ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: January 14, 2021

Q. Xiang
Yale University
J. Zhang
Tongji/Yale University
F. Le
IBM
Y. Yang
Yale University

July 13, 2020

ALTO Extension: Unified Resource Representation
draft-xiang-alto-unified-representation-03.txt

Abstract

The ALTO protocol [RFC7285] provides network information to applications so that applications can make network informed decisions to improve the performance. However, the base ALTO protocol only provides coarse-grained end-to-end metrics, which are insufficient to satisfy the demands of applications to solve more complex network optimization problems. The ALTO Path Vector extension [DRAFT-PV] has been introduced to allow ALTO clients to query information such as capacity regions for a given set of flows. However, the current design of this extension has a limited expressiveness. The goal of this document is to introduce a unified resource representation service as an extension of ALTO (ALTO-UR), which allows the ALTO clients to query and get the capacity regions of more complex resource information, such as Shared-Risk-Link-Group (SRLG), multi-path routing, multicast and on-demand routing, for a given set of flows.

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 https://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 14, 2021.

Table of Contents

1. Introduction

As discussed in [DRAFT-PV], the "one-big-switch" abstraction used in the ALTO base protocol lacks the ability to support emerging use cases, such as inter-datacenter data transfers, because this abstraction cannot reveal the resource sharing, i.e., the capacity region, for a set of flows. The ALTO Path Vector extension addresses this insufficiency by using the path vector abstraction to express the capacity region in a set of linear inequalities. However, in an internal discussion with the leading persons of several important ALTO use cases, it is revealed that the expressiveness of the ALTO Path Vector extension is limited in three aspects:

- It cannot provide compact encoding of the SRLG for a set of flows;
- It assumes that each flow in the client's query will use a single-path route, and hence cannot encode the resource sharing for flows that are forwarded along multi-path routes or multicast flows;
- It assumes that the route of each flow in the client's query is pre-computed, and hence cannot encode the resource sharing for flows that use on-demand routing, e.g., the path computation element (PCE) protocol.

To cope with these issues, this document introduces a new ALTO extension, the unified representation (ALTO-UR). This extension expands the linear inequality encoding of capacity regions used in ALTO-PV, to a generic, complete encoding, which uses mathematical programming constraints to represent the capacity regions for a set of flows.

2. 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 [RFC2119].

3. Changes Since Version -02

- Add a discussion of an ongoing investigation of a language-based resource discovery framework utilizing the unified resource representation in Section 8.

4. Limitations of the ALTO Path Vector Extension

The limitations of the ALTO-PV extension are illustrated with the same dumbbell topology used in [DRAFT-PV]. Assume that the bandwidth of every link is 100 Mbps, and that the SRLG of each link is shown in

Figure 1. Consider an application overlay (e.g., a large data analytics system) which wants to schedule the traffic among a set of end host source-destination pairs, say eh1 -> eh2 and eh1 -> eh4.

