ALTO
Internet-Draft
Intended status: Standards Track
Expires: 15 January 2021

K. Gao
Sichuan University
Y. Lee

S. Randriamasy
Nokia Bell Labs
Y.R. Yang
Yale University
J. Zhang
Tongji University

14 July 2020

# ALTO Extension: Path Vector
# draft-ietf-alto-path-vector-11

Abstract

This document is an extension to the base Application-Layer Traffic Optimization protocol [RFC7285]. The current ALTO Cost Services only allow applications to obtain cost values on an end-to-end path defined by its source and destination. The present extension provides abstracted information on particular network components or elements traversed by a path between its source and destination. Examples of such abstracted components are networks, data centers or links. This is useful for applications whose performance is impacted by particular network components they traverse or by their properties. Applications having the choice among several connection paths may use this information to select paths accordingly and improve their performance. In particular, they may infer that several paths share common links and prevent traffic bottlenecks by avoiding such paths. This document introduces a new cost type called Path Vector. A Path Vector is an array of entities that each identifies an abstracted representation of a network part and that are called Abstract Network Element (ANE). Each ANE is defined by a set of properties. ANE properties are conveyed by an ALTO information resource called "Property Map", that can be packed together with the Path Vectors in a multipart response. They can also be obtained via a separate ALTO request to a Property Map. An ALTO Property Map is an extension to the ALTO protocol, that is specified in another document entitled "Unified Properties for the ALTO Protocol" [I-D.ietf-alto-unified-props-new].

Status of This Memo

This Internet-Draft will expire on 15 January 2021.

Copyright Notice



Table of Contents

## 1. Introduction

Network performance metrics are crucial to the Quality of Experience (QoE) of today's applications. The ALTO protocol allows Internet Service Providers (ISPs) to provide guidance, such as topological distance between different end hosts, to overlay applications. Thus, the overlay applications can potentially improve the QoE by better orchestrating their traffic to utilize the resources in the underlying network infrastructure.

Existing ALTO Cost Map and Endpoint Cost Service provide only cost information on an end-to-end path defined by its <source, destination> endpoints: The base protocol [RFC7285] allows the services to expose the topological distances of end-to-end paths, while various extensions have been proposed to extend the capability of these services, e.g., to express other performance metrics [I-D.ietf-alto-performance-metrics], to query multiple costs simultaneously [RFC8189], and to obtain the time-varying values [I-D.ietf-alto-cost-calendar].

While the existing extensions are sufficient for many overlay applications, however, the QoE of some overlay applications depends not only on the cost information of end-to-end paths, but also on some intermediate network components and their properties. For example, job completion time, which is an important QoE metric for a large-scale data analytics application, is impacted by shared bottlenecks inside the carrier network.

Predicting such information can be very complex without the help of the ISP [AAAI2019]. With proper guidance from the ISP, an overlay application may be able to schedule its traffic for better QoE. In the meantime, it may be helpful as well for ISPs if applications could avoid using bottlenecks or challenging the network with poorly scheduled traffic.

Despite the benefits, ISPs are not likely to expose details on their network paths: first for the sake of confidentiality, second because it may result in a huge volume and overhead, and last because it is difficult for ISPs to figure out what information and what details an

application needs. Likewise, applications do not necessarily need all the network path details and are likely not able to understand them.

Therefore, it is beneficial for both parties if an ALTO server provides ALTO clients with an "abstract network state" that provides the necessary details to applications, while hiding the network complexity and confidential information. An "abstract network state" is a selected set of abstract representations of intermediate network components traversed by the paths between <source, destination> pairs combined with properties of these components that are relevant to the overlay applications' QoE. Both an application via its ALTO client and the ISP via the ALTO server can achieve better confidentiality and resource utilization by appropriately abstracting relevant path components. The pressure on the server scalability can also be reduced by abstracting components and their properties and combining them in a single response.

This document extends [RFC7285] to allow an ALTO server convey "abstract network state", for paths defined by their <source, destination> pairs. To this end, it introduces a new cost type called "Path Vector". A Path Vector is an array of identifiers of so-called Abstract Network Element (ANE). An ANE represents an abstract intermediate component traversed by a path. It can be associated with various properties. The associations between ANEs and their properties are encoded in an ALTO information resource called Unified Property Map, which is specified in [I-D.ietf-alto-unified-props-new].

For better confidentiality, this document aims to minimize information exposure. In particular, this document enables and recommends that first ANEs are constructed on demand, and second an ANE is only associated with properties that are requested by an ALTO client. A Path Vector response involved two ALTO Maps: the Cost Map that contains the Path Vector results and the up-to-date Unified Property Map that contains the properties requested for these ANEs. To enforce consistency and improve server scalability, this document uses the "multipart/related" message defined in [RFC2387] to return the two maps in a single response.

The rest of the document are organized as follows. Section 3 introduces the extra terminologies that are used in this document. Section 4 uses an illustrative example to introduce the additional requirements of the ALTO framework, and discusses potential use cases. Section 5 gives an overview of the protocol design. Section 6 and Section 7 specify the Path Vector extension to the ALTO IRD and the information resources, with some concrete examples presented in Section 8. Section 9 discusses the backward

compatibility with the base protocol and existing extensions. Security and IANA considerations are discussed in Section 11 and Section 12 respectively.

## 2. Requirements Languages

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

When the words appear in lower case, they are to be interpreted with their natural language meanings.

## 3. Terminology

This document extends the ALTO base protocol [RFC7285] and the Unified Property Map extension [I-D.ietf-alto-unified-props-new]. In addition to the terms defined in these documents, this document also uses the following additional terms:

* Abstract Network Element (ANE): An Abstract Network Element is a representation of network components. It can be a link, a middlebox, a virtualized network function (VNF), etc., or their aggregations. An ANE can be constructed either statically in advance or on demand based on the requested information. In a response, each ANE is represented by a unique ANE Name. Note that an ALTO client must not assume ANEs in different responses but with the same ANE Name refer to the same network component(s).

* Path Vector: A Path Vector, or an ANE Path Vector, is a JSON array of ANE Names. It conveys the information that the path between a source and a destination traverses the ANEs in the same order as they appear in the Path Vector.

* Path Vector resource: A Path Vector resource refers to an ALTO resource which supports the extension defined in this document.

* Path Vector cost type: The Path Vector cost type is a special cost type, which is specified in Section 6.5. When this cost type is present in an IRD entry, it indicates that the information resource is a Path Vector resource. When this cost type is present in a Cost Map or an Endpoint Cost Map, it indicates each cost value must be interpreted as a Path Vector.

* Path Vector request: A Path Vector request refers to the POST message sent to an ALTO Path Vector resource.

* Path Vector response: A Path Vector response refers to the multipart/related message returned by a Path Vector resource.

# 4. Problem Statement

## 4.1. Design Requirements

This section gives an illustrative example of how an overlay application can benefit from the Path Vector extension.

Assume that an application has control over a set of flows, which may go through shared links or switches and share a bottleneck. The application hopes to schedule the traffic among multiple flows to get better performance. The capacity region information for those flows will benefit the scheduling. However, existing cost maps can not reveal such information.

Specifically, 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, sw2/sw4 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 link eh1 -> sw1 and link sw1 -> sw5 are 150 Mbps, and the bandwidth of the rest links are 100 Mbps.

```
                              +------+
                              |      |
                            --+ sw6  +--
                          /   |      |  \
    PID1 +-----+         /    +------+   \          +-----+  PID2
    eh1__|     |_       /                 \     ____|     |__eh2
1.2.3.4  | sw1 | \   +--|---+         +---|--+ /    | sw2 |  2.3.4.5
         +-----+  \  |      |         |      |/     +-----+
                   \_| sw5  +---------+ sw7  |
    PID3 +-----+   / |      |         |      |\     +-----+  PID4
    eh3__|     |__/  +------+         +------+ \____|     |__eh4
3.4.5.6  | sw3 |                                    | sw4 |  4.5.6.7
         +-----+                                    +-----+
```

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-scale data analytics system) which wants to optimize the total throughput of the traffic among a set of end host <source, destination> pairs, say eh1 -> eh2 and eh1 -> eh4.  The application can request a cost map providing end-to-end available bandwidth, using "availbw" as cost-metric and "numerical" as cost-mode.

