ALTO WG   K. Gao
Internet-Draft   Sichuan University
Intended status: Standards Track   Y. Lee
Expires: May 6, 2020   Huawei
                                                         S. Randriamasy
                                                        Nokia Bell Labs
                                                                Y. Yang
                                                        Yale University
                                                               J. Zhang
                                                      Tongji University
                                                       November 3, 2019

ALTO Extension: Path Vector
draft-ietf-alto-path-vector-09

Abstract

   This document defines an ALTO extension that allows an ALTO
   information resource to provide not only preferences but also
   correlations of the paths between different PIDs or endpoints. The
   extended information, including aggregations of network components on
   the paths and their properties, can be used to improve the robustness
   and performance for applications in some new usage scenarios, such as
   high-speed data transfers and traffic optimization using in-network
   storage and computation.

   This document reuses the mechanisms of the ALTO base protocol and the
   Unified Property extension, such as Information Resource Directory
   (IRD) capabilities and entity domains, to negotiate and exchange path
   correlation information. Meanwhile, it uses an extended compound
   message to fully represent the path correlation information, for
   better server scalability and message modularity. Specifically, the
   extension 1) introduces abstract network element (ANE) as an
   abstraction for an aggregation of network components and encodes a
   network path as a "path vector", i.e., an array of ANEs traversed
   from the source to the destination, 2) encodes properties of abstract
   network elements in a unified property map, and 3) encapsulates the
   two types of information in a multipart message.

Requirements Language

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   The ALTO protocol is aimed to provide applications with knowledge of
   the underlying network topologies from the point of views of ISPs.
   The base protocol [RFC7285] defines cost maps and endpoint cost
   services that expose the preferences of network paths for a set of
   source and destination pairs.

   While the preferences of network paths are already sufficient for a
   wide range of applications, new traffic patterns and new network
   technologies are emerging that are well beyond the domain for which
   existing ALTO maps are engineered. This trend includes but is not
   limited to:

   Very-high-speed data transfers: Applications, such as Content
      Distribution Network (CDN) overlays, geo-distributed data centers
      and large-scale data analytics, are foundations of many Internet
      services today and have very large traffic between a source and a
      destination. Thus, the interference between traffic of different
      source and destination pairs cannot be omitted, which cannot be
      provided by or inferred from existing ALTO base protocol and
      extensions.

   In-network storage and computation: Emerging networking technologies
      such as network function virtualization and mobile edge computing
      provide storage and computation inside the network. Applications
      can leverage these resources to further improve their performance,
      for example, using in-network caching to reduce latency and
      bandwidth from a given source to multiple clients. However,
      existing ALTO extensions provide no map resources to discover
      available in-network services, nor any information to help ALTO
      clients determine how to effectively and efficiently use these
      services.

   This document specifies a new extension to incorporate these newly
   emerged scenarios into the ALTO framework. The essence of this
   extension is that an ALTO server exposes correlations of network
   paths in addition to preferences of network paths.

The correlations of network paths are represented by path vectors.
Each element in a path vector, which is referred to as an abstract
network element (ANE), is the aggregation of network components on
the path, such as routers, switches, links and clusters of in-network
servers. If an abstract network element appears in multiple network
paths, the traffic along these paths will join at this abstract
network element and are subject to the corresponding resource
constraints.

The availability of the path correlations by itself can help ALTO
clients conduct better traffic scheduling. For example, an ALTO
client can use the path correlations to conduct more intelligent end-
to-end measurement and identify traffic bottlenecks.

By augmenting these abstract network elements with different
properties, an ALTO server can provide a more fine-grained view of
the network. ALTO clients can use this view to derive information
such as shared risk resource groups, capacity regions and available
in-network cache locations, which can be used to improve the
robustness and performance of the application traffic.

Given specific properties, an ALTO server may construct the abstract
network elements on demand. For example, as shown in Section 4.2,
when an ALTO client only demands the capacity region, an ALTO server
can identify and construct an abstract network element for each
bottleneck link for the specific query. Thus, an ALTO server can
minimize the information exposed to a client.

