ALTO
Internet-Draft
Intended status: Standards Track
Expires: 10 September 2020

K. Gao
Sichuan University
Y. Lee

S. Randriamasy
Nokia Bell Labs
Y.R. Yang
Yale University
J. Zhang
Tongji University
9 March 2020

ALTO Extension: Path Vector
draft-ietf-alto-path-vector-10

Abstract

This document is an extension to the base Application-Layer Traffic Optimization protocol [RFC7285]. The current ALTO Cost Services 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 parts or elements traversed by a path between its source and destination. Examples of such abstracted parts are networks, data centers or links. This is useful for applications whose performance is impacted by particular network parts 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 is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 10 September 2020.

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.

The base protocol [RFC7285] defines Cost Map and Endpoint Cost Service that expose the topological distances of a set of <source, destination> pairs. Various extensions have been proposed extend the capability of these services 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].

Existing ALTO services provide only cost information on an end-to-end path defined by its <source, destination> endpoints. However, the QoE of many overlay applications depends not only on the end-to-end costs, 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]. On the other hand, ISPs are not likely to expose details on their network paths: first for the sake of confidentiality, second because it may represent 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.

It may be helpful as well for ISPs if applications could avoid using bottlenecks or challenging the network with poorly scheduled traffic. 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 gives an overview of the protocol design. Section 4 and Section 5 specify the Path Vector extension to the ALTO IRD and the information resources, with some concrete examples presented in Section 6. Section 7 discusses the backward compatibility with the base protocol and existing extensions. Security and IANA considerations are discussed in Section 9 and Section 10 respectively.

2. Use Cases

2.1. Capacity Region for Multi-Flow Scheduling

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.

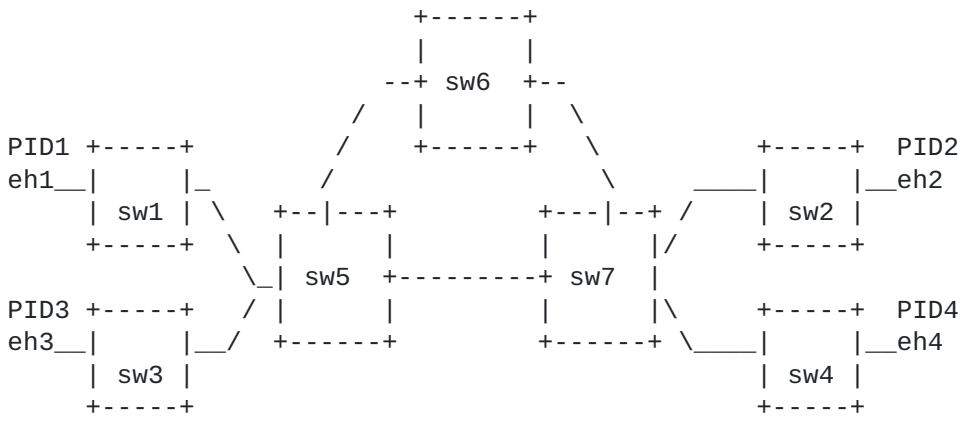

Figure 1: Raw Network Topology

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

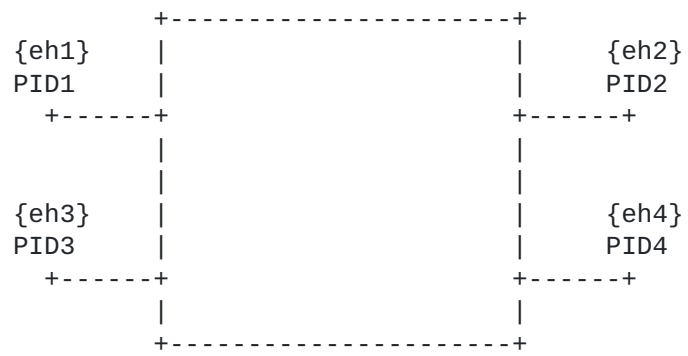

Figure 2: Base Single-Node Topology Abstraction

Consider an application overlay (e.g., a large data analysis system) which wants to schedule 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 ALTO server that the bandwidth of eh1 -> eh2 and eh1 -> eh4 are both 100 Mbps. But this information is not enough. 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:

* The network needs to expose more detailed routing information to show the shared bottlenecks.

* The network needs to provide the necessary abstraction to hide the real topology information while providing enough information to applications.

The path vector extension defined in this document propose a solution to provide these details.

2.2. Recent Use Cases

This section highlights some recent use cases that are reported in IETF and ALTO working group. See [I-D.bernstein-alto-topo] for a more comprehensive survey of use cases where extended network topology information is needed.