The application will receive from the ALTO server that the bandwidth of eh1 -> eh2 and eh1 -> eh4 are both 100 Mbps.  But this information is not enough to determine the optimal total throughput.  Consider the following two cases:

* Case 1: If eh1 -> eh2 uses the path eh1 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh2 and eh1 -> eh4 uses path eh1 -> sw1 -> sw5 -> sw7 -> sw4 -> eh4, then the application will obtain 150 Mbps at most.

* Case 2: If eh1 -> eh2 uses the path eh1 -> sw1 -> sw5 -> sw7 -> sw2 -> eh2 and eh1 -> eh4 uses the path eh1 -> sw1 -> sw5 -> sw7 -> sw4 -> eh4, then the application will obtain only 100 Mbps at most.

To allow applications to distinguish the two aforementioned cases, the network needs to provide more details.  In particular:

* For eh1 -> eh2, the ALTO server must give more details which is critical for the overlay application to distinguish between Case 1 and Case 2 and to compute the optimal total throughput accordingly.

* The ALTO server must allow the client to distinguish the common network components shared by eh1 -> eh2 and eh1 -> eh4, e.g., eh1 - sw1 and sw1 - sw5 in Case 1.

* The ALTO server must give details on the properties of the network components used by eh1 -> eh2 and eh1 -> eh4, e.g., the available bandwidth between eh1 - sw1, sw1 - sw5, sw5 - sw7, sw5 - sw6, sw6 - sw7, sw7 - sw2, sw7 - sw4, sw2 - eh2, sw4 - eh4 in Case 1.

In general, we can conclude that to support the multiple flow scheduling use case, the ALTO framework must be extended to satisfy the following additional requirements:

AR1: An ALTO server must provide essential information on intermediate network components on the path of a <source, destination> pair that are critical to the QoE of the overlay application.

AR2: An ALTO server must provide essential information on how the paths of different <source, destination> pairs share a common network component.

AR3: An ALTO server must provide essential information on the properties associated to the network components.

The Path Vector extension defined in this document propose a solution to provide these details.

### 4.2. Recent Use Cases

While the multiple flow scheduling problem is used to help identify the additional requirements, the Path Vector extension can be applied to a wide range of applications. This section highlights some real use cases that are recently reported. See [I-D.bernstein-alto-topo] for a more comprehensive survey of use cases where extended network topology information is needed.

#### 4.2.1. Large-scale Data Analytics

One potential use case of the Path Vector extension is for large-scale data analytics such as [SENSE] and [LHC], where data of Gigabytes, Terabytes and even Petabytes are transferred. For these applications, the QoE is usually measured as the job completion time, which is related to the completion time of the slowest data transfer. With the Path Vector extension, an ALTO client can identify bottlenecks inside the network. Therefore, the overlay application can make optimal traffic distribution or resource reservation (i.e., proportional to the size of the transferred data), leading to optimal job completion time and network resource utilization.

### 4.2.2. Context-aware Data Transfer

It is sometimes important to know how the capabilities of various network components between two end hosts, especially in the mobile environment. With the Path Vector extension, an ALTO client may query the "network context" information, i.e., whether the two hosts are connected to the access network through a wireless link or a wire, and the capabilities of the access network. Thus, the client may use different data transfer mechanisms, or even deploy different 5G User Plane Functions (UPF) [I-D.ietf-dmm-5g-uplane-analysis] to optimize the data transfer.

### 4.2.3. CDN and Service Edge

A growing trend in today's applications is to bring storage and computation closer to the end user for better QoE, such as Content Delivery Network (CDN), AR/VR, and cloud gaming, as reported in various recent documents ([I-D.contreras-alto-service-edge], [I-D.huang-alto-mowie-for-network-aware-app], and [I-D.yang-alto-deliver-functions-over-networks]).

With the Path Vector extension, an ALTO server can selectively reveal the CDNs and service edges that reside along the paths between different end hosts, together with their properties such as available Service Level Agreement (SLA) plans. Otherwise, the ALTO client may have to make multiple queries and potentially with the complete list of CDNs and/or service edges. While both approaches offer the same information, making multiple queries introduces larger delay and more overhead on both the ALTO server and the ALTO client.

## 5. Path Vector Extension: Overview

This section gives a non-normative overview of the Path Vector extension. It is assumed that readers are familiar with both the base protocol [RFC7285] and the Unified Property Map extension [I-D.ietf-alto-unified-props-new].

To satisfies the additional requirements, this extension:

1. introduces Abstract Network Element (ANE) as the abstraction of intermediate network components,

2. extends the Cost Map and Endpoint Cost Service to convey the intermediate network components traversed by the path of a <source, destination> pair as Path Vectors,

3. uses the Unified Property Map to convey the association between the intermediate network components and their properties.

Thus, an ALTO client can learn about the intermediate network components that are critical to the QoE of a <source, destination> pair by investigating the corresponding Path Vector value (AR1), identify common network components if an ANE appears in the Path Vectors of multiple <source, destination> pairs (AR2), and retrieve the properties of the network components by searching the Unified Property Map (AR3).

## 5.1. Abstract Network Element

This extension introduces Abstract Network Element (ANE) as an indirect and network-agnostic way to specify an aggregation of intermediate network components between a source and a destination. Specifically, an ANE is a string of type ANEName as specified in Section 6.1 and its associated set of properties.

### 5.1.1. ANE Domain

In this extension, the associations between ANE and the properties are conveyed in a Unified Property Map. Thus, they must follow the mechanisms specified in the [I-D.ietf-alto-unified-props-new].

Specifically, this document defines a new entity domain called "ane" as specified in Section 5.1.1 and defines two initial properties for the "ane" domain.

### 5.1.2. Ephemeral ANE and Persistent ANE

For different requests, there can be different ways of grouping network components and assigning ANEs. For example, an ALTO server may define an ANE for each aggregated bottleneck link between the sources and destinations specified in the request. As the aggregated bottleneck links vary for different combinations of sources and destinations, the ANEs are ephemeral and are no longer valid after the request completes. Thus, the scope of ephemeral ANEs are limited to the corresponding Path Vector response.

While ephemeral ANEs returned by a Path Vector response do not exist beyond that response, some of them may represent entities that are persistent and defined in a standalone Property Map. Indeed, it may be useful for clients to occasionally query properties on persistent entities, without caring about the path that traverses them. Persistent entities have a persistent ID that is registered in a Property Map, together with their properties.

### 5.1.3. Property Filtering

Resource-constrained ALTO clients may benefit from the filtering of Path Vector query results at the ALTO server, as an ALTO client may only require a subset of the available properties.

Specifically, the available properties for a given resource are announced in the Information Resource Directory as a new capability called "ane-property-names". The selected properties are specified in a filter called "ane-property-names" in the request body, and the response must return and only return the selected properties for the ANEs in the response.

The "ane-property-names" capability for Cost Map and for Endpoint Cost Service are specified in Section 7.1.4 and Section 7.2.4 respectively. The "ane-property-names" filter for Cost Map and Endpoint Cost Service are specified in Section 7.1.3 and Section 7.2.3 accordingly.

## 5.2. Path Vector Cost Type

For an ALTO client to correctly interpret the Path Vector, this extension specifies a new cost type called the Path Vector cost type, which must be included both in the Information Resource Directory and the ALTO Cost Map or Endpoint Cost Map so that an ALTO client can correct interpret the cost values.