2. Changes since -08

This revision

o fixes a few spelling errors

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

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 ones defined in these documents, this document also
uses the following additional terms:

o Abstract network element (ANE): An abstract network element is an
   abstraction of network components. It can be a link, a middlebox,
   a virtualized network function (VNF), etc., or their aggregations.
   An abstract network element can be constructed either statically

in advance or on demand based on the requested information. In a
response, each abstract network element is represented by a unique
ANE identifier. Note that an ALTO client MUST NOT assume ANEs in
different responses but with the same identifier refer to the same
aggregation of network components.

o Path vector: A path vector is an array of ANE identifiers. It
   presents an abstract network path between source/destination
   points such as PIDs or endpoints.

o Path vector resource: A path vector resource refers to an ALTO
   resource which supports the extension defined in this document.

o Path vector query/request: A path vector query or request refers
   to the POST message sent to an ALTO path vector resource,
   requesting for path vectors of different source/destination pairs
   and associated properties.

o Path vector response: A path vector response refers to the
   multipart/related message returned by a path vector resource. It
   consists of a path vector part, i.e., the (endpoint) cost map part
   which contains the path vector information, and a property map
   part.

4. Use Cases

This section describes typical use cases of the path vector
extension. For each example, we demonstrate that it is beyond the
capabilities of the current ALTO framework and thus provides a new
usage scenario. In this first two example, we also demonstrate the
benefits of constructing abstract network elements on demand.

4.1. Shared Risk Resource Group

Consider an application which controls 4 end hosts (eh1, eh2, eh3 and
eh4), which are connected by an ISP network with 5 switches (sw1,
sw2, sw3, sw4 and sw5) and 5 links (l1, l2, l3, l4 and l5), as shown
in Figure 1. Assume the end hosts are running data storage services
and some analytics tasks, which requires high data availability. In
order to determine the replica placement, the application must know
how the end hosts will be partitioned if certain network failures
happen.

For that purpose, the application uses an ALTO client, which
communicates with an ALTO server provided by the ISP network. Since
the Endpoint Cost Service with only scalar cost values cannot provide
essential information for the application, thus, both the client and
the server have the path vector extension enabled.

Assume the ISP uses shortest path routing. For simplicity, consider
the data availability on eh4. The network components on the paths
from all other end hosts to eh4 are as follows:

eh1->eh4: sw1, l1, sw3, l4, sw5
eh2->eh4: sw2, l2, sw3, l4, sw5
eh3->eh4: sw4, l5, sw5

