

ALTO  
Internet-Draft  
Intended status: Standards Track  
Expires: 16 September 2020

J. Zhang  
Tongji University  
K. Gao  
Sichuan University  
J. Wang

Q. Xiang  
Y.R. Yang  
Yale University  
15 March 2020

ALTO Extension: Flow-based Cost Query  
draft-gao-alto-fcs-07

Abstract

ALTO cost maps and endpoint cost services map a source-destination pair into a cost value. However, current filter specifications, which define the set of source-destination pairs in an ALTO query, have two limitations: 1) Only very limited address types are supported (IPv4 and IPv6), which is not sufficient to uniquely identify a flow in networks with fine-grained routing, such as the emerging Software Defined Networks; 2) The base ALTO protocol only defines filters enumerating all sources and all destinations, leading to redundant information in the response; 3) Cannot distinguish transmission types of flows in the query, which makes the server hard to respond the accurate resource consumption. To address these three issues, this document extends the base ALTO protocol with a more fine-grained filter type which allows ALTO clients to select only the concerned source-destination pairs and announce the flow-specific information like data transmission type, and a more expressive address space which allows ALTO clients to make queries beyond the limited IP addresses.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 16 September 2020.

#### Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Table of Contents

1. Introduction . . . . .	3
1.1. Requirement Language . . . . .	5
1.2. Terminology . . . . .	5
1.2.1. Flow . . . . .	5
1.2.2. Data Transmission Type . . . . .	5
2. Overview of Approaches . . . . .	5
2.1. Extended Endpoint Address . . . . .	6
2.2. Flow-based Filter . . . . .	6
2.3. Flow-specific Announcement . . . . .	6
3. Extended Endpoint Address . . . . .	7
3.1. Address Type . . . . .	7
3.2. Endpoint Address . . . . .	8
3.2.1. MAC Address . . . . .	8
3.2.2. Internet Domain Name . . . . .	8
3.2.3. IPv4 Socket Address . . . . .	8
3.2.4. IPv6 Socket Address . . . . .	8
3.3. Address Type Compatibility . . . . .	9
3.4. Examples . . . . .	9
4. Flow-specific Announcement . . . . .	9
4.1. Flow-specific Announcement Type . . . . .	9
4.2. Data Transmission Type Announcement . . . . .	10
5. Extended Cost Query Filters . . . . .	10
5.1. Filtered Cost Map Extension . . . . .	10
5.1.1. Capabilities . . . . .	10
5.1.2. Accept Input Parameters . . . . .	11
5.2. Response . . . . .	12
5.3. Endpoint Cost Service Extension . . . . .	12

5.3.1. Capabilities . . . . .	12
5.3.2. Accept Input Parameters . . . . .	13
5.4. Response . . . . .	14
5.5. Examples . . . . .	15
5.5.1. Information Resource Directory . . . . .	15
5.5.2. Flow-based Filtered Cost Map Example . . . . .	17
5.5.3. Flow-based Endpoint Cost Service Example #1 . . . . .	18
5.5.4. Flow-based Endpoint Cost Service Example #2 . . . . .	19
6. Security Considerations . . . . .	21
7. IANA Considerations . . . . .	21
7.1. ALTO Address Type Registry . . . . .	21
7.2. ALTO Address Type Compatibility Registry . . . . .	22
7.3. ALTO Flow-specific Announcement Registry . . . . .	23
8. References . . . . .	24
8.1. Normative References . . . . .	24
8.2. Informative References . . . . .	24
Appendix A. Acknowledgment . . . . .	25
Appendix B. Change Logs . . . . .	25
Authors' Addresses . . . . .	27

## 1. Introduction

Application-Layer Traffic Optimization (ALTO) protocol [RFC7285] defines several cost query services, like Filtered Cost Map and Endpoint Cost Service, to allow applications to query path costs. Generally, an ALTO cost query service can be regarded as a function transforming a given subset of a specific query space into a network view abstract, where the subset is specified by a PID filter or an endpoint filter. However, the current specification has some limitations.

First, in the base ALTO protocol [RFC7285], the endpoint filter only contains the source and destination IP addresses. In practice, both Internet Service Providers (ISP) and local network administrators may conduct policy-based routing, e.g., P2P traffic may be constrained and has a smaller bandwidth than HTTP traffic. Also, web services with different QoS requirements may be hosted on the same machine with the same IP address but different paths with different QoS metrics.