The Path Vector cost type must convey both the interpretation and semantics in the "cost-mode" and "cost-metric" respectively. Unfortunately, a single "cost-mode" value cannot fully specify the interpretation of a Path Vector, which is a compound data type. For example, in programming languages such as Java, a Path Vector will have the type of JSONArray[ANEName].

Instead of extending the "type system" of ALTO, this document takes a simple and backward compatible approach. Specifically, the "cost-mode" of the Path Vector cost type is "array", which indicates the value is a JSON array. Then, an ALTO client must check the value of the "cost-metric". If the value is "ane-path", meaning the JSON array should be further interpreted as a path of ANENames.

The Path Vector cost type is specified in Section 6.5.

### 5.3. Multipart Path Vector Response

For a basic ALTO information resource, a response contains only one type of ALTO resources, e.g., Network Map, Cost Map, or Property Map. Thus, only one round of communication is required: An ALTO client sends a request to an ALTO server, and the ALTO server returns a response, as shown in Figure 3.

```
         ALTO client                              ALTO server
              |-------------- Request ---------------->|
              |<------------- Response ----------------|
```

Figure 3: A Typical ALTO Request and Response

The Path Vector extension, on the other hand, involves two types of information resources: Path Vectors conveyed in a Cost Map or an Endpoint Cost Map, and ANE properties conveyed in a Unified Property Map. Instead of two consecutive message exchanges, the Path Vector extension enforces one round of communication. Specifically, the Path Vector extension requires the ALTO client to include the source and destination pairs and the requested ANE properties in a single request, and encapsulates both Path Vectors and properties associated with the ANEs in a single response, as shown in Figure 4.

```
         ALTO client                              ALTO server
              |------------- PV Request -------------->|
              |<----- PV Response (Cost Map Part) -----|
              |<--- PV Response (Property Map Part) ---|
```

Figure 4: The Path Vector Extension Request and Response

This design is based on the following considerations:

1. Since ANEs may be constructed on demand, and potentially based on the requested properties (See Section 5.1 for more details). If sources and destinations are not in the same request as the properties, an ALTO server either cannot construct ANEs on-demand, or must wait until both requests are received.

2. As ANEs may be constructed on demand, mappings of each ANE to its underlying network devices and resources can be specific to the request. In order to respond to the Property Map request correctly, an ALTO server must store the mapping of each Path Vector request until the client fully retrieves the property information. The "stateful" behavior may substantially harm the server scalability and potentially lead to Denial-of-Service attacks.

One approach to realize the one-round communication is to define a new media type to contain both objects, but this violates modular design. This document follows the standard-conforming usage of "multipart/related" media type defined in [RFC2387] to elegantly combine the objects. Path Vectors are encoded as a Cost Map or an Endpoint Cost Map, and the Property Map is encoded as a Unified Propert Map. They are encapsulated as parts of a multipart message. The modular composition allows ALTO servers and clients to reuse the data models of the existing information resources. Specifically, this document addresses the following practical issues using "multipart/related".

### 5.3.1. Identifying the Media Type of the Root Object

ALTO uses media type to indicate the type of an entry in the Information Resource Directory (IRD) (e.g., "application/alto-costmap+json" for Cost Map and "application/alto-endpointcost+json" for Endpoint Cost Map). Simply putting "multipart/related" as the media type, however, makes it impossible for an ALTO client to identify the type of service provided by related entries.

To address this issue, this document uses the "type" parameter to indicate the root object of a multipart/related message. For a Cost Map resource, the "media-type" in the IRD entry must be "multipart/related" with the parameter "type=application/alto-costmap+json"; for an Endpoint Cost Service, the parameter must be "type=application/alto-endpointcost+json".

### 5.3.2. References to Part Messages

The ALTO SSE extension (see [I-D.ietf-alto-incr-update-sse]) uses "client-id" to demultiplex push updates. However, "client-id" is provided for each request, which introduces ambiguity when applying SSE to a Path Vector resource.

To address this issue, an ALTO server must assign a unique identifier to each part of the "multipart/related" response message. This identifier, referred to as a Part Resource ID (See Section 6.6 for details), must be present in the part message's "Resource-Id" header. The MIME part header must also contain the "Content-Type" header, whose value is the media type of the part (e.g., "application/alto-costmap+json", "application/alto-endpointcost+json", or "application/alto-propmap+json").

If an ALTO server provides incremental updates for this Path Vector resource, it must generate incremental updates for each part separately. The client-id must have the following format:

```
pv-client-id '.' part-resource-id
```

where pv-client-id is the client-id assigned to the Path Vector request, and part-resource-id is the "Resource-Id" header value of the part. The media-type must match the "Content-Type" of the part.

The same problem applies to the part messages as well. The two parts must contain a version tag, which SHOULD contain a unique Resource ID. This document requires the resource-id in a Version Tag to have the following format:

```
pv-resource-id '.' part-resource-id
```

where pv-resource-id is the resource ID of the Path Vector resource in the IRD entry, and the part-resource-id has the same value as the "Resource-Id" header of the part.

#### 5.3.3. Order of Part Messages

According to [RFC2387], the Path Vector part, whose media type is the same as the "type" parameter of the multipart response message, is the root object. Thus, it is the element the application processes first. Even though the "start" parameter allows it to be placed anywhere in the part sequence, it is RECOMMENDED that the parts arrive in the same order as they are processed, i.e., the Path Vector part is always put as the first part, followed by the property map part. It is also RECOMMENDED that when doing so, an ALTO server SHOULD NOT set the "start" parameter, which implies the first part is the root object.

## 6. Specification: Basic Data Types

### 6.1. ANE Name

An ANE Name is encoded as a JSON string with the same format as that of the type PIDName (Section 10.1 of [RFC7285]).

The type ANEName is used in this document to indicate a string of this format.

### 6.2. ANE Domain

The ANE domain associates property values with the Abstract Network Elements in a Property Map. Accordingly, the ANE domain always depends on a Property Map.

### 6.2.1. Entity Domain Type

ane

### 6.2.2. Entity Identifier Encoding

The entity identifier of the "ane" domain has the same format as defined in Section 5.1.3 in [I-D.ietf-alto-unified-props-new], and the DomainTypeSpecificEntityID part has the same format as the ANEName type.

### 6.2.3. Hierarchy and Inheritance

There is no hierarchy or inheritance for properties associated with ANEs.

### 6.2.4. Media Type of Defining Resource

When resource specific domains are defined with entities of domain type "ane", the defining resource for entity domain type "ane" MUST be a Property Map. The media type of defining resources for the "ane" domain is:

application/alto-propmap+json

Specifically, the defining resource of ephemeral ANEs is the Property Map part of the multipart response. The defining resource of persistent ANEs is the Property Map on which standalone queries for properties of persistent ANEs are made.

## 6.3. ANE Property Name

An ANE Property Name is encoded as a JSON string with the same format as that of Entity Property Name (Section 5.2.2 of [I-D.ietf-alto-unified-props-new]).

## 6.4. Initial ANE Property Types

In this document, two initial ANE property types are specified, "max-reservable-bandwidth" and "persistent-entity-id".

Note that the two property types defined in this document do not depend on any information resource, so their ResourceID part must be empty.

```
                                        ----- L1
                                       /
           PID1   +---------------+ 10 Gbps +----------+    PID3
    1.2.3.0/24+---+ +-----------+ +---------+          +---+3.4.5.0/24
                  | |   MEC1    | |         |          |
                  | +-----------+ |   +-----+          |
           PID2   |               |   |     +----------+
    2.3.4.0/24+---+               |   |         NET3
                  |               |   | 15 Gbps
                  |               |   |        \
                  +---------------+   |         -------- L2
                        NET1          |
                               +---------------+
                               | +-----------+ |   PID4
                               | |   MEC2    | +---+4.5.6.0/24
                               | +-----------+ |
                               +---------------+
                                     NET2
```

Figure 5: Examples of ANE Properties

In this document, Figure 5 is used to illustrate the use of the two initial ANE property types. There are 3 sub-networks (NET1, NET2 and NET3) and two interconnection links (L1 and L2). It is assumed that each sub-network has sufficiently large bandwidth to be reserved.