While an ALTO server can simply return the information above to the
client, it can benefit from on-demand aggregation of network
components.

```
                        +-----------------+
          ------------->|                 |<---------
         /   ---------->|   ALTO Client   |<------    \
        /   /           +-----------------+      \    \
       |   |                     ^                 |    |
       |   |                     |                 |    |
       |   |                     v                 |    |
       |   |           +-----------------+         |    |
       |   |  ..........|                 |......   |    |
       |   |  .         |   ALTO Server   |    .    |    |
       |   |  .         +-----------------+    .    |    |
       |   |  .                                .    |    |
       |   v  . +-----+              +-----+ .  v    |
       |  eh1 --|     |-      l3.   -|     |-- eh3   |
       |      . | sw1 | \..l1      ../ | sw4 | .     |
       |      . +-----+  \  +-----+  /  +-----+ .    |
       |      .          --|     |--       |    .    |
       |      .            | sw3 |    l5..|      .    |
       |      .          --|     |--       |    .    |
       |      . +-----+  /  +-----+  \  +-----+ .    |
       |      . |     | /..l2    l4..\ |     | .     |
        -->eh2 --| sw2 |-              -| sw5 |-- eh4<--
              . +-----+              +-----+ .
              ....................................
```
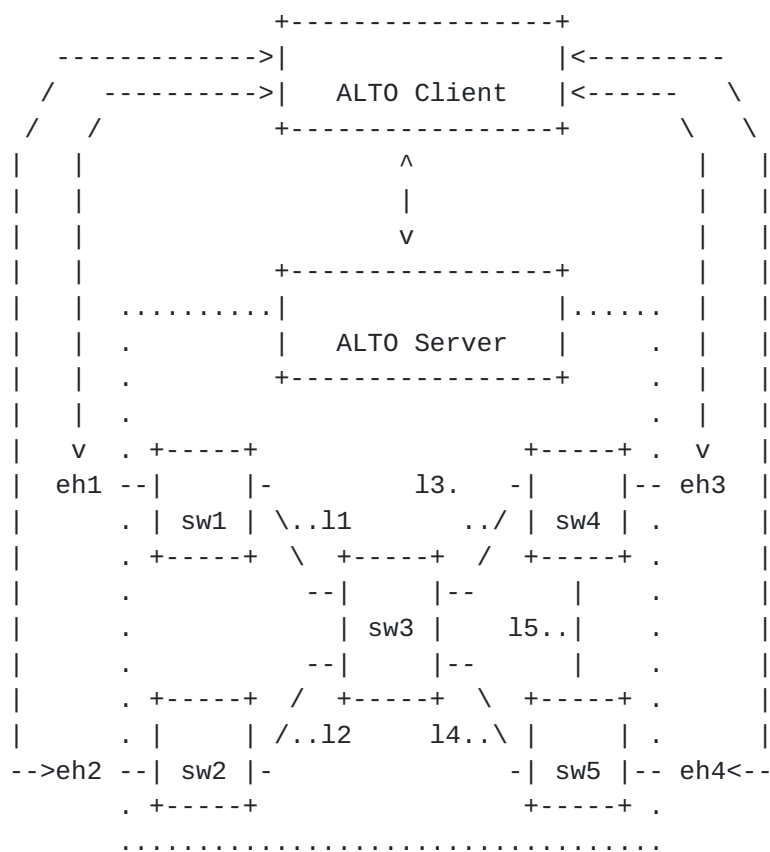
Figure 1: Topology for the Shared Risk Resource Group and the
Capacity Region Use Cases

These network components can be categorized into 5 categories:

1. Failure will only disconnect eh1 to eh4: sw1, l1.

2. Failure will only disconnect eh2 to eh4: sw2, l2.

3. Failure will only disconnect eh3 to eh4: sw4, l5.

4. Failure will only disconnect eh1 and eh2 to eh4: sw3, l4.

5. Failure will disconnect eh1, eh2 and eh3 to eh4: sw5.

The ALTO server can then aggregate sw1 and l1 as an abstract network
element, ane1. By applying the aggregation to the categories, the
response may be as follows:

```
eh1->eh4: ane1, ane4, ane5
eh2->eh4: ane2, ane4, ane5
eh3->eh4: ane3, ane5
```

Thus, the application can still derive the potential network
partitions for all possible network failures without knowing the
exact network topology, which protects the privacy of the ISP. Note
this aggregation is specific to the query, i.e., the response is
constructed on demand. If we change a source or a destination in the
query, for example exchange the role of eh3 and eh4, we get the same
failure categories but each category has a different set of links and
switches.

4.2. Capacity Region

This use case uses the same topology and application settings as in
Section 4.1 as shown in Figure 1. Assume the capacity of each link
is 10 Gbps, except l5 whose capacity is 5 Gbps. Assume the
application is running a map-reduce task, where the optimal traffic
scheduling is usually referred to the co-flow scheduling problem.
Consider a simplified co-flow scheduling problem, e.g., the first
stage of a map-reduce task which needs to transfer data from two data
nodes (eh1 and eh3) to the mappers (eh2 and eh4). In order to
optimize the job completion time, the application needs to determine
the bottleneck of the transfers.