2.2.1. 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.

2.2.2. Context-aware Data Transfer

It is sometimes important to know how the capabilities of various network components between two end hosts. 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.

2.2.3. 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 introduce larger delay and more overhead on both the ALTO server and the ALTO client.

2.3. 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 an abstraction 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 aggregation of network components.

* 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 4.5. 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.

3. 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].

Fundamentally, this extension conveys two pieces of information:

1. The abstract network state: The abstract network state is modeled as an annotated graph, where each node is an Abstract Network Element (ANE) and each annotation is a property associated with an ANE.

2. Routing information: The routing information is modeled as an array of nodes in the annotated graph that is traversed by the path between a source and a destination.

However, it can be observed that the routing information already conveys the connectivity of the abstract network. Thus, this extensions allows an ALTO server to provide the routing information and the association between ANEs and their properties. Specifically, this document uses the following designs:

1. This extension conveys the routing information in the abstract network in an ALTO Cost Map or Endpoint Cost Map which accepts a Path Vector, i.e., a JSON array of ANEs traversed by the path between a source and a destination, as the cost value. With the Path Vectors, an ALTO client can simultaneously reconstruct the structure of the abstract network and the routing for the paths between endpoints.

2. This extension uses the ALTO Unified Property Map to convey the properties associated with the ANEs, which offers more fine-grained abstract network state for overlay applications.

3. This extension uses the multipart message TBD-ALTO-MULTIPART to include both information resources in the same Path Vector response.

3.1. Abstract Network Element

This extension introduce Abstract Network Element (ANE) as an indirect and network-agnostic way to specify an aggregation of intermediate network components which can be treated as if they are placed in the same location in the network, based on geo-location, OSPF domain, service type, algebraic properties, or other criteria.

3.1.1. ANE Name

Each ANE is uniquely identified by a string of type ANEName as specified in Section 4.1. An important observation is that for different requests, an ALTO server may selectively apply different methods to create the abstract network state based on confidentiality and performance considerations. Thus, the ANEs inside the abstract network may be constructed on demand. This indicates that the scope of an ANEName is limited to the Path Vector response.

Since each ANE is also an entity in the Unified Property Map, the ANE Name MUST conform to the encoding of an Entity Identifier. Thus, this document also specifies a new EntityDomainName following the instructions in [I-D.ietf-alto-unified-props-new].

3.1.2. ANE Properties

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] with some additional considerations.

1. As a property may not exist in every ANE, it must be interpreted in the same way by the ALTO server and the ALTO client. Thus, when an ANE property is specified, its intended semantics MUST specify how to interpret the case that a requested ANE property does not exist in an ANE.

2. As each ANE is an aggregation of multiple network components, its properties are the aggregated results of the components' properties. For different ALTO server implementations, different properties MAY have different rules when they are aggregated into a single ANE. For example, if an ANE is the aggregation of two networks where each network contains a CDN, an ALTO server may selectively expose one CDN, expose none, or expose both in the ANE, according to its own aggregation policies.

   However, it is common that an ALTO client needs to compute the aggregated property value of some ANEs, e.g., to infer the end-to-end property for a <source, destination> pair. It is RECOMMENDED that the intended semantics of an ANE property specifies how to compute the aggregated value without loss of information. Thus, the information is interpreted by the ALTO server and the ALTO client in the same way. For example, properties with algebraic properties can be aggregated following the algebraic rules [TON2019].

   NOTE: The aggregation rule ONLY specifies how to compute the aggregated property for a Path Vector, NOT how the ANEs can be aggregated in the Path Vector response. This is because the change of Path Vectors may change the routing information and the abstract network topology, leading to inaccurate results.

3. An ALTO Path Vector resource MAY only support a set of ANE properties. Meanwhile, an ALTO client MAY only require a subset of the available properties. Thus, a property negotiation process is required.

   This document uses a similar approach as the negotiation process of cost types: 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 SHOULD be specified in a new filter called "ane-property-names"

in the request body; the response MUST return and only return the selected properties for the ANEs in the response, if applicable.

3.2. Path Vector

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 ATLO 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 4.5

3.3. Multipart Path Vector Response

For a basic ALTO information resource, the 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.