### 6.4.1. New ANE Property Type: Maximum Reservable Bandwidth

Identifier: "max-reservable-bandwidth"

Intended Semantics: The maximum reservable bandwidth property stands for the maximum bandwidth that can be reserved for all the traffic that traverses an ANE. The value MUST be encoded as a non-negative numerical cost value as defined in Section 6.1.2.1 of [RFC7285] and the unit is bit per second. If this property is requested but not present in an ANE, it MUST be interpreted as that the ANE does not support bandwidth reservation.

Security Considerations: ALTO entity properties expose information to ALTO clients. ALTO service providers should be made aware of the security ramifications related to the exposure of an entity property.

To illustrate the use of "max-reservable-bandwidth", consider the network in Figure 5. An ALTO server can create an ANE for each interconnection link, where the initial value for "max-reservable-bandwidth" is the link capacity.

### 6.4.2. New ANE Property Type: Persistent Entity ID

Identifier: "persistent-entity-id"

Intended Semantics: The persistent entity ID property is the entity identifier of the persistent ANE associated with an ephemeral ANE. The value of this property is encoded with the format defined in Section 5.1.3 of [I-D.ietf-alto-unified-props-new]. In this format, the entity ID combines:

* a defining information resource for the ANE on which a "persistent-entity-id" is queried, which is the property map defining the ANE as a persistent entity, together with the properties

* the persistent name of the ANE in this property map

With this format, the client has all the needed information for further standalone query properties on the persistent ANE.

Security Considerations: ALTO entity properties expose information to ALTO clients. ALTO service providers should be made aware of the security ramifications related to the exposure of an entity property.

To illustrate the use of "persistent-entity-id", consider the network in Figure 5. Assume the ALTO server has a Property Map resource called "mec-props" that defines persistent ANEs "MEC1" and "MEC2" that represent the corresponding mobile edge computing (MEC) clusters. The "persistent-entity-id" of the ephemeral ANE that is associated with MEC1 has the value "mec-props.ane:MEC1".

## 6.5. Path Vector Cost Type

This document defines a new cost type, which is referred to as the "Path Vector" cost type. An ALTO server MUST offer this cost type if it supports the Path Vector extension.

### 6.5.1. Cost Metric: ane-path

The cost metric "ane-path" indicates the value of such a cost type conveys an array of ANE names, where each ANE name uniquely represents an ANE traversed by traffic from a source to a destination.

#### 6.5.2. Cost Mode: array

The cost mode "array" indicates that every cost value in a Cost Map or an Endpoint Cost Map MUST be interpreted as a JSON array object.

Note that this cost mode only requires the cost value to be a JSON array of JSONValue. However, an ALTO server that enables this extension MUST return a JSON array of ANEName (Section 6.1) when the cost metric is "ane-path".

### 6.6. Part Resource ID

A Part Resource ID is encoded as a JSON string with the same format as that of the type ResourceID (Section 10.2 of [RFC7285]).

NOTE: Even though the client-id assigned to a Path Vector request and the Part Resource ID may contain up to 64 characters by their own definition, their concatenation (see Section 5.3.2) MUST also conform to the same length constraint. The same requirement applies to the resource ID of the Path Vector resource, too. Thus, it is RECOMMENDED to limit the length of resource ID and client ID related to a Path Vector resource to 31 characters.

## 7. Specification: Service Extensions

### 7.1. Multipart Filtered Cost Map for Path Vector

This document introduces a new ALTO resource called multipart filtered cost map resource, which allows an ALTO server to provide other ALTO resources associated to the cost map resource in the same response.

#### 7.1.1. Media Type

The media type of the multipart filtered cost map resource is "multipart/related;type=application/alto-costmap+json".

#### 7.1.2. HTTP Method

The multipart filtered cost map is requested using the HTTP POST method.

### 7.1.3. Accept Input Parameters

The input parameters of the multipart filtered cost map are supplied in the body of an HTTP POST request. This document extends the input parameters to a filtered cost map with a data format indicated by the media type "application/alto-costmapfilter+json", which is a JSON object of type PVReqFilteredCostMap, where:

```
object {
  [EntityPropertyName ane-property-names<0..*>;]
} PVReqFilteredCostMap : ReqFilteredCostMap;
```

with fields:

ane-property-names: A list of properties that are associated with the ANEs. Each property in this list MUST match one of the supported ANE properties indicated in the resource's "ane-property-names" capability. If the field is NOT present, it MUST be interpreted as an empty list, indicating that the ALTO server MUST NOT return any property in the Unified Property part.

Example: Consider the network in Figure 1. If an ALTO client wants to query the "max-reservable-bandwidth" between PID1 and PID2, it can submit the following request.

```
POST /costmap/pv HTTP/1.1
Host: alto.example.com
Accept: multipart/related;type=application/alto-costmap+json,
        application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-costmapfilter+json

{
  "cost-type": {
    "cost-mode": "array",
    "cost-metric": "ane-path"
  },
  "pids": {
    "srcs": [ "PID1" ],
    "dsts": [ "PID2" ]
  },
  "ane-property-names": [ "max-reservable-bandwidth" ]
}
```

### 7.1.4. Capabilities

The multipart filtered cost map resource extends the capabilities defined in Section 11.3.2.4 of [RFC7285]. The capabilities are defined by a JSON object of type PVFilteredCostMapCapabilities:

```
object {
  [EntityPropertyName ane-property-names<0..*>;]
} PVFilteredCostMapCapabilities : FilteredCostMapCapabilities;
```

with fields:

cost-type-names: The "cost-type-names" field MUST only include the Path Vector cost type, unless explicitly documented by a future extension. This also implies that the Path Vector cost type MUST be defined in the "cost-types" of the Information Resource Directory's "meta" field.

cost-constraints: If the "cost-type-names" field includes the Path Vector cost type, "cost-constraints" field MUST be "false" or not present unless specifically instructed by a future document.

testable-cost-type-names: If the "cost-type-names" field includes the Path Vector cost type, the Path Vector cost type MUST NOT be included in the "testable-cost-type-names" field unless specifically instructed by a future document.

ane-property-names: Defines a list of ANE properties that can be returned. If the field is NOT present, it MUST be interpreted as an empty list, indicating the ALTO server cannot provide any ANE property.

### 7.1.5. Uses

This member MUST include the resource ID of the network map based on which the PIDs are defined. If this resource supports "persistent-entity-id", it MUST also include the defining resources of persistent ANEs that may appear in the response.

### 7.1.6. Response

The response MUST indicate an error, using ALTO protocol error handling, as defined in Section 8.5 of [RFC7285], if the request does no.

The "Content-Type" header of the response MUST be "multipart/related" as defined by [RFC2387] with the following parameters:

type: The type parameter MUST be "application/alto-costmap+json". Note that [RFC2387] permits both parameters with and without the double quotes.

start: The start parameter is as defined in [RFC2387]. If present, it MUST have the same value as the "Resource-Id" header of the Path Vector part.

boundary: The boundary parameter is as defined in [RFC2387].

The body of the response consists of two parts:

* The Path Vector part MUST include "Resource-Id" and "Content-Type" in its header. The value of "Resource-Id" MUST has the format of a Part Resource ID. The "Content-Type" MUST be "application/alto-costmap+json".