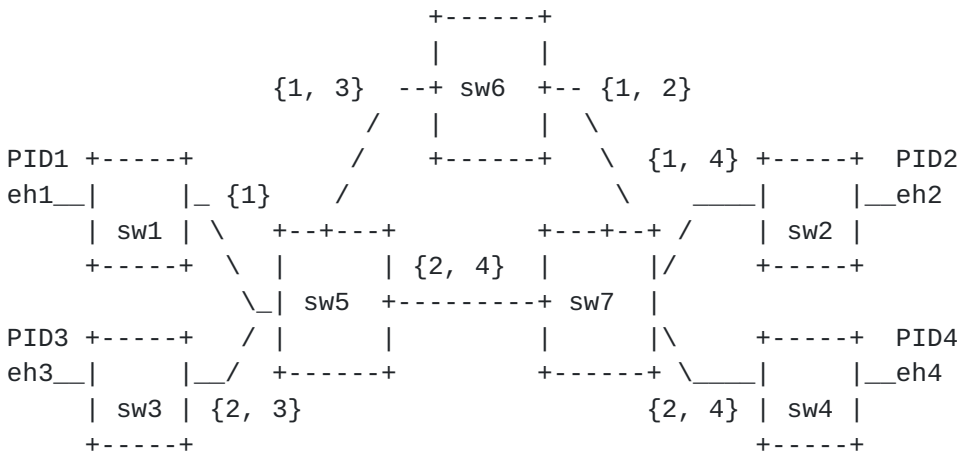

Figure 1: A Dumbbell Network Topology

- Assume the application is only interested in the SRLG of both flows, not the bandwidth. The route of eh1 -> eh2 is eh1 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh2, and the route of eh3 -> eh4 is eh3 -> sw1 -> sw5 -> sw7 -> sw2 -> eh4. The minimal yet accurate information returned to the application should be {2, 3, 4}, the SRLG of both flows, since flow 1 has a set of SRLG {1, 2, 3, 4} and flow 2 has a set of SRLG {2, 3, 4}. In contrast, in the current ALTO-PV service, the ALTO server needs to return the ane-path of each flow and the SRLG of every ane, where ane and ane-path are defined in [DRAFT-PV]. This response is redundant and causes unnecessary information exposure to the application, e.g., the information of flow has an SRLG 1 should not be returned to the application.

- Assume the application is only interested in the bandwidth of both flows. The route of eh1 -> eh2 is eh1 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh2, and the route of eh3 -> eh4 is a multi-path route, i.e., {eh3 -> sw1 -> sw5 -> sw7 -> sw2 -> eh4, eh3 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh4}, where each path would forward 50 percent of the traffic for eh3 -> eh4. The ALTO-PV service cannot reveal the traffic split of the multi-path route for eh3 -> eh4, or the bandwidth sharing for both flows on link sw5 -> sw6.

- Assume the network has a PCE sever, through which the application can reserve bandwidth for both flows. Before the application makes the reservation request, the application queries the ALTO server to get the bandwidth capacity region of both flows, which it wants to use to decide how much bandwidth to reserve for each flow. Suppose when the ALTO server receives a query, the network only has two precomputed routes for both flows: eh1 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh2, and eh3 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh4. Through the ALTO-PV service, the application receives the information that the total bandwidth it can reserve for both flows cannot exceed 100 Mbps. However, one important feature of the PCE server is that it can compute the route for reservation request on-demand, and hence it can find routes with a larger bandwidth. In this example, if the application submits a request to reserve 100 Mbps bandwidth for each flow, the PCE server can compute two on-demand routes, i.e., eh1 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh2 and eh3 -> sw1 -> sw5 -> sw7 -> sw2 -> eh4, and still return a success signal to the application. This shows that the ALTO-PV service cannot encode the capacity region for flows who use on-demand routing.

5. Overview of the Unified Representation Extension

Although different patches and extensions can be introduced to address the aforementioned insufficiencies of the ALTO-PV service, it is desirable to design a service that provides a generic solution that can encode different types of resource sharing for a set of flows. To this end, this document introduces the ALTO Unified Representation (ALTO-UR) service.

5.1. Basic idea

The basic idea of the ALTO-UR service is to use mathematical programming constraints to represent the capacity region for a set of flows. Different from linear inequalities used in the ALTO-PV service, mathematical programming constraints can represent a much wider range of resource information. To illustrate the expressiveness of mathematical programming constraints, we revisit the examples in Figure 1.

Assume the route of eh1 -> eh2 is eh1 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh2, and the route of eh3 -> eh4 is eh3 -> sw1 -> sw5 -> sw7 -> sw2 -> eh4. Denote the SRLG of flow eh1 -> eh2 as f1:SRLG, and that of flow eh3->eh4 as f2:SRLG. Then the SRLG of both flows can be represented as

f1:SRLG intersect f2:SRLG = {2, 3, 4}

Assume the route of eh1 -> eh2 is eh1 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh2, and the route of eh3 -> eh4 is a multi-path route, i.e., {eh3 -> sw1 -> sw5 -> sw7 -> sw2 -> eh4, eh3 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh4}, where each path would forward 50 percent of the traffic for eh3 -> eh4. Denote the available bandwidth of eh1 -> eh2 as f1-bw and those of eh3 -> eh4 along two paths as f2-bw-p1 and f2-bw-p2. The bandwidth sharing of two flows can be represented as:

```
f1:bw <= 100 Mbps, for (sw1, sw5), (sw7, sw2)
f2:bw:p1 + f2:bw:p2 <= 100 Mbps, for (sw3, sw5), (sw7, sw4)
f1:bw  + f2:bw:p1 <= 100 Mbps, for (sw5, sw6), (sw6, sw7)
f2:bw:p2 <= 100 Mbps, for (sw5, sw7)
f2:bw:p1 = f2:bw:p2
```