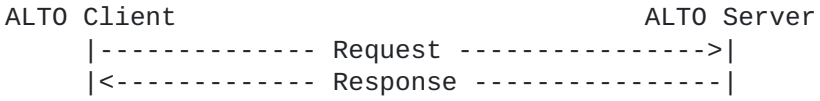

Figure 3: A Typical ALTO Request and Response

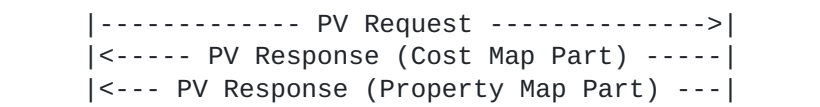

Figure 4: The Path Vector Extension Request and Response

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.

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 3.1 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 second 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 CAN 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 uses 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".

3.3.1. 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".

3.3.2. 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 4.6 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 happens inside 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.

3.3.3. Order of Part Messages

According to RFC 2387 [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.

4. Basic Data Types

4.1. ANE Name

An ANE Name is encoded as a JSON string, which has the same format as EntityIdentifer (Section 3.1.3 of [I-D.ietf-alto-unified-props-new]) and the EntityDomainName MUST be "ane", indicating that this entity belongs to the "ane" Entity Domain.

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

4.2. ANE Domain

This document specifies a new ALTO entity domain called "ane" in addition to the ones in [I-D.ietf-alto-unified-props-new]. The ANE domain associates property values with the ANEs in a network. The entity in ANE domain is often used in the Path Vector by Cost Map or Endpoint Cost Service resources. Accordingly, the ANE domain always depends on a Cost Map or an Endpoint Cost Map.

4.2.1. Entity Domain Type

ane

4.2.2. Domain-Specific Entity Identifier

The entity identifier of ANE domain uses the same encoding as ANEName (Section 4.1).

4.2.3. Hierarchy and Inheritance

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

4.3. New Resource-Specific Entity Domain Exports

4.3.1. ANE Domain of Cost Map Resource

If an ALTO Cost Map resource supports the Path Vector cost type, it can export an "ane" typed entity domain defined by the union of all sets of ANE names, where each set of ANE names are an "ane-path" metric cost value in this ALTO Cost Map resource.

4.3.2. ANE Domain of Endpoint Cost Service Resource

If an ALTO Endpoint Cost Service resource supports the Path Vector cost type, it can export an "ane" typed entity domain defined by the union of all sets of ANE names, where each set of ANE names are an "ane-path" metric cost value in this ALTO Endpoint Cost Service resource.

4.4. ANE Property Name

An ANE Property Name is encoded as an Entity Property Name (Section 3.2.2 of [I-D.ietf-alto-unified-props-new]) where

* the ResourceID part of an ANE Property Name MUST be empty;

* the EntityPropertyType part MUST be a valid property of an ANE entity, i.e., the mapping of the ANE domain type and the Entity Property Type MUST be registered to the ALTO Resource Entity Property Mapping Registries (Section 11.5 in [I-D.ietf-alto-unified-props-new]).

4.4.1. ANE Property: Maximum Reservable Bandwidth

The maximum reservable bandwidth property conveys the maximum bandwidth that can be reserved for all the traffic that traverses an ANE. The Entity Property Type of the maximum reservable bandwidth is "maxresbw", and 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 has sufficiently large bandwidth to be reserved. If the ANE does not support bandwidth reservation, the value MUST be present and be set to 0.

The aggregated value of a Path Vector is the minimum value of all the ANEs in the Path Vector.

4.4.2. ANE Property: Persistent Entities

The persistent entities property conveys the physical or logical network entities (e.g., links, in-network caching service) that are contained by an ANE. It is indicated by the property name "persistent-entities". The value is encoded as a JSON array of entity identifiers ([I-D.ietf-alto-unified-props-new]). These entity identifiers are persistent so that a client CAN further query their properties for future use.

If this property is requested but is missing for a given ANE, it MUST be interpreted as that no such entities exist in this ANE.

4.5. 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.

4.5.1. Cost Metric: ane-path

This cost metric conveys an array of ANE names, where each ANE name uniquely represents an ANE traversed by traffic from a source to a destination.

4.5.2. Cost Mode: array

This cost mode 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 4.1) when the cost metric is "ane-path".

4.6. Part Resource ID

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

WARNING: 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 3.3.2) 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.

5. Service Extensions

5.1. 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.

5.1.1. Media Type

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

5.1.2. HTTP Method

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

5.1.3. 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.

5.1.4. 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.

5.1.5. Uses

The resource ID of the network map based on which the PIDs in the returned cost map will be defined. If this resource supports "persistent-entities", it MUST also include ALL the resources that exposes the entities that MAY appear in the response.

5.1.6. Response

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

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 MUST be a quoted string where the quoted

part has the same value as the "Resource-ID" header in the first part.