  The body of the Path Vector part MUST be a JSON object with the same format as defined in Section 11.2.3.6 of [RFC7285]. The JSON object MUST include the "vtag" field in the "meta" field, which provides the version tag of the returned cost map. The resource ID of the version tag MUST follow the format in Section 5.3.2. The "meta" field MUST also include the "dependent-vtags" field, whose value is a single-element array to indicate the version tag of the network map used, where the network map is specified in the "uses" attribute of the multipart filtered cost map resource in IRD.

* The Unified Property Map part MUST also include "Resource-Id" and "Content-Type" in its header. The value of "Resource-Id" has the format of a Part Resource ID. The "Content-Type" MUST be "application/alto-propmap+json".

  The body of the Unified Property Map part MUST be a JSON object with the same format as defined in Section 4.6 of [I-D.ietf-alto-unified-props-new]. The JSON object MUST include the "dependent-vtags" field in the "meta" field. The value of the "dependent-vtags" field MUST be an array of VersionTag objects as defined by Section 10.3 of [RFC7285]. The "vtag" of the Path Vector part MUST be included in the "dependent-vtags". If "persistent-entity-id" is requested, the version tags of the dependent resources that may expose the entities in the response MUST also be included. The PropertyMapData has one member for each ANEName that appears in the Path Vector part, which is an entity identifier belonging to the self-defined entity domain as defined in Section 5.1.2.3 of [I-D.ietf-alto-unified-props-new]. The EntityProps has one member for each property requested by an ALTO client if applicable.

If the "start" parameter is not present, the Path Vector part MUST be the first part in the multipart response.

Example: Consider the network in Figure 1. The response of the example request in Section 7.1.3 is as follows, where "ANE1" represents the aggregation of all the switches in the network.

```
HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: multipart/related; boundary=example-1;
              type=application/alto-costmap+json

--example-1
Resource-Id: costmap
Content-Type: application/alto-costmap+json

{
  "meta": {
    "vtag": {
      "resource-id": "filtered-cost-map-pv.costmap",
      "tag": "d827f484cb66ce6df6b5077cb8562b0a"
    },
    "dependent-vtags": [
      {
        "resource-id": "my-default-networkmap",
        "tag": "75ed013b3cb58f896e839582504f6228"
      }
    ],
    "cost-type": { "cost-mode": "array", "cost-metric": "ane-path" }
  },
  "cost-map": {
    "PID1": { "PID2": ["ANE1"] }
  }
}
--example-1
Resource-Id: propmap
Content-Type: application/alto-propmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "filtered-cost-map-pv.costmap",
        "tag": "d827f484cb66ce6df6b5077cb8562b0a"
      }
    ]
  },
  "property-map": {
    ".ane:ANE1": { "max-reservable-bandwidth": 100000000 }
  }
}
```

### 7.2. Multipart Endpoint Cost Service for Path Vector

This document introduces a new ALTO resource called multipart endpoint cost resource, which allows an ALTO server to provide other ALTO resources associated to the endpoint cost resource in the same response.

#### 7.2.1. Media Type

The media type of the multipart endpoint cost resource is "multipart/related;type=application/alto-endpointcost+json".

#### 7.2.2. HTTP Method

The multipart endpoint cost resource is requested using the HTTP POST method.

#### 7.2.3. Accept Input Parameters

The input parameters of the multipart endpoint cost resource are supplied in the body of an HTTP POST request. This document extends the input parameters to an endpoint cost map with a data format indicated by the media type "application/alto-endpointcostparams+json", which is a JSON object of type PVEndpointCostParams, where

```
object {
  [EntityPropertyName ane-property-names<0..*>;]
} PVReqEndpointcost : ReqEndpointcost;
```

with fields:

ane-property-names: This document defines the "ane-property-names" in PVReqEndpointcost as the same as in PVReqFilteredCostMap. See Section 7.1.3.

Example: Consider the network in Figure 1. If an ALTO client wants to query the "max-reservable-bandwidth" between eh1 and eh2, it can submit the following request.

```
POST /ecs/pv HTTP/1.1
Host: alto.example.com
Accept: multipart/related;type=application/alto-endpointcost+json,
        application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-endpointcostparams+json

{
  "cost-type": {
    "cost-mode": "array",
    "cost-metric": "ane-path"
  },
  "endpoints": {
    "srcs": [ "ipv4:1.2.3.4" ],
    "dsts": [ "ipv4:2.3.4.5" ]
  },
  "ane-property-names": [ "max-reservable-bandwidth" ]
}
```

### 7.2.4. Capabilities

The capabilities of the multipart endpoint cost resource are defined by a JSON object of type PVEndpointcostCapabilities, which is defined as the same as PVFilteredCostMapCapabilities. See Section 7.1.4.

### 7.2.5. Uses

If this resource supports "persistent-entity-id", it MUST also include the defining resources of persistent ANEs that may appear in the response.

### 7.2.6. Response

The response MUST indicate an error, using ALTO protocol error handling, as defined in Section 8.5 of [RFC7285], if the request is invalid.

The "Content-Type" header of the response MUST be "multipart/related" as defined by [RFC7285] with the following parameters:

type: The type parameter MUST be "application/alto-endpointcost+json".

start: The start parameter is as defined in Section 7.1.6.

boundary: The boundary parameter is as defined in [RFC2387].

The body consists of two parts:

* The Path Vector part MUST include "Resource-Id" and "Content-Type" in its header. The value of "Resource-Id" MUST has the format of a Part Resource ID. The "Content-Type" MUST be "application/alto-endpointcost+json".

  The body of the Path Vector part MUST be a JSON object with the same format as defined in Section 11.5.1.6 of [RFC7285]. The JSON object MUST include the "vtag" field in the "meta" field, which provides the version tag of the returned endpoint cost map. The resource ID of the version tag MUST follow the format in Section 5.3.2.

* The Unified Property Map part MUST also include "Resource-Id" and "Content-Type" in its header. The value of "Resource-Id" MUST has the format of a Part Resource ID. The "Content-Type" MUST be "application/alto-propmap+json".

  The body of the Unified Property Map part MUST be a JSON object with the same format as defined in Section 4.6 of [I-D.ietf-alto-unified-props-new]. The JSON object MUST include the "dependent-vtags" field in the "meta" field. The value of the "dependent-vtags" field MUST be an array of VersionTag objects as defined by Section 10.3 of [RFC7285]. The "vtag" of the Path Vector part MUST be included in the "dependent-vtags". If "persistent-entity-id" is requested, the version tags of the dependent resources that may expose the entities in the response MUST also be included. The PropertyMapData has one member for each ANEName that appears in the Path Vector part, which is an entity identifier belonging to the self-defined entity domain as defined in Section 5.1.2.3 of [I-D.ietf-alto-unified-props-new]. The EntityProps has one member for each property requested by the ALTO client if applicable.

If the "start" parameter is not present, the Path Vector part MUST be the first part in the multipart response.

Example: Consider the network in Figure 1. The response of the example request in Section 7.2.3 is as follows.

```
HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: multipart/related; boundary=example-1;
              type=application/alto-endpointcost+json

--example-1
Resource-Id: ecs
Content-Type: application/alto-endpointcost+json

{
  "meta": {
    "vtag": {
      "resource-id": "ecs-pv.costmap",
      "tag": "d827f484cb66ce6df6b5077cb8562b0a"
    },
    "dependent-vtags": [
      {
        "resource-id": "my-default-networkmap",
        "tag": "75ed013b3cb58f896e839582504f6228"
      }
    ],
    "cost-type": { "cost-mode": "array", "cost-metric": "ane-path" }
  },
  "cost-map": {
    "ipv4:1.2.3.4": { "ipv4:2.3.4.5": ["ANE1"] }
  }
}
--example-1
Resource-Id: propmap
Content-Type: application/alto-propmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "ecs-pv.costmap",
        "tag": "d827f484cb66ce6df6b5077cb8562b0a"
      }
    ]
  },
  "property-map": {
    ".ane:ANE1": { "max-reservable-bandwidth": 100000000 }
  }
}
```

## 8. Examples

This section lists some examples of Path Vector queries and the corresponding responses. Some long lines are truncated for better readability.