Second, in the base ALTO protocol [RFC7285], the query space is defined by a source list and a destination list. For a query with N sources and M destinations, the response contains N\*M entries. While such a query schema is well suited for peer-to-peer (P2P) applications where files of the same seed are stored on all hosts, it may lead to a lot of redundancy in use cases such as modern data analytics systems where replicas of the same dataset are stored on only a small subset of servers. Consider a system where the number

of replicas is 3 (the default in HDFS), jointly scheduling  $N$  concurrent transfers only needs a maximum of  $3N$  entries but the base ALTO protocol may return up to  $N^2$  entries.

Third, in the base ALTO protocol [RFC7285], the query does not distinguish among the different transmission types like unicast and multicast. For some use cases like the multi-flow scheduling demonstrated by [I-D.ietf-alto-path-vector], the data transmission between endpoints could be beyond unicast. And in those cases, different transmission types may affect the network resource consumption. If applications can receive the path costs distinguishing the different transmission types, it can help applications perform their data transmission decision better.

Thus, we conclude that the following additional requirements (AR) MUST be satisfied to allow ALTO clients make more accurate and efficient cost queries.

AR-1: The ALTO server SHOULD allow the ALTO client to specify accurate query space in cost query services.

The base ALTO protocol only includes IPv4 and IPv6 addresses as endpoint address types, which may not be sufficient to accurately identify an endpoint with emerging flow-based routing mechanisms. ALTO clients MAY suffer from suboptimal decisions because of such inaccuracy. Thus, the ALTO protocol SHOULD be extended so that clients are able to specify accurate query space, i.e., using more fine-grained endpoint address types.

AR-2: The ALTO server SHOULD allow the ALTO client to specify only the essential query space.

Existing PIDFilter (see Sec 11.3.2.3 in [RFC7285]) and EndpointFilter (see Sec 11.5.1.3 in [RFC7285]) represent the cross-product of sources and destinations, and can introduce a lot of redundancy in certain use cases. This limitation greatly harms the scalability of the ALTO protocol. Thus, the ALTO protocol SHOULD be extended so that ALTO clients are able to specify only the essential cost query space, i.e., the concerned source-destination pairs.

AR-3: The ALTO server SHOULD allow the ALTO client to specify different data transmission types for transmissions in the query space.

The input parameters of existing ALTO cost query services only allow the ALTO client to specify the queried transmissions by sources and destinations. The transmission between each source

and destination will always be considered as the unicast. This limitation may make the ALTO client lose the accurate available resources. Thus, the ALTO protocol SHOULD be extended so that ALTO clients are able to specify different transmission types.

In this document, we describe an ALTO extension specifying flow-based cost queries. The rest of this document is organized as follows. Section 3 introduces several new address types that extend the query space of ALTO cost services. Section 5 describes the extended schema on Filtered Cost Map (FCM) and Endpoint Cost Service (ECS) to support cost queries of arbitrary source-destination combinations with the optional flow-specific information. Section 6 and Section 7 discuss security and IANA considerations.

### 1.1. Requirement Language

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

### 1.2. Terminology

This document uses the same terms as defined in [RFC7285] and [RFC8189] with the following additional term:

#### 1.2.1. Flow

In this document, a flow refers to all communications between two endpoints. A flow is "valid" if and only if there CAN be valid communications between the two endpoints, which oftentimes requires that that two endpoint addresses have compatible address types.

#### 1.2.2. Data Transmission Type

This document use the term "Data Transmission Type" or "Transmission Type" to indicate how an application sends the network flows. See Section 4.2.

## 2. Overview of Approaches

This section presents a non-normative overview of the extension to support flow-based cost query. It assumes the readers are familiar with Filtered Cost Map and Endpoint Cost Service defined in [RFC7285] and their extensions defined in [RFC8189].

### 2.1. Extended Endpoint Address

To allow ALTO clients specify accurate query space in cost query services (AR-1), this document defines several new endpoint address types. An endpoint address with a new type is referred to as an extended endpoint address.

Since the address types of both the source and the destination correspond to the same network flow, they MUST NOT conflict. This document defines an address type conflict table to indicate conflicts. If some source and destination address types in a query conflict with each others, an ALTO server MUST return the corresponding error.

### 2.2. Flow-based Filter

To allow ALTO clients specify only the essential query space in cost query services (AR-2), both PIDFilter and EndpointFilter in the base protocol MUST be extended. The extended filters are referred to as flow-based filters.