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

The body of the response consists of two parts.

The first 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 first 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 3.3.2. 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 second 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 second 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 first part MUST be included in the "dependent-vtags". If "persistent-entities" 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 first part, where the EntityProps has one member for each property requested by the client if applicable.

5.2. 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.

5.2.1. Media Type

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

5.2.2. HTTP Method

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

5.2.3. 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 5.1.3.

5.2.4. 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 5.1.4.

5.2.5. Uses

If a multipart endpoint cost resource supports "persistent-entities", the "uses" field in its IRD entry MUST include ALL the resources which exposes the entities that MAY appear in the response.

5.2.6. 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 MUST be a quoted string where the quoted part has the same value as the "Resource-ID" header in the first part.

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

The body consists of two parts:

The first 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 first 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 3.3.2.

The second 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 second 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 first part MUST be included in the "dependent-vtags". If "persistent-entities" 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 first part, where the EntityProps has one member for each property requested by the client if applicable.

6. Examples

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

6.1. Example: Information Resource Directory

Below is an example of an Information Resource Directory which enables the path vector extension. Some critical modifications include:

* The "path-vector" cost type (Section 4.5) is defined in the "cost-types" of the "meta" field.

* The "cost-map-pv" information resource provides a multipart filtered cost map resource, which exposes the Maximum Reservable Bandwidth ("maxresbw") property.

* The "http-proxy-props" information resource provides a filtered unified property map resource, which exposes the HTTP proxy entity domain (encoded as "http-proxy") and the "price" property. Note that HTTP proxy is NOT a valid entity domain yet and is used here only for demonstration.

* The "endpoint-cost-pv" information resource provides a multipart endpoint cost resource. It exposes the Maximum Reservable Bandwidth ("maxresbw") property and the Persistent Entity property ("persistent-entities"). The persistent entities MAY come from the "http-proxy-props" resource.

* The "update-pv" information resource provides the incremental update ([I-D.ietf-alto-incr-update-sse]) service for the "endpoint-cost-pv" resource.

```
{
  "meta": {
    "cost-types": {
      "path-vector": {
        "cost-mode": "array",
        "cost-metric": "ane-path"
      }
    }
  },
  "resources": {
    "my-default-networkmap": {
      "uri" : "http://alto.example.com/networkmap",
      "media-type" : "application/alto-networkmap+json"
    },
    "cost-map-pv": {
      "uri": "http://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": [ "maxresbw" ]
        },
        "uses": [ "my-default-networkmap" ]
      },
      "http-proxy-props": {
        "uri": "http://alto.example.com/proxy-props",
        "media-type": "application/alto-propmap+json",
        "accepts": "application/alto-propmapparams+json",
        "capabilities": {
          "mappings": {
            "http-proxy": [ "price" ]
          }
        }
      },
      "endpoint-cost-pv": {
        "uri": "http://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": [ "maxresbw", "persistent-entities" ]
        },
        "uses": [ "http-proxy-props" ]
      },
      "update-pv": {
        "uri": "http://alto.example.com/updates/pv",
        "media-type": "text/event-stream",
        "uses": [ "endpoint-cost-pv" ],
        "accepts": "application/alto-updatestreamparams+json",
        "capabilities": {
          "support-stream-control": true
        }
      }
    }
  }
```

6.2. Example: Multipart Filtered Cost Map

The following examples demonstrate the request to the "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 three ANEs, where "ane:L001" is shared by traffic from "PID1" to both "PID2" and "PID3".

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": [ "PID2", "PID3" ]
  }
}

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": "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": [ "ane:L001", "ane:L003" ],
        "PID3": [ "ane:L001", "ane:L004" ]
      }
    }
  }
  --example-1
  Resource-Id: propmap
  Content-Type: application/alto-propmap+json

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

6.3. 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.