### 8.1. Example: Information Resource Directory

To give a comprehensive example of the Path Vector extension, we consider the network in Figure 5. The example ALTO server provides the following information resources:

* "my-default-networkmap": A Network Map resource which contains the PIDs in the network.

* "filtered-cost-map-pv": A Multipart Filtered Cost Map resource for Path Vector, which exposes the "max-reservable-bandwidth" property for the PIDs in "my-default-networkmap".

* "ane-props": A filtered Unified Property resource that exposes the information for persistent ANEs in the network.

* "endpoint-cost-pv": A Multipart Endpoint Cost Service for Path Vector, which exposes the "max-reservable-bandwidth" and the "persistent-entity-id" properties.

* "update-pv": An Update Stream service, which provides the incremental update service for the "endpoint-cost-pv" service.

Below is the Information Resource Directory of the example ALTO server. To enable the Path Vector extension, the "path-vector" cost type (Section 6.5) is defined in the "cost-types" of the "meta" field, and is included in the "cost-type-names" of resources "filetered-cost-map-pv" and "endpoint-cost-pv".


```
{
  "meta": {
    "cost-types": {
      "path-vector": {
        "cost-mode": "array",
        "cost-metric": "ane-path"
      }
    }
  },
  "resources": {
    "my-default-networkmap": {
      "uri" : "https://alto.example.com/networkmap",
      "media-type" : "application/alto-networkmap+json"
```

```
      },
      "filtered-cost-map-pv": {
        "uri": "https://alto.example.com/costmap/pv",
        "media-type": "multipart/related;
                       type=application/alto-costmap+json",
        "accepts": "application/alto-costmapfilter+json",
        "capabilities": {
          "cost-type-names": [ "path-vector" ],
          "ane-property-names": [ "max-reservable-bandwidth" ]
        },
        "uses": [ "my-default-networkmap" ]
      },
      "ane-props": {
        "uri": "https://alto.example.com/ane-props",
        "media-type": "application/alto-propmap+json",
        "accepts": "application/alto-propmapparams+json",
        "capabilities": {
          "mappings": {
            ".ane": [ "cpu" ]
          }
        }
      },
      "endpoint-cost-pv": {
        "uri": "https://alto.exmaple.com/endpointcost/pv",
        "media-type": "multipart/related;
                       type=application/alto-endpointcost+json",
        "accepts": "application/alto-endpointcostparams+json",
        "capabilities": {
          "cost-type-names": [ "path-vector" ],
          "ane-property-names": [
            "max-reservable-bandwidth", "persistent-entity-id"
          ]
        },
        "uses": [ "ane-props" ]
      },
      "update-pv": {
        "uri": "https://alto.example.com/updates/pv",
        "media-type": "text/event-stream",
        "uses": [ "endpoint-cost-pv" ],
        "accepts": "application/alto-updatestreamparams+json",
        "capabilities": {
          "support-stream-control": true
        }
      }
    }
  }
```

### 8.2. Example: Multipart Filtered Cost Map

The following examples demonstrate the request to the "filtered-cost-map-pv" resource and the corresponding response.

The request uses the "path-vector" cost type in the "cost-type" field. The "ane-property-names" field is missing, indicating that the client only requests for the Path Vector but not the ANE properties.

The response consists of two parts. The first part returns the array of ANEName for each source and destination pair. There are two ANEs, where "L1" represents the interconnection link L1, and "L2" represents the interconnection link L2.

The second part returns an empty Property Map. Note that the ANE entries are omitted since they have no properties (See Section 3.1 of [I-D.ietf-alto-unified-props-new]).

```
POST /costmap/pv HTTP/1.1
Host: alto.example.com
Accept: multipart/related;type=application/alto-costmap+json,
        application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-costmapfilter+json

{
  "cost-type": {
    "cost-mode": "array",
    "cost-metric": "ane-path"
  },
  "pids": {
    "srcs": [ "PID1" ],
    "dsts": [ "PID3", "PID4" ]
  }
}

HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: multipart/related; boundary=boundary;
              type=application/alto-costmap+json

--boundary
Resource-Id: costmap
Content-Type: application/alto-costmap+json

{
  "meta": {
```

```
        "vtag": {
          "resource-id": "filtered-cost-map-pv.costmap",
          "tag": "d827f484cb66ce6df6b5077cb8562b0a"
        },
        "dependent-vtags": [
          {
            "resource-id": "my-default-networkmap",
            "tag": "75ed013b3cb58f896e839582504f6228"
          }
        ],
        "cost-type": {
          "cost-mode": "array",
          "cost-metric": "ane-path"
        }
      },
      "cost-map": {
        "PID1": {
          "PID3": [ "L1" ],
          "PID4": [ "L1", "L2" ]
        }
      }
    }
    --boundary
    Resource-Id: propmap
    Content-Type: application/alto-propmap+json

    {
      "meta": {
        "dependent-vtags": [
          {
            "resource-id": "filtered-cost-map-pv.costmap",
            "tag": "d827f484cb66ce6df6b5077cb8562b0a"
          }
        ]
      },
      "property-map": {
      }
    }
```

### 8.3. Example: Multipart Endpoint Cost Resource

The following examples demonstrate the request to the "endpoint-cost-pv" resource and the corresponding response.

The request uses the path vector cost type in the "cost-type" field, and queries the Maximum Reservable Bandwidth ANE property and the Persistent Entity property.

The response consists of two parts. The first part returns the array of ANEName for each valid source and destination pair, where "NET1" represent sub-network NET1, and "AGGR" is the aggregation of L1 and NET3.

The second part returns the requested properties of ANEs. Since NET1 has sufficient bandwidth, it sets the "max-reservable-bandwidth" to a sufficiently large number. It also represents a persistent ANE defined in the "ane-props" resource, identified by "ane-props.ane:datacenter1". The aggregated "max-reservable-bandwidth" of ane:AGGR is constrained by the link capacity of L1. The "persistent-entity-id" property is omitted as both L1 and NET3 do not represent any persistent entity.

```
POST /endpointcost/pv HTTP/1.1
Host: alto.example.com
Accept: multipart/related;
        type=application/alto-endpointcost+json,
        application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-endpointcostparams+json

{
  "cost-type": {
    "cost-mode": "array",
    "cost-metric": "ane-path"
  },
  "endpoints": {
    "srcs": [ "ipv4:1.2.3.4", "ipv4:2.3.4.5" ],
    "dsts": [ "ipv4:3.4.5.6" ]
  },
  "ane-property-names": [
    "max-reservable-bandwidth",
    "persistent-entity-id"
  ]
}

HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: multipart/related; boundary=boundary;
              type=application/alto-endpointcost+json

--boundary
Resource-Id: ecs
Content-Type: application/alto-endpointcost+json

{
  "meta": {
```

```
      "vtags": {
        "resource-id": "endpoint-cost-pv.ecs",
        "tag": "bb6bb72eafe8f9bdc4f335c7ed3b10822a391cef"
      },
      "cost-type": {
        "cost-mode": "array",
        "cost-metric": "ane-path"
      }
    },
    "endpoint-cost-map": {
      "ipv4:1.2.3.4": {
        "ipv4:3.4.5.6":   [ "NET1", "AGGR" ]
      },
      "ipv4:2.3.4.5": {
        "ipv4:3.4.5.6":   [ "NET1", "AGGR" ]
      }
    }
  }
  --boundary
  Resource-Id: propmap
  Content-Type: application/alto-propmap+json

  {
    "meta": {
      "dependent-vtags": [
        {
          "resource-id": "endpoint-cost-pv.ecs",
          "tag": "bb6bb72eafe8f9bdc4f335c7ed3b10822a391cef"
        },
        {
          "resource-id": "ane-props",
          "tag": "bf3c8c1819d2421c9a95a9d02af557a3"
        }
      ]
    },
    "property-map": {
      ".ane:NET1": {
        "max-reservable-bandwidth": 50000000000,
        "persistent-entity-id": "ane-props.ane:datacenter1",
      },
      ".ane:AGGR": {
        "max-reservable-bandwidth": 10000000000
      }
    }
  }
```


After the client obtains "ane-props.ane:datacenter1", it can query the "ane-props" resource to get the properties of the persistent ANE.