A straight-forward way of satisfying AR-2 is to have an ALTO client list all its concerned flows. Despite its simplicity, it MAY be too large in size, especially when many flows have common sources or common destinations in the query. Also from the implementation's perspective, it cannot reuse the functionality to parse a PIDFilter/EndpointFilter.

Thus, the flow-based filters defined in this document allow ALTO clients to include multiple PIDFilter/EndpointFilter objects in the same query. Apparently, if we replace each PIDFilter/EndpointFilter of N sources and M destinations with NM filters that have exactly one source and destination, the two representations refer to the same set of flows. As a result, one can aggregate flows with common sources or destinations in one PIDFilter/EndpointFilter object without introducing redundant flows.

From the implementation's perspective, one MAY reuse an ALTO library which parses PIDFilter/EndpointFilter and/or converts them into a set of source-destination pairs.

### 2.3. Flow-specific Announcement

Some properties are only related to the transmission and cannot be encoded into endpoints, e.g., the data transmission type of a flow. However, these properties may help the ALTO client get more accurate costs.

To allow the ALTO client to specify these flow-specific properties (AR-3), this document introduces an extensible field in the flow-based filter. The ALTO client can announce the flow-specific information in this field. An announcement, whose type is encoded as a FlowSpecAnnounceType (Section 4.1), can represent transmission type, equal cost multipath assumption and other kinds of flow-specific information.

This document adopts an extensible design for this announcement field. Although only the data transmission type is defined in this document, other announcement properties can be defined in future documents and are not in the scope of this document.

### 3. Extended Endpoint Address

This document registers new address types and defines the corresponding formats for endpoint addresses of each new address type.

#### 3.1. Address Type

The new AddressType identifiers defined in this document are as follows:

eth: An endpoint address with type "eth" is the address of an Ethernet interface. It is used to uniquely identify an endpoint in the data link layer.

domain: An endpoint address with type "domain" is the domain name of a web service. It is used to uniquely identify a web service which MAY be translated to one or more IPv4 address(es).

domain6: An endpoint address with type "domain6" is the domain name of a web service. It is used to uniquely identify a web service which MAY be translated to one or more IPv6 address(es).

tcp: An endpoint address with type "tcp" is the address of a TCP socket. It is used to uniquely identify an IPv4 TCP socket in the transport layer.

tcp6: An endpoint address with type "tcp6" is the address of a TCP socket. It is used to uniquely identify an IPv6 TCP socket in the transport layer.

udp: An endpoint address with type "udp" is the address of a UDP socket. It is used to uniquely identify an IPv4 UDP socket in the transport layer.

udp6: An endpoint address with type "udp6" is the address of a UDP socket. It is used to uniquely identify an IPv6 UDP socket in the transport layer.

### 3.2. Endpoint Address

This document defines EndpointAddr when AddressType is in Section 7.1.

#### 3.2.1. MAC Address

An Endpoint Address of type "eth" is encoded as a MAC address, whose format is encoded as specified by either format EUI-48 in [EUI48] or EUI-64 in [EUI64].

#### 3.2.2. Internet Domain Name

An Endpoint Address of type "domain" or "domain6" is encoded as a domain name in the Internet, as specified in Section 11 of [RFC2181]. It MUST have at least one corresponding A ("domain") or AAAA ("domain6") record in the DNS.

#### 3.2.3. IPv4 Socket Address

An Endpoint Address of type "tcp" or "udp" is encoded as an IPv4 socket address. It is encoded as a string of the format Host:Port with the ":" character as a separator. The Host component of an IPv4 socket address is encoded as specified by either an IPv4 address (see Section 10.4.3.1 of [RFC7285]) or an IPv4-compatible domain name (see Section 3.2.2). The Port component of an IPv4 socket address is encoded as an integer between 1 and 65535.

#### 3.2.4. IPv6 Socket Address

An Endpoint Address of type "tcp6" or "udp6" is encoded as an IPv6 socket address. It is also encoded as a string of the format Host:Port with the ":" character as a separator. The Host component of an IPv6 socket address is encoded as specified by either an IPv6 address (see Section 10.4.3.2 of [RFC7285]) enclosed in the "[" and "]" characters as recommended in [RFC2732] or an IPv6-compatible domain name (see Section 3.2.2). The Port component of IPv6 socket address is encoded as an integer between 1 and 65535.