Assume the routes for both flows are computed on demand and each flow can only use a single path. For eh1 -> eh2, use f1-bw-p1 and f1-bw-p2 to represent the available bandwidth of routes eh1 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh2 and eh1 -> sw1 -> sw5 -> sw7 -> sw2 -> eh2, respectively. For eh3 -> eh4, use f2-bw-p1 and f2-bw-p2 to represent the available bandwidth of routes eh3 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh4 and eh3 -> sw1 -> sw5 -> sw7 -> sw2 -> eh4, respectively. The bandwidth capacity region of both flows can be represented as:

```
f1:bw:p1 + f1:bw:p2 <= 100 Mbps, for (sw1, sw5), (sw7, sw2)
f2:bw:p1 + f2:bw:p2 <= 100 Mbps, for (sw3, sw5), (sw7, sw4)
f1:bw:p1 + f2:bw:p1 <= 100 Mbps, for (sw5, sw6), (sw6, sw7)
f1:bw:p2 + f2:bw:p2 <= 100 Mbps, for (sw5, sw7)
f1:bw:p1 = 0 or f1:bw:p2 = 0
f2:bw:p1 = 0 or f2:bw:p2 = 0
```

The next few subsections present the approaches adopted by the ALTO unified representation extension.

5.2. New Cost Type to Encode Mathematical Programming Variables

This document introduces the unified representation cost type, with the following cost mode and cost metric.

5.2.1. Cost Mode: array

The cost mode of the notation cost type is "array", which is defined in [DRAFT-PV]. The values are arrays of JSONValue. The specific type of each element in the array depends on the cost metric.

5.2.2. Cost Metric: variable-list

This document specifies a new cost metric called "variable-list". This cost metric indicates that the cost value is a list of variables that will be used in mathematical programming constraints.

5.3. New Entity Domain to Provide Mathematical Programming Constraints

This document adopts the property map defined in [DRAFT-UP] to encode the properties of abstract network elements. A new domain "cstr" (short for constraint) is registered in the property map. Each entity in the "cstr" domain has an identifier of an CSTR. Each CSTR has one property, which represents the semantics of this constraint, e.g., a "bw-cstr" property indicates that this constraint represents the bandwidth sharing among flows. This property is provided in information resources called "Property Map Resource" and "Filtered Property Map Resource". The "Filtered Property Map" resource which supports the "cstr" domain is used to encode the properties of cstr entities, and it is called a cstr Property Map in this document.

5.4. Multipart Response to Provide the Unified Representation

To ensure the consistency between the unified representation cost map and the corresponding CSTR property map, this document adopts the design of [DRAFT-PV] to allow a response to contain both the unified representation in a filtered cost map and the associated CSTR property map.

5.5. Designing New Interfaces for More Flexible Queries

Current ALTO protocol and its extensions allow applications to query resource information by specifying IP addresses of endpoints and simple filters. However, with the emerging of new networking architecture (e.g., software defined networking and network function virtualization) and the fine-grained resource requirement of applications (e.g., link-disjoint paths and endpoint precedence), applications need a more flexible interface to specify queries of resource information.

[DRAFT-FCS] proposes a flexible flow query extension service to allow applications to specify query entities based on flexible matching conditions (e.g., TCP/IP 5-tuple) instead of IP addresses only. However, it still does not allow an application to specify more fine-grained resource requirements. For example, a multi-domain service function chaining application may want to get the endpoint cost information of a chain of endpoints in the order of firewall, load balancer and data analyzer. The endpoint cost information of a chain

of endpoints not in this order is of no interest to the application and should not be returned to the application.

As such, this document makes an initial proposal to design new interfaces for application to express fine-grained requirements of resource in the query. In addition to allowing the ALTO client to specify endpoints using flexible matching conditions (e.g., TCP/IP 5 tuple), one key idea in this proposal is to use a resource filter design

Specifically, two types of resource properties are defined. The first type is value property, which are typical quality of services metrics for a given flow. Examples of the value property include the bandwidth, delay, loss rate and so on. The second type is set property. Examples of such a property include the forwarding and processing devices used by a flow (denoted as nodes), the physical links used by a flow (denoted as links) and the shared risk link group (SRLG) of all the devices and links used by a flow.

With these two types of resource properties, the atomic resource requirement predicates in resource filters are the comparison expressions on value properties and set properties. Resource requirement predicates can be composited using conjunction, disjunction and negation. The ALTO client can send resource query with composed resource requirement predicates to ALTO server, which only returns the information of resources that satisfy such predicates to ALTO client.

The details of this new interface will be fully specified in the next version of this document.