## 8.4. Example: Incremental Updates

In this example, an ALTO client subscribes to the incremental update for the multipart endpoint cost resource "endpoint-cost-pv".

```
POST /updates/pv HTTP/1.1
Host: alto.example.com
Accept: text/event-stream
Content-Type: application/alto-updatestreamparams+json
Content-Length: [TBD]

{
  "add": {
    "ecspvsub1": {
      "resource-id": "endpoint-cost-pv",
      "input": <ecs-input>
    }
  }
}
```

Based on the server-side process defined in [I-D.ietf-alto-incr-update-sse], the ALTO server will send the "control-uri" first using Server-Sent Event (SSE), followed by the full response of the multipart message.

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/event-stream

event: application/alto-updatestreamcontrol+json
data: {"control-uri": "https://alto.example.com/updates/streams/123"}

event: multipart/related;boundary=boundary;
       type=application/alto-endpointcost+json,ecspvsub1
data: --boundary
data: Resource-ID: ecsmap
data: Content-Type: application/alto-endpointcost+json
data:
data: <endpoint-cost-map-entry>
data: --boundary
data: Resource-ID: propmap
data: Content-Type: application/alto-propmap+json
data:
data: <property-map-entry>
data: --boundary--
```

When the contents change, the ALTO server will publish the updates for each node in this tree separately.

```
event: application/merge-patch+json, ecspvsub1.ecsmap
data: <Merge patch for endpoint-cost-map-update>