### 3.3. Address Type Compatibility

In practice, a flow with endpoint addresses with different types MAY NOT be valid. For example, a source endpoint with an IPv4 address CANNOT establish a network connection with a destination endpoint with an IPv6 address. Neither can a source with a TCP socket address and a destination with a UDP socket address.

Thus, to explicitly define the compatibility between AddressType identifiers, every ALTO AddressType identifier MUST provide a list of AddressType identifiers that are compatible with it in the "ALTO Address Type Compatibility Registry" Section 7.2. For all sources and destinations in a PIDFilter/EndpointFilter, if the AddressType identifiers of a given pair DO NOT appear in the ALTO Address Type Compatibility Registry, an ALTO server MUST return an ALTO error response with the error code "E\_INVALID\_FIELD\_VALUE" with optional information to help diagnose the incompatibility.

### 3.4. Examples

Some valid endpoint addresses are demonstrated as follows:

```
"eth:98-e0-d9-9c-df-81"  
"domain:www.example.com"  
"tcp:198.51.100.34:5123"  
"udp6:[2000::1:2345:6789:abcd]:8080"
```

## 4. Flow-specific Announcement

Each flow-specific announcement refers to a property of the traffic between a set of source and destination pairs. The interpretation of a property, however, depends on the meaning, i.e., the type, of the property. This document uses flow-specific announcement type as an indicator of the "type" information, and requires the flow-specific announcement types be registered in an IANA registry called "ALTO Flow-specific Announcement Registry".

### 4.1. Flow-specific Announcement Type

It is encoded as a JSONString and has the same format as a PID Name defined in Section 10.1 in [RFC7285]. The type FlowSpecAnnounceType is used in this document to indicate a string of that format.

#### 4.2. Data Transmission Type Announcement

This document defines a flow-specific announcement called the data transmission type. The type of this announcement is indicated by the string "transmission-type", and the value of this announcement is a JSONString of either "unicast", "anycast", "broadcast" or "multicast".

#### 5. Extended Cost Query Filters

This section describes extensions to [RFC7285] and [RFC8189] to support flow-based cost queries.

This document uses the notation rules specified in Section 8.2 of [RFC7285] and also the notation rules for optional fields in Section 4 of [RFC8189].

##### 5.1. Filtered Cost Map Extension

This document extends the Filtered Cost Map as defined in Section 11.3.2 of [RFC7285] and Section 4.1 of [RFC8189], by adding a new capability and new input parameters.

The media type, HTTP method, and "uses" specifications (described in Sections 11.3.2.1, 11.3.2.2, and 11.3.2.5 of [RFC7285], respectively) are not changed.

The format of the response is the same as defined in Section 4.1.3 of [RFC8189]. But this document recommends how to generate the response based on the extended input parameters.

###### 5.1.1. Capabilities

The Filtered Cost Map capabilities are extended with two additional members:

- \* flow-based-filter
- \* flow-spec-announce

The capability "flow-based-filter" indicates whether this resource supports flow-based cost queries, and the capability "flow-spec-announce" indicates which flow-specific announcements are supported. The FilteredCostMapCapabilities object in Section 4.1.1 of [RFC8189] is extended as follows:

```
object {  
  JSONString cost-type-names<1..*>;  
  [JSONBool cost-constraints;]  
  [JSONNumber max-cost-types;]  
  [JSONString testable-cost-type-names<1..*>;]  
  [JSONBool flow-based-filter;]  
  [FlowSpecAnnounceType flow-spec-announce<1..*>;]  
} FilteredCostMapCapabilities;
```

cost-type-names and cost-constraints: As defined in Section 11.3.2.4 of [RFC7285].

max-cost-types and testable-cost-type-names: As defined in Section 4.1.1 of [RFC8189].

flow-based-filter: If true, an ALTO Server allows a field "pid-flows" to be included in the requests. If not present, this field MUST be interpreted as if it is false.

flow-spec-announce: It MUST NOT be present if "flow-based-filter" is not true. If present, the value is the an array of supported flow-specific announcement types.