If the ALTO server encodes the routing cost as bandwidth of the path,
the client will obtain the following information:

```
eh1->eh2: 10 Gbps,
eh1->eh4: 10 Gbps,
eh3->eh2: 10 Gbps,
eh3->eh4:  5 Gbps.
```

However, it does not provide sufficient information to determine the
bottleneck. With the path vector extension, the ALTO server will
first return the correlations of network paths between eh1, eh3 and
eh2, eh4, as follows:

```
   eh1->eh2: ane1 (l1), ane2 (l2),
   eh1->eh4: ane1 (l1), ane4 (l4),
   eh3->eh2: ane3 (l3), ane2 (l2),
   eh3->eh3: ane5 (l5).
```

Meanwhile, the ALTO server also returns the capacity of each ANE:

```
   ane1.capacity = 10 Gbps,
   ane2.capacity = 10 Gbps,
   ane3.capacity = 10 Gbps,
   ane4.capacity = 10 Gbps,
   ane5.capacity =  5 Gbps.
```

With the correlation of network paths and the link capacity property, the client is able to derive the capacity region of data transfer rates. Let $x1$ denote the transfer rate of eh1->eh2, $x2$ denote the rate of eh1->eh4, $x3$ denote the rate of eh3->eh2, and $x4$ denote the rate of eh3->eh4. The application can derive the following information from the responses:

```
        eh1->eh2  eh1->eh4  eh3->eh2  eh3->eh4        capaity
ane1     1         1         0         0        |    10 Gbps
ane2     1         0         1         0        |    10 Gbps
ane3     0         0         1         0        |    10 Gbps
ane4     0         1         0         0        |    10 Gbps
ane5     0         0         0         1        |     5 Gbps
```

Specifically, the coefficient matrix on the left hand side is the transposition of the matrix directly derived from the path vector part, and the right-hand-side vector is directly derived from the property map part. Thus, the bandwidth constraints of the data transfers are as follows:

```
x1 + x2 <= 10 Gbps (ane1),
x1 + x3 <= 10 Gbps (ane2),
x3      <= 10 Gbps (ane3),
x2      <= 10 Gbps (ane4),
x4      <=  5 Gbps (ane5).
```

Now we demonstrate how the property can lead to better on-demand aggregation. For the capacity region use case, we can easily conclude that each abstract network element refers to a linear constraint. For the example, we can see that the constraints of ane3 and ane4 are redundant, i.e., they can be removed without affecting the final capacity region. Thus, an ALTO server can return the following information:

```
   eh1->eh2: ane1 (l1), ane2 (l2),
   eh1->eh4: ane1 (l1),
   eh3->eh2: ane2 (l2),
   eh3->eh3: ane5 (l5),
```

   and

```
   ane1.capacity = 10 Gbps,
   ane2.capacity = 10 Gbps,
   ane5.capacity =  5 Gbps.
```

4.3. In-Network Caching

   Consider an application which controls 3 end hosts (eh1, eh2 and
   eh3), which are connected by an ISP network and the Internet, as
   shown in Figure 2. Assume two clients at end hosts eh2 and eh3 are
   downloading the same data from a data server at eh1. Meanwhile, the
   network provider offers an in-network caching service at the gateway.

```
                    +-------------+
            ------->|             |<----------------------
           /  ----->| ALTO Client |<-------              \
          /  /      +-------------+      |                 \
         /  /                            v                  |
        /  /                     +-------------+            |
       /  /    ......................| ALTO Server |......   |
      /  /     .                  +-------------+    .   |
     /  /      .                 +---------+          .   |
    |  |       .                -+ Caching |          .   |
    |  |       .               / | Proxy   |          .   |
    |  |S      .+-------+      /  +---------+          .   |
    |  -->eh1--| sub   |_     |                        .   |
    |         .| net 1 | \   +------+         +----------+.  |
    |         .+-------+  ---|      |         |          |.  v C2
    |         .             | Gate +---------+ Internet |--eh3
    |   C1    .+-------+   --| way  |         |          |.
    ----->eh2--| sub   |__/ +------+         +----------+.
              .| net 2 |                              .
              .+-------+                              .
              ...........................................
```

            Figure 2: Topology for the In-Network Caching Use Case.