6. Example

6.1. Protocol Extension

To allow the ALTO client to query and receive the mathematical programming constraints for a set of flows, the Filtered Cost Map and Endpoint Cost Service of the ALTO protocol need to be extended. The current design adopted in this document uses a similar approach as the ALTO-PV extension does in [DRAFT-PV]: (1) extending the FilteredCostMapCapabilities object with a new member "property-map" and (2) using a multipart service to send both the unified representation cost map and the CSTR property map together. This design is illustrated in the next few subsections.

However, for the ALTO-UR service, this is still an early stage design. As a major next step for ALTO-UR service, other design options are being investigated with the aim of enabling better

modularity and extensibility, and the protocol extension will be updated in the next version accordingly.

6.2. Workflow

A typical workflow of an ALTO client using the unified representation extension is as follows:

1. Send a GET request for the whole Information Resource Directory.

2. Look for the resource of the Cost Map Service which contains the unified representation cost type and get the resource ID of the dependent cstr property map.

3. Check whether the capabilities of the property map includes the desired "prop-types".

4. Send a unified-representation request which accepts "multipart/related" media type following "application/alto-costmap+json" or "application/endpointcost+json"

6.3. Information Resource Directory Example

An example of an Information Resource Directory is as follows. In this example, filtered cost map "cost-map-ur" supports the unified-representation extension. The property map "propmap-cstr" support two properties, "bw-cstr" and "srlg-cstr", representing bandwidth constraint and SRLG constraint, respectively.

```
{
  "meta": {
    "cost-types": {
      "ur": {
        "cost-mode": "array",
        "cost-metric": "variable-list"
        }
  }
  "resources": {
    "my-default-networkmap": {
      "uri" : "http://alto.example.com/networkmap",
      "media-type" : "application/alto-networkmap+json"
      }
    "cost-map-ur" : {
      "uri": "http://alto.example.com/costmap/ur",
      "media-type": "application/alto-costmap+json",
      "accepts": "application/alto-costmapfilter+json",
      "capabilities": {
        "cost-type-names": [ "ur"]
      },
      "property-map": "propmap-cstr",
      "uses": [ "my-default-networkmap" ]
    },
    "propmap-cstr" : {
      "uri": "http://alto.exmaple.com/propmap/cstr",
      "media-type": "application/alto-propmap+json",
      "accepts": "application/alto-propmapparams+json",
      "capabilities": {
        "domain-types": [ "cstr" ],
        "prop-types": [ "bw-cstr", "srlg-cstr" ]
      }
    }
  }
}
```

6.4. Cost Map Service Example

The following is an example of the cost map service in the ALTO-UR extension. In the returned cost map, flow 1 has two bandwidth variables, f1:bw:p1 and f2:bw:p2, and one SRLG variable, f1:srlg. And flow 2 has one bandwidth variable, f2:bw, and one SRLG variable, f2:srlg. Four mathematical programming constraints are returned in the property map. The first three are bandwidth sharing constraints, and the fourth is an SRLG constraint.

```
POST /costmap/pv HTTP/1.1
Host: alto.example.com
```

```
Accept: multipart/related, application/alto-costmap+json,
application/alto-propmap+json, application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-costmapfilter+json
{
  "cost-type": {
    "cost-mode": "array",
    "cost-metric": "variable-list"
  },
  "pids": {
    "srcs": [ "PID1" ],
    "dsts": [ "PID2", "PID3" ]
  }
}

HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: multipart/related; boundary=42

--42
Content-Type: application/alto-costmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "default-network-map",
        "tag": "75ed013b3cb58f896e839582504f622838ce670f"
      }
    ],
    "cost-type": {
      "cost-mode": "array",
      "cost-metric": "variable-list"
    },
  },

  "cost-map": {
    "PID1": {
      "PID2": [ "f1:bw:p1", "bw:p2", "f1:srlg"]
      "PID3": [ "f2:bw:p1", "f2:srlg" ]
    }
  }
}

--42
Content-Type: application/alto-propmap+json

{
```

```
  "property-map": {
    "cstr:001": { "bw-cstr": "[0][0] add [0][1] leq 100"},
    "cstr:002": { "bw-cstr": "[0][0] eq [0][1]"},
    "cstr:003": { "bw-cstr": "[1][0] add [0][0] leq 100"},
    "cstr:004": { "srlg-cstr": "[0][2] intersect [1][1] eq {2, 3, 4}"},
  }
}
```

7. Use Case: Programmable End-to-End Multi-Domain Route Control

This section describes a multi-domian use case that can benefit from a unified resource presentation extension of ALTO. Specifically, a novel multi-domain network programming framework is recently proposed to achieve programmable, end-to-end interdomain route control. Specifically, in this framework, each autonomus system (AS) deploys an ALTO server to provide a programming abstraction. Collectively, these ALTO servers provide the client a single, abstract, programmable network spanning multiple individual networks. Unlike existing SDN abstractions (e.g., OpenFlow and P4), it includes a built-in layer to extract and learn the interactions of interdomain policies of individual networks (e.g., route selection preference), providing a unified abstraction.