#### 5.1.2. Accept Input Parameters

The ReqFilteredCostMap object in Section 4.1.2 of [RFC8189] is extended as follows:

```
object {  
  [CostType cost-type;]  
  [CostType multi-cost-types<1..*>;]  
  [CostType testable-cost-types<1..*>;]  
  [JSONString constraints<0..*>;]  
  [JSONString or-constraints<1..*><1..*>;]  
  [PIDFilter pids;]  
  [ExtPIDFilter pid-flows<1..*>;]  
} ReqFilteredCostMap;  
  
object {  
  [FlowSpecAnnounceMap flow-spec-announce;]  
} ExtPIDFilter : PIDFilter;  
  
object-map {  
  FlowSpecAnnounceType -> JSONValue;  
} FlowSpecAnnounceMap;
```

cost-type, multi-cost-types, testable-cost-types, constraints, or-constraints: As defined in Section 4.1.2 of [RFC8189].

**pids:** As defined in Section 11.3.2.3 of [RFC7285].

**pid-flows:** Defined as a list of ExtPIDFilter objects. The ALTO server MUST interpret PID pairs appearing in multiple ExtPIDFilter objects as if they appeared only once. If the capability "flow-spec-announce" is present, the "flow-spec-announce" input parameter can be specified. The value of this field is of type FlowSpecAnnounceMap, which maps a FlowSpecAnnounceType to its corresponding value. The interpretation of the value MUST follow the definition of the flow-specific announcement in the ALTO Flow-specific Announcement Registry.

An ALTO client MUST include either "pids" or "pid-flows" in a query but MUST NOT include both at the same time.

## 5.2. Response

This document does not change the format of the response entity. But the ALTO server responds the request with "pid-flows" filter as follows:

The ALTO server MUST include the path costs of pairs in each ExtPIDFilter in the "pid-flows" filter. If the "flow-spec-announce" field is specified in some ExtPIDFilter, the path costs for flows in this ExtPIDFilter SHOULD respond the flow-specific information announced by this field.

## 5.3. Endpoint Cost Service Extension

This document extends the Endpoint Cost Service as defined in Section 11.5.1 of [RFC7285] and Section 4.2 of [RFC8189], by adding a new capability and input parameters.

The media type, HTTP method, and "uses" specifications (described in Sections 11.5.1.1, 11.5.1.2, and 11.5.1.5 of [RFC7285], respectively) are unchanged.

The format of the response is the same as defined in Section 4.2.3 of [RFC8189]. But this document recommends how to generate the response based on the extended input parameters.

### 5.3.1. Capabilities

The extension to EndpointCostCapabilities includes three additional members:

- \* flow-based-filter

- \* address-types
- \* flow-spec-announce

Only if the capability "flow-based-filter" is present and its value is "true", the ALTO server supports the flow-based extension for this endpoint cost service. The capability "address-types" indicates which endpoint address types are supported by this resource, it MUST NOT be specified if "flow-based-filter" is absent or the value is false. The capability "flow-spec-announce" indicates which flow-specific announcements are supported, just like it works in the Filtered Cost Map resource.

```
object {  
  [JSONBool    flow-based-filter;]  
  [JSONString  address-types<0..*>;]  
  [FlowSpecAnnounceType flow-spec-announce<1..*>;]  
} EndpointCostCapabilities : FilteredCostMapCapabilities;
```

flow-based-filter: If true, an ALTO Server MUST accept field "endpoint-flows" in the requests. If not present, this field MUST be interpreted as if it is specified false.

address-types: Defines a list of AddressType identifiers encoded as a JSONArray of JSONString. All AddressType identifiers MUST be registered in the "ALTO Address Type Registry" (see Section 14.4 of [RFC7285]). An ALTO server SHOULD NOT claim "ipv4" and "ipv6" in this field explicitly, because they are supported by default. If not present, this field MUST be interpreted as if it is an empty array, i.e., the ALTO server only supports the default "ipv4" and "ipv6" address types.

flow-spec-announce: It MUST NOT be present if "flow-based-filter" is not true. If present, the value is the an array of supported flow-specific announcement types.