   With the path vector extension enabled, the ALTO server can expose
   two types of information

   Without the traffic correlation information, the ALTO client cannot
   know whether or how the traffic goes through the proxy. For example,

    if subnet1 and subnet2 are directly connected and the traffic from
    eh1 to eh2 bypasses the gateway, the in-network cache can only be
    used for traffic from C2 to S and is less effective.

5. Overview

    This section gives a top-down overview of approaches adopted by the
    path vector extension, with discussions to fully explore the design
    space. 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].

5.1. Workflow

    The workflow of the base ALTO protocol consists of one round of
    communication: An ALTO client sends a request to an ALTO server, and
    the ALTO server returns a response, as shown in Figure 3. Each
    response contains only one type of ALTO resources, e.g., network
    maps, cost maps, or property maps.

```
        +-------------+                        +-------------+
        | ALTO Client |                        | ALTO Server |
        +-------------+                        +-------------+
              |                 Request                |
              |--------------------------------------->|
              |                                        |
              |                 Response               |
              |<---------------------------------------|
              |                                        |
              .                    .                   .
              .                    .                   .
              .                    .                   .
              |               PV Request               |
              |--------------------------------------->|
              |                                        |
              |      PV Response (Cost Map Part)        |
              |<---------------------------------------|
              |                                        |
              |     PV Response (Property Map Part)     |
              |<---------------------------------------|
              |                                        |
```

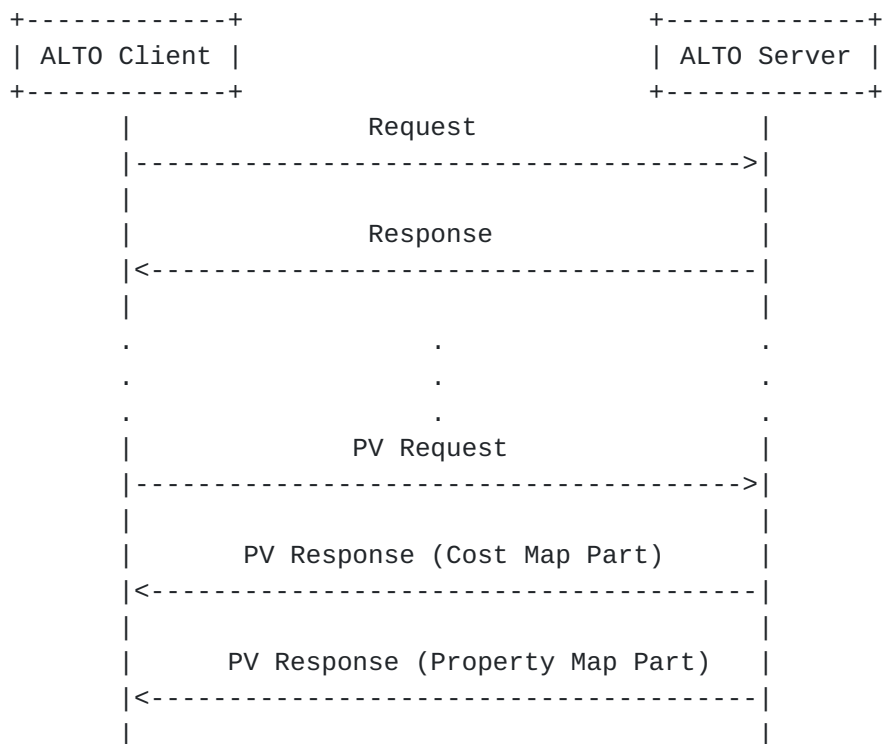   Figure 3: Information Exchange Process of the base ALTO Protocol and
                         the Path Vector Extension

    The path vector extension, on the other hand, involves two types of
    information resources. First, path vectors, which represent the
    correlations of network paths for all <source, destination> pairs in

the request, are encoded as an (endpoint) cost map with an extended
cost type. Second, properties associated with the ANEs are encoded
as a property map.