The second part returns the requested properties of ANEs in the first part. The "ane:NET001" element contains an HTTP proxy entity, which can be further used by the client. Since it does not contain a "maxresbw" property, the client SHOULD assume it does NOT support bandwidth reservation but will NOT become a traffic bottleneck, as specified in Section 4.4.1.

```
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": [ "ipv4:192.0.2.2" ],
    "dsts": [ "ipv4:192.0.2.89",
              "ipv4:203.0.113.45",
              "ipv6:2001:db8::10" ]
  },
  "ane-property-names": [ "maxresbw", "persistent-entities" ]
}

HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: multipart/related; boundary=example-2;
              type=application/alto-endpointcost+json

--example-2
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": {
    "ipv4:192.0.2.2": {
      "ipv4:192.0.2.89":   [ "ane:NET001", "ane:L002" ],
      "ipv4:203.0.113.45": [ "ane:NET001", "ane:L003" ]
    }
  }
```

```
}
--example-2
Resource-Id: propmap
Content-Type: application/alto-propmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "endpoint-cost-pv.ecs",
        "tag": "bb6bb72eafe8f9bdc4f335c7ed3b10822a391cef"
      },
      {
        "resource-id": "http-proxy-props",
        "tag": "bf3c8c1819d2421c9a95a9d02af557a3"
      }
    ]
  },
  "property-map": {
    "ane:NET001": {
      "persistent-entities": [ "http-proxy:192.0.2.1" ]
    },
    "ane:L002": { "maxresbw": 48000000 },
    "ane:L003": { "maxresbw": 35000000 }
  }
}
```

6.4. 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": "http://alto.example.com/updates/streams/1414"}

event: multipart/related;boundary=example-3;
       type=application/alto-endpointcost+json,ecspvsub1
data: --example-3
data: Resource-ID: ecsmap
data: Content-Type: application/alto-endpointcost+json
data:
data: <endpoint-cost-map-entry>
data: --example-3
data: Resource-ID: propmap
data: Content-Type: application/alto-propmap+json
data:
data: <property-map-entry>
data: --example-3--
```

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>
```

7. Compatibility

7.1. 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.

7.2. Compatibility with Multi-Cost Extension

This document does not specify how to integrate the "path-vector" cost mode with the multi-cost extension [RFC8189]. Although there is no reason why somebody has to compound the path vectors with other cost types in a single query, there is no compatible issue doing it without constraint tests.

7.3. 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.

7.4. 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.)

8. General Discussions

8.1. 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 defined syntax is too simple and can only be used to test the scalar cost value. For more complex cost types, like the "array" mode defined in this document, it does not work well. It will be helpful to propose more general constraint tests to better perform the query.

In practice, it is too complex to customize a language for the general-purpose boolean tests, and can be a duplicated work. So it may be a good idea to integrate some already defined and widely used query languages (or their subset) to solve this problem. The candidates can be XQuery and JSONiq.

8.2. 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.

9. 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 constraints on 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.

10. IANA Considerations

10.1. ALTO Cost Mode Registry

This document specifies a new cost mode "path-vector". However, the base ALTO protocol does not have a Cost Mode Registry where new cost mode can be registered. This new cost mode will be registered once the registry is defined either in a revised version of [RFC7285] or in another future extension.

10.2. ALTO Entity Domain Registry

This document registers a new entry to the ALTO Domain Entity Registry, as instructed by Section 9.2 of [I-D.ietf-alto-unified-props-new]. See below in Table 1.

Identifier	Entity Address Encoding	Hierarchy & Inheritance
ane	See Section 4.2.2	None

Table 1: ALTO Entity Domain

10.3. ALTO Entity Property Type Registry

Two initial entries are registered to the ALTO Domain "ane" in the "ALTO Entity Property Type Registry". See below in Table 2.

Identifier	Intended Semantics
ane:maxresbw	See Section 4.4.1
ane:persistent-entities	See Section 4.4.2

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

10.4. ALTO Resource Entity Domain Export Registries

10.4.1. costmap

Entity Domain Type	Export Function
ane	See Section 4.3.1

Table 3: ALTO Cost Map Entity Domain Export

10.4.2. endpointcost

Entity Domain Type	Export Function
ane	See Section 4.3.2

Table 4: ALTO Endpoint Cost Entity Domain Export

11. 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.

12. References

12.1. 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-19, 2 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-alto-cost-calendar-19.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-20, 20 February 2020, <http://www.ietf.org/internet-drafts/draft-ietf-alto-incr-update-sse-20.txt>.

[I-D.ietf-alto-performance-metrics]
WU, Q., Yang, Y., Lee, Y., Dhody, D., and S. Randriamasy, "ALTO Performance Cost Metrics", Work in Progress, Internet-Draft, draft-ietf-alto-performance-metrics-08, 4 November 2019, <http://www.ietf.org/internet-drafts/draft-ietf-alto-performance-metrics-08.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-10, 4 November 2019, <http://www.ietf.org/internet-drafts/draft-ietf-alto-unified-props-new-10.txt>.

[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>.

[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>.

12.2. 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]
"TBD", 2020.

[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., Shen, H., and L. Chen, "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-00, 29 November 2019, <http://www.ietf.org/internet-drafts/draft-yang-alto-deliver-functions-over-networks-00.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 -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 B. 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