event: application/merge-patch+json, ecspvsub1.propmap
data: <Merge patch for property-map-update>
```

## 9. Compatibility

### 9.1. Compatibility with Legacy ALTO Clients/Servers

The multipart filtered cost map resource and the multipart endpoint cost resource has no backward compatibility issue with legacy ALTO clients and servers. Although these two types of resources reuse the media types defined in the base ALTO protocol for the accept input parameters, they have different media types for responses. If the ALTO server provides these two types of resources, but the ALTO client does not support them, the ALTO client will ignore the resources without conducting any incompatibility.

### 9.2. Compatibility with Multi-Cost Extension

This document does not specify how to integrate the Path Vector cost type with the multi-cost extension [RFC8189]. While it is not RECOMMENDED to put the Path Vector cost type with other cost types in a single query, there is no compatible issue.

### 9.3. Compatibility with Incremental Update

The extension specified in this document is not compatible with the original incremental update extension [I-D.ietf-alto-incr-update-sse]. A legacy ALTO client cannot recognize the compound client-id, and a legacy ALTO server may use the same client-id for updates of both parts.

ALTO clients and servers must follow the specifications given in this document to ensure compatibility with the incremental update extension.

### 9.4. Compatibility with Cost Calendar

The extension specified in this document is compatible with the Cost Calendar extension [I-D.ietf-alto-cost-calendar]. When used together with the Cost Calendar extension, the cost value between a source and a destination is an array of path vectors, where the k-th path vector refers to the abstract network paths traversed in the k-th time interval by traffic from the source to the destination.

When used with time-varying properties, e.g., maximum reservable bandwidth (maxresbw), a property of a single entity may also have different values in different time intervals. In this case, an ANE with different property values must be considered as different ANEs.

The two extensions combined together can provide the historical network correlation information for a set of source and destination pairs. A network broker or client may use this information to derive other resource requirements such as Time-Block-Maximum Bandwidth, Bandwidth-Sliding-Window, and Time-Bandwidth-Product (TBP) (See [SENSE] for details).

# 10. General Discussions

## 10.1. Constraint Tests for General Cost Types

The constraint test is a simple approach to query the data. It allows users to filter the query result by specifying some boolean tests. This approach is already used in the ALTO protocol. [RFC7285] and [RFC8189] allow ALTO clients to specify the "constraints" and "or-constraints" tests to better filter the result.

However, the current syntax can only be used to test scalar cost types, and cannot easily express constraints on complex cost types, e.g., the Path Vector cost type defined in this document.

In practice, developing a language for general-purpose boolean tests can be complex and is likely to be a duplicated work. Thus, it is worth looking into the direction of integrating existing well-developed query languages, e.g., XQuery and JSONiq, or their subset with ALTO.

Filtering the Path Vector results or developing a more sophisticated filtering mechanism is beyond the scope of this document.

## 10.2. General Multipart Resources Query

Querying multiple ALTO information resources continuously may be a general requirement. And the coming issues like inefficiency and inconsistency are also general. There is no standard solving these issues yet. So we need some approach to make the ALTO client request the compound ALTO information resources in a single query.

# 11. Security Considerations

This document is an extension of the base ALTO protocol, so the Security Considerations [RFC7285] of the base ALTO protocol fully apply when this extension is provided by an ALTO server.

The Path Vector extension requires additional considerations on two security considerations discussed in the base protocol: confidentiality of ALTO information (Section 15.3 of [RFC7285]) and availability of ALTO service (Section 15.5 of [RFC7285]).

For confidentiality of ALTO information, a network operator should be aware of that this extension may introduce a new risk: the Path Vector information may make network attacks easier. For example, as the Path Vector information may reveal more fine-grained internal network structures than the base protocol, an ALTO client may detect the bottleneck link and start a distributed denial-of-service (DDoS) attack involving minimal flows to conduct the in-network congestion.

To mitigate this risk, the ALTO server should consider protection mechanisms to reduce information exposure or obfuscate the real information, in particular, in settings where the network and the application do not belong to the same trust domain. But the implementation of Path Vector extension involving reduction or obfuscation should guarantee the requested properties are still accurate.

For availability of ALTO service, an ALTO server should be cognizant that using Path Vector extension might have a new risk: frequent requesting for path vectors might conduct intolerable increment of the server-side storage and break the ALTO server. It is known that the computation of Path Vectors is unlikely to be cacheable, in that the results will depend on the particular requests (e.g., where the flows are distributed). Hence, the service providing Path Vectors may become an entry point for denial-of-service attacks on the availability of an ALTO server. To avoid this risk, authenticity and authorization of this ALTO service may need to be better protected.

## 12. IANA Considerations

### 12.1. ALTO Entity Domain Registry

This document registers a new entry to the ALTO Domain Entity Registry, as instructed by Section 12.2 of [I-D.ietf-alto-unified-props-new]. The new entry is as shown below in Table 1.

| Identifier | Entity Address Encoding | Hierarchy & Inheritance | Media Type of Defining Resource |
|---|---|---|---|
| ane | See Section 6.2.2 | None | application/alto-propmap+json |

Table 1: ALTO Entity Domain

Identifier: See Section 6.2.1.

Entity Identifier Encoding: See Section 6.2.2.

Hierarchy: None

Inheritance: None

Media Type of Defining Resource: See Section 6.2.4.

Security Considerations: In some usage scenarios, ANE addresses carried in ALTO Protocol messages may reveal information about an ALTO client or an ALTO service provider. Applications and ALTO service providers using addresses of ANEs will be made aware of how (or if) the addressing scheme relates to private information and network proximity, in further iterations of this document.

## 12.2. ALTO Entity Property Type Registry

Two initial entries are registered to the ALTO Domain "ane" in the "ALTO Entity Property Type Registry", as instructed by Section 12.3 of [I-D.ietf-alto-unified-props-new]. The two new entries are shown below in Table 2.

| Identifier | Intended Semantics |
|---|---|
| max-reservable-bandwidth | See Section 6.4.1 |
| persistent-entity-id | See Section 6.4.2 |

Table 2: Initial Entries for ane Domain in the ALTO Entity Property Types Registry

## 13. Acknowledgments

The authors would like to thank discussions with Andreas Voellmy, Erran Li, Haibin Song, Haizhou Du, Jiayuan Hu, Qiao Xiang, Tianyuan Liu, Xiao Shi, Xin Wang, and Yan Luo. The authors thank Greg Bernstein (Grotto Networks), Dawn Chen (Tongji University), Wendy Roome, and Michael Scharf for their contributions to earlier drafts.

## 14. References

### 14.1. Normative References

[I-D.ietf-alto-cost-calendar]
Randriamasy, S., Yang, Y., WU, Q., Lingli, D., and N. Schwan, "Application-Layer Traffic Optimization (ALTO) Cost Calendar", Work in Progress, Internet-Draft, draft-ietf-alto-cost-calendar-21, 17 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-alto-cost-calendar-21.txt>.

[I-D.ietf-alto-incr-update-sse]
Roome, W. and Y. Yang, "ALTO Incremental Updates Using Server-Sent Events (SSE)", Work in Progress, Internet-Draft, draft-ietf-alto-incr-update-sse-22, 20 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-alto-incr-update-sse-22.txt>.

[I-D.ietf-alto-performance-metrics]
WU, Q., Yang, Y., Lee, Y., Dhody, D., Randriamasy, S., and L. Contreras, "ALTO Performance Cost Metrics", Work in Progress, Internet-Draft, draft-ietf-alto-performance-metrics-11, 12 June 2020, <http://www.ietf.org/internet-drafts/draft-ietf-alto-performance-metrics-11.txt>.

[I-D.ietf-alto-unified-props-new]
Roome, W., Randriamasy, S., Yang, Y., Zhang, J., and K. Gao, "Unified Properties for the ALTO Protocol", Work in Progress, Internet-Draft, draft-ietf-alto-unified-props-new-11, 9 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-alto-unified-props-new-11.txt>.

[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>.

[RFC2387] Levinson, E., "The MIME Multipart/Related Content-type",

RFC 2387, DOI 10.17487/RFC2387, August 1998, <https://www.rfc-editor.org/info/rfc2387>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

## 14.2. Informative References

[AAAI2019] Xiang, Q., Yu, H., Aspnes, J., Le, F., Kong, L., and Y.R. Yang, "Optimizing in the dark: Learning an optimal solution through a simple request interface", Proceedings of the AAAI Conference on Artificial Intelligence 33, 1674-1681 , 2019.

[I-D.bernstein-alto-topo]
Bernstein, G., Yang, Y., and Y. Lee, "ALTO Topology Service: Uses Cases, Requirements, and Framework", Work in Progress, Internet-Draft, draft-bernstein-alto-topo-00, 21 October 2013, <http://www.ietf.org/internet-drafts/draft-bernstein-alto-topo-00.txt>.

[I-D.contreras-alto-service-edge]
Contreras, L., Perez, D., and C. Rothenberg, "Use of ALTO for Determining Service Edge", Work in Progress, Internet-Draft, draft-contreras-alto-service-edge-00, 4 November 2019, <http://www.ietf.org/internet-drafts/draft-contreras-alto-service-edge-00.txt>.

[I-D.huang-alto-mowie-for-network-aware-app]
Huang, W., Zhang, Y., Yang, R., Xiong, C., Lei, Y., Han, Y., and G. Li, "MoWIE for Network Aware Application", Work in Progress, Internet-Draft, draft-huang-alto-mowie-for-network-aware-app-00, 9 March 2020, <http://www.ietf.org/internet-drafts/draft-huang-alto-mowie-for-network-aware-app-00.txt>.

[I-D.ietf-dmm-5g-uplane-analysis]
Homma, S., Miyasaka, T., Matsushima, S., and D. Voyer, "User Plane Protocol and Architectural Analysis on 3GPP 5G System", Work in Progress, Internet-Draft, draft-ietf-dmm-5g-uplane-analysis-03, 3 November 2019, <http://www.ietf.org/internet-drafts/draft-ietf-dmm-5g-uplane-analysis-03.txt>.

[I-D.yang-alto-deliver-functions-over-networks]
Yang, S., Cui, L., Xu, M., Yang, Y., and R. Huang, "Delivering Functions over Networks: Traffic and Performance Optimization for Edge Computing using ALTO", Work in Progress, Internet-Draft, draft-yang-alto-deliver-functions-over-networks-01, 13 July 2020, <http://www.ietf.org/internet-drafts/draft-yang-alto-deliver-functions-over-networks-01.txt>.

[LHC] "CERN - LHC", 2019, <https://atlas.cern/tags/lhc>.

[SENSE] "Services - SENSE", 2019, <http://sense.es.net/services>.

[TON2019] Gao, K., Xiang, Q., Wang, X., Yang, Y.R., and J. Bi, "An objective-driven on-demand network abstraction for adaptive applications", IEEE/ACM Transactions on Networking (TON) Vol 27, no. 2 (2019): 805-818., 2019.

## Appendix A. Changes since -10

Revision -11

* replaces "part" with "components" in the abstract;

* identifies additional requirements (AR) derived from the flow scheduling example, and introduces how the extension addresses the additional requirements

* fixes the inconsistent use of "start" parameter in multipart responses;

* specifies explicitly how to handle "cost-constraints";

* uses the latest IANA registration mechanism defined in [I-D.ietf-alto-unified-props-new];

* renames "persistent-entities" to "persistent-entity-id";

* makes "application/alto-propmap+json" as the media type of defining resources for the "ane" domain;

* updates the examples;

* adds the discussion on ephemeral and persistent ANEs.

## Appendix B. Changes since -09

Revision -10

* revises the introduction which

  - extends the scope where the PV extension can be applied beyond the "path correlation" information

* brings back the capacity region use case to better illustrate the problem

* revises the overview to explain and defend the concepts and decision choices

* fixes inconsistent terms, typos

## Appendix C. Changes since -08

This revision

* fixes a few spelling errors

* emphasizes that abstract network elements can be generated on demand in both introduction and motivating use cases

## Appendix D. Changes Since Version -06

* We emphasize the importance of the path vector extension in two aspects:

  1. It expands the problem space that can be solved by ALTO, from preferences of network paths to correlations of network paths.

  2. It is motivated by new usage scenarios from both application's and network's perspectives.

* More use cases are included, in addition to the original capacity region use case.

* We add more discussions to fully explore the design space of the path vector extension and justify our design decisions, including the concept of abstract network element, cost type (reverted to -05), newer capabilities and the multipart message.

* Fix the incremental update process to be compatible with SSE -16 draft, which uses client-id instead of resource-id to demultiplex updates.

* Register an additional ANE property (i.e., persistent-entities) to cover all use cases mentioned in the draft.

Authors' Addresses

Kai Gao
China
610000
Chengdu
No.24 South Section 1, Yihuan Road
Sichuan University

Email: kaigao@scu.edu.cn

Young Lee

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
91460 Nozay
France

Email: sabine.randriamasy@nokia-bell-labs.com

Yang Richard Yang
Yale University
51 Prospect Street
New Haven,  CT
United States of America

Email: yry@cs.yale.edu

Jingxuan Jensen Zhang
China
201804
Shanghai
4800 Caoan Road
Tongji University

Email: jingxuan.n.zhang@gmail.com