Instead of making two consecutive queries, however, the path vector
extension adopts a workflow which also consists of only one round of
communication, based on the following reasons:

1. ANE Computation Flexibility. For better scalability, flexibility
    and privacy, Abstract Network Elements MAY be constructed on
    demand, and potentially based on the properties (See Section 5.2
    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. Server Scalability. As ANEs are constructed on demand, mappings
    of each ANE to its underlying network devices and resources CAN
    be different in different queries. 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.

Thus, the path vector extension encapsulates all essential
information in one request, and returns both path vectors and
properties associated with the ANEs in a single response. See
Section 5.3 for more details.

5.2. Abstract Network Element

A key design in the path vector extension is abstract network
element. Abstract network elements can be statically generated, for
example, based on geo-locations, OSPF areas, or simply the raw
network topology. They CAN also be generated on demand based on a
client's request. This on-demand ANE generation allows for better
scalability, flexibility and privacy enhancement.

Consider an extreme case where the client only queries the bandwidth
between one source and one destination in the topology shown in
Figure 4. Without knowing in advance the desired property, an ALTO
server MAY need to include all network components on the paths for
high accuracy. However, with the prior knowledge that the client
only asks for the bandwidth information, an ALTO server CAN either 1)
selectively pick the link with the smallest available bandwidth, or
2) dynamically generate a new ANE whose available bandwidth is the
smallest value of the links' on the path. Thus, an ALTO server can

provide accurate information with very little leak of its internal
network topology. For more general cases, ANEs MAY be constructed
based on algebraic aggregations, please see [TON2019] for more
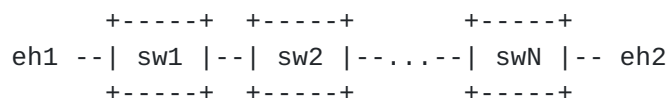details.

```
         +-----+  +-----+         +-----+
   eh1 --| sw1 |--| sw2 |--...--| swN |-- eh2
         +-----+  +-----+         +-----+
```

Figure 4: Topology for Dynamic ANE Example.

An ANE is uniquely identified by an ANE identifier (see Section 6.1)
in the same response. However, since ANEs CAN be generated
dynamically, an ALTO client MUST NOT assume that ANEs with the same
identifier but from different queries refer to the same aggregation
of network components. This approach simplifies the management of
ANE identifiers at ALTO servers, and increases the difficulty to
infer the real network topology with cross queries. It is
RECOMMENDED that the identifiers of statically generated ANEs be
anonymized in the path vector response, for example, by shuffling the
ANEs and shrinking their identifier space to [1, N], where N is the
number of ANEs etc.

5.3. Protocol Extensions

Section 5.1 has well articulated the reasons to complete the
information exchange in a single round of communication. This
section introduces the three major extended components to the base
ALTO protocol and the Unified Property Map extension, as shown in
Table 1.

```
   +------------------------+-------+----------+-----------+
   | Component              | IRD   | Request  | Response  |
   +------------------------+-------+----------+-----------+
   | Path Vector Cost Type  | Yes   | Yes      | Yes       |
   | Property Negotiation   | Yes   | Yes      | Yes       |
   | Multipart Message      | Yes   | No       | Yes       |
   +------------------------+-------+----------+-----------+
```

Table 1: Extended Components and Where They Apply.

5.3.1. Path Vector Cost Type

Existing cost modes defined in [RFC7285] allow only scalar cost
values. However, the path vector extension MUST convey vector format
information. To fulfill this requirement, this document defines a
new cost mode named "array", which indicates that the cost value MUST
be interpreted as an array of JSONValue. This document also

introduces a new cost metric "ane-path" to convey an array of ANE identifiers.

The combination of the "array" cost mode and the "ane-path" cost metric also complies best with the ALTO base protocol, where cost mode specifies the interpretation of a cost value, and cost metric conveys the meaning.