In particular, given a destination IP prefix p specified by a client, each autonomous system (AS) exposes its routing information base (RIB), i.e., all available routes it has to reach p, and the corresponding price for the client to use each route, by deploying an ALTO server. A client queries the available routes and the corresponding prices at different ASes. With the retrieved information, it then itereatively selects the best route based on its internal objective function and budget, and interacts with different ASes to check if the selected route is policy compliant, i.e., whether all ASes along this route will export the preceding segment to its upstream neighbor. After finding the best policy compliant route, the client then interacts with ASes of the route in a backward order along the route to setup the AS path. A blackbox based optimization algorithm has been developed to allow the client to swiftly find the best policy compliant route. A paper describing this framework has been accepted by IEEE INFOCOM 2020.

8. Ongoing Design Update: A Language-Based Resource Discovery Framework

To further investigate the feasibility, challenges and benefits of using a unified resource representation to facilitate application-network integration, the authors of this draft are investigating a language-based resource discovery framework. This framework consists of three key design points. First, this framework uses generic

mathematical programming constraints as a unified, compact representation of network information, as proposed in this draft. Second, it utilizes the equivalence between relational algebra and first order logic to provide a SQL-style language for application / ALTO client to express their intents on discovering resources in the network. Third, the framework develops a compiler to translate an application's resource discovery intent into a constraint programming problem with a set of logical constraints, whose feasible solutions correspond to qualified resources for application. In addition, an algorithm is also designed to improve the network's efficiency in finding qualified resources. Details of this framework and its early evaluation results have been accepted by ACM SIGCOMM NAI 2020 Workshop, with title ""Towards Deep Network & Application Integration: Possibilities, Challenges, and Research Directions".

9. Security and Privacy Considerations

The unified representation extension may expose more private information to applications than the ALTO base protocol does. However, as shown in the motivating example of providing SRLG information for a set of flows, this extension has the capability of exposing less private information than the ALTO-PV extension does, while having a better expressiveness on providing fine-grained resource information to applications. A systematic study on the security and privacy issues of the ALTO-UR extension is one of the major next steps.

10. References

10.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

10.2. Informative References

[DRAFT-FCS] Zhang, J., Gao, K., Wang, J., Xiang, Q., and Y. Yang, "ALTO Extension: Flow-based Cost Query", 2017, <https://datatracker.ietf.org/doc/draft-gao-alto-fcs/>.

[DRAFT-NFCHAIN] Perez, D. and C. Rothenberg, "ALTO-based Broker-assisted Multi-domain Orchestration", 2018, <https://datatracker.ietf.org/doc/html/draft-lachosrothenberg-alto-brokermdo-01>.

[DRAFT-PV]
Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Extension: Abstract Path Vector as a Cost Mode", 2015, <https://tools.ietf.org/html/draft-yang-alto-path-vector-01>.

[DRAFT-RSA]
Gao, K., Wang, X., Xiang, Q., Gu, C., Yang, Y., and G. Chen, "A Recommendation for Compressing ALTO Path Vectors", 2017, <https://datatracker.ietf.org/doc/draft-gao-alto-routing-state-abstraction/>.

[DRAFT-UP]
Roome, W., Chen, S., Randriamasy, S., Yang, Y., and J. Zhang, "Unified Properties for the ALTO Protocol", 2015, <https://datatracker.ietf.org/doc/draft-ietf-alto-unified-props-new/>.

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <https://www.rfc-editor.org/info/rfc7285>.

[RFC8189] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost Application-Layer Traffic Optimization (ALTO)", RFC 8189, DOI 10.17487/RFC8189, October 2017, <https://www.rfc-editor.org/info/rfc8189>.

Authors' Addresses

Qiao Xiang
Yale University
51 Prospect Street
New Haven, CT
USA

Email: qiao.xiang@cs.yale.edu

J. Jensen Zhang
Tongji/Yale University
51 Prospect Street
New Haven, CT
USA

Email: jingxuan.zhang@yale.edu

Franck Le
IBM
Thomas J. Watson Research Center
Yorktown Heights, NY
USA

Email: fle@us.ibm.com

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

Email: yry@cs.yale.edu