7285]:

```
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-metric": "routingcost",
                   "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":[]}
  }
}
```


To interpret the path-vector in a cost map providing path vectors, the client will need access to the properties of the network elements named in the path vectors. Such properties should be provided from an element property service. Hence, the "dependent-tags" of a cost map supporting path vectors MUST include two dependent resources: one for a network map, and the other for an element property service. This document does not define the property service. The appendix gives one definition, but it can be a different one.

4. Security Considerations

This document has not conducted its security analysis.

5. IANA Considerations

This document requires the definition of a new cost-mode named path-vector.

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. Network Element Properties Map

A missing piece to complete the path-vector design to resolve the ambiguity in the use case is how to provide information on the elements of the path vectors. A minimal approach is to introduce network element properties (NEP) maps, where each NEP map provides a mapping from a network element to its properties such as bandwidth or shared risk link group (srlg).

A schema of an NEP map is:

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

object-map {
  JSONString bw;
  JSONString srlg<0,*>;
  [JSONString type;] // should be from an enumeration only
} 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
  }
}
```

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