5.3.2. Property Negotiation

Similar to cost types, an ALTO server MAY only support a given set of ANE properties in a path vector information resource. 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 and more specifically, in a new capability called "ane-properties"; the selected properties SHOULD be specified in a new filter called "ane-properties" in the request body; the response MUST return and only return the selected properties for the ANEs in the response, if applicable.

5.3.3. Multipart/Related Message

Path vectors and the property map containing the ANEs are two different types of objects, but they need to be encoded in one message. One approach 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".

5.3.3.1. 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".

5.3.3.2. 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 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 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.

5.3.3.3. **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.

6. **Basic Data Types**

6.1. **ANE Identifier**

   An ANE identifier is encoded as a JSON string. The string MUST be no
   more than 64 characters, and it MUST NOT contain characters other
   than US-ASCII alphanumeric characters (U+0030-U+0039, U+0041-U+005A,
   and U+0061-U+007A), the hyphen ("-", U+002D), the colon (":",
   U+003A), the at sign ("@", code point U+0040), the low line ("_",
   U+005F), or the "." separator (U+002E). The "." separator is
   reserved for future use and MUST NOT be used unless specifically
   indicated in this document, or an extension document.

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

6.2. **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.

6.2.1. **Cost Metric: ane-path**

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

6.2.2. **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 ANEIdentifier (Section 6.1)
when the cost metric is "ane-path".

6.3. 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 maps or
endpoint cost resources. Accordingly, the ANE domain always depends
on a cost map or an endpoint cost map.

6.3.1. Entity Domain Type

ane

6.3.2. Domain-Specific Entity Identifier

The entity identifier of ANE domain uses the same encoding as
ANEIdentifier (Section 6.1).

6.3.3. Hierarchy and Inheritance

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

6.4. New Resource-Specific Entity Domain Exports

6.4.1. ANE Domain of Cost Map Resource

If an ALTO cost map resource supports "ane-path" cost metric, 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.

6.4.2. ANE Domain of Endpoint Cost Resource

If an ALTO endpoint cost resource supports "ane-path" cost metric, 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 resource.

6.5. ANE Properties

6.5.1. ANE Property: Maximum Reservable Bandwidth

The maximum reservable bandwidth property conveys the maximum
bandwidth that can be reserved for traffic from a source to a
destination and is indicated by the property name "maxresbw". The
value MUST be encoded as a 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 is missing for a given ANE, it MUST
be interpreted as that the ANE does not support bandwidth reservation
but have sufficiently large bandwidth for all traffic that traverses
it.

6.5.2. ANE Property: Persistent Entity

The persistent entity property conveys the physical or logical
network entities (e.g., links, in-network caching service) that are
contained by an abstract network element. It is indicated by the
property name "persistent-entity". 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.

6.6. 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 5.3.3.2) 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.

7. 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 {
  [PropertyName ane-properties<0..*>;]
} PVReqFilteredCostMap : ReqFilteredCostMap;
```

   with fields:

   ane-properties: 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-properties"
      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.

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 {
  [PropertyName ane-properties<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.

ane-properties: 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

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.

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 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 5.3.3.2. 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 ANE identifier that appears in the first part, where the EntityProps has one member for each property requested by the client if applicable.

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 {
  [PropertyName ane-properties<0..*>;]
} PVReqEndpointcost : ReqEndpointcost;
```

with fields:

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

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 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.

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 [RFC2387] 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 5.3.3.2.

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 ANE identifier that appears in the first part, where the EntityProps has one member for each property requested by the client if applicable.

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

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

o The "path-vector" cost type (Section 6.2) is defined in the "cost-types" of the "meta" field.

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

o 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.

o 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.

   o 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-properties": [ "maxresbw" ]
          },
          "uses": [ "my-default-networkmap" ]
        },
        "http-proxy-props": {
          "uri": "http://alto.example.com/proxy-props",
          "media-type": "application/alto-propmap+json",
          "accpets": "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-properties": [ "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
        }
      }
    }
  }
```

8.2. 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-properties" 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 ANE identifiers 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": {
    }
  }
```

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 ANE identifiers 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 6.5.1.

```
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-properties": [ "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 }
    }
  }
```

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": "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>
```

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 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.

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 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.

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 network internal 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 guarantees 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.

12. IANA Considerations

12.1. 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.

12.2. ALTO Entity Domain Registry

As proposed in Section 9.2 of [I-D.ietf-alto-unified-props-new],
"ALTO Domain Entity Registry" is requested. Besides, a new domain is
to be registered, listed in Table 2.

```
+-------------+-------------------------+-------------------------+
| Identifier  | Entity Address Encoding | Hierarchy & Inheritance |
+-------------+-------------------------+-------------------------+
| ane         | See Section 6.3.2       | None                    |
+-------------+-------------------------+-------------------------+
```

Table 2: ALTO Entity Domain

12.3. ALTO Entity Property Type Registry

The "ALTO Entity Property Type Registry" is required by the ALTO
Domain "ane", listed in Table 3.

```
+-------------------------+-----------------------------------------+
| Identifier              | Intended Semantics                      |
+-------------------------+-----------------------------------------+
| ane:maxresbw            | The maximum reservable bandwidth for    |
|                         | the ANE                                 |
| ane:persistent-entities | An array of identifiers of persistent   |
|                         | entities that reside in an ANE          |
+-------------------------+-----------------------------------------+
```

Table 3: ALTO Entity Property Types

12.4. ALTO Resource Entity Domain Export Registries

12.4.1. costmap

```
+--------------------+-------------------+
| Entity Domain Type | Export Function   |
+--------------------+-------------------+
| ane                | See Section 6.4.1 |
+--------------------+-------------------+
```

Table 4: ALTO Cost Map Entity Domain Export.

12.4.2. endpointcost

```
+--------------------+-------------------+
| Entity Domain Type | Export Function   |
+--------------------+-------------------+
| ane                | See Section 6.4.2 |
+--------------------+-------------------+
```

Table 5: ALTO Endpoint Cost Entity Domain Export.

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

   [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>.

   [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

   [I-D.bernstein-alto-topo]
              Bernstein, G., Yang, Y., and Y. Lee, "ALTO Topology
              Service: Uses Cases, Requirements, and Framework", draft-
              bernstein-alto-topo-00 (work in progress), October 2013.

   [I-D.ietf-alto-cost-calendar]
              Randriamasy, S., Yang, Y., Wu, Q., Lingli, D., and N.
              Schwan, "ALTO Cost Calendar", draft-ietf-alto-cost-
              calendar-01 (work in progress), February 2017.

   [I-D.ietf-alto-incr-update-sse]
              Roome, W. and Y. Yang, "ALTO Incremental Updates Using
              Server-Sent Events (SSE)", draft-ietf-alto-incr-update-
              sse-16 (work in progress), March 2019.

   [I-D.ietf-alto-performance-metrics]
              Wu, Q., Yang, Y., Lee, Y., Dhody, D., and S. Randriamasy,
              "ALTO Performance Cost Metrics", draft-ietf-alto-
              performance-metrics-06 (work in progress), November 2018.

   [I-D.ietf-alto-unified-props-new]
              Roome, W., Randriamasy, S., Yang, Y., and J. Zhang,
              "Unified Properties for the ALTO Protocol", draft-ietf-
              alto-unified-props-new-07 (work in progress), March 2019.

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

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

Authors' Addresses

   Kai Gao
   Sichuan University
   Chengdu   610000
   China

   Email: kaigao@scu.edu.cn

   Young Lee
   Huawei
   TX
   USA

   Email: leeyoung@huawei.com

   Sabine Randriamasy
   Nokia Bell Labs
   Route de Villejust
   NOZAY   91460
   FRANCE

   Email: Sabine.Randriamasy@nokia-bell-labs.com

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

   Email: yry@cs.yale.edu

   Jingxuan Jensen Zhang
   Tongji University
   4800 Caoan Road
   Shanghai   201804
   China

   Email: jingxuan.n.zhang@gmail.com