### 5.3.2. Accept Input Parameters

The ReqEndpointCostMap object in Section 4.2.2 of [RFC8189] is extended as follows:

```

object {
  [CostType cost-type;]
  [CostType multi-cost-types<1..*>;]
  [CostType testable-cost-types<1..*>;]
  [JSONString constraints<0..*>;]
  [JSONString or-constraints<1..*><1..*>;]
  [EndpointFilter endpoints;]
  [ExtEndpointFilter endpoint-flows<1..*>;]
} ReqEndpointCostMap;

object {
  [FlowSpecAnnounceMap flow-spec-announce;]
} ExtEndpointFilter : EndpiontFilter;

```

cost-type, multi-cost-types, testable-cost-types, constraints, or-constraints:

As defined in Section 4.1.2 of [RFC8189].

endpoints: As defined in Section 11.5.1.3 of [RFC7285].

endpoint-flows: Defined as a list of ExtEndpointFilter objects. The ALTO server MUST interpret endpoint pairs appearing in multiple ExtEndpointFilter objects as if they appeared only once. If the capability "flow-spec-announce" is present, the "flow-spec-announce" input parameter can be specified. The interpretation of this field is the same as in Section 5.1.2.

If the AddressType of the source and destination in the same EndpointFilter do not conform the compatibility rule defined in Table 1 of Section 7.1, an ALTO server MUST return an ALTO error response with the error code "E\_INVALID\_FIELD\_VALUE".

An ALTO client MUST specify either "endpoints" or "endpoint-flows", but MUST NOT specify both in the same query.

#### 5.4. Response

This document does not change the format of the response entity. But the ALTO server responds the request with "pid-flows" filter as follows:

The ALTO server MUST include the path costs of pairs in each ExtPIDFilter in the "pid-flows" filter. If the "flow-spec-announce" field is specified in some ExtPIDFilter, the path costs for flows in this ExtPIDFilter SHOULD respond the flow-specific information announced by this field. Especially, if "transmission-type" is specified as "multicast", the ALTO server SHOULD expose all the destination address as a multicast group address, and append the

shared trees to the multicast destination addresses into the response if possible.

## 5.5. Examples

### 5.5.1. Information Resource Directory

The following is an example of IRD with relevant resources of the ALTO server. It provides a default network map, a property map of "ane" domain, a filtered cost map and two endpoint cost resources. All of three cost query resources (filtered cost map and endpoint cost resources) support "flow-based-filter". One endpoint cost resource support "flow-spec-announce" and the compound query extension defined in I-D.ietf-alto-path-vector.

Examples followed this section use the same IRD in this document.

```
GET /directory HTTP/1.1
Host: alto.example.com
Accept: application/alto-directory+json,
        application/alto-error+json

HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: application/alto-directory+json
```

```
{
  "meta" : {
    "default-alto-network-map" : "my-default-network-map",
    "cost-types" : {
      "num-hopcount" : {
        "cost-mode" : "numerical",
        "cost-metric" : "hopcount"},
      "num-routingcost" : {
        "cost-mode" : "numerical",
        "cost-metric" : "routingcost"},
      "ord-routingcost" : {
        "cost-mode" : "ordinal",
        "cost-metric" : "routingcost"},
      "path-vector" : {
        "cost-mode" : "array",
        "cost-metric" : "ane-path"}
    },
    .....,
    Other ALTO cost types as described in RFC7285
    .....,
  },
  "resources" : {
```

```
"my-default-network-map" : {
  "uri" : "http://alto.example.com/networkmap",
  "media-type" : "application/alto-networkmap+json"
},
"flow-based-cost-map" : {
  "uri" : "http://alto.example.com/costmap/multi/filtered",
  "media-type" : "application/alto-costmap+json",
  "accepts" : "application/alto-costmapfilter+json",
  "uses" : [ "my-default-network-map" ],
  "capabilities" : {
    "max-cost-types" : 2,
    "flow-based-filter" : true,
    "cost-type-names" : [ "num-hopcount",
                          "num-routingcost" ]
  }
},
"flow-based-endpoint-cost" : {
  "uri" : "http://alto.example.com/endpointcost/lookup",
  "media-type" : "application/alto-endpointcost+json",
  "accepts" : "application/alto-endpointcostparams+json",
  "capabilities" : {
    "address-types": ["tcp", "udp"],
    "flow-based-filter" : true,
    "cost-type-names" : [ "ord-routingcost",
                          "num-routingcost" ]
  }
},
"path-vector-endpoint-cost" : {
  "uri" : "http://alto.example.com/pathvector/lookup",
  "media-type": "multipart/related;
                type=application/alto-costmap+json",
  "accepts" : "application/alto-endpointcostparams+json",
  "capabilities" : {
    "address-types": ["tcp", "tcp6"],
    "flow-based-filter" : true,
    "flow-spec-announce" : [ "transmission-type" ]
    "cost-type-names" : [ "path-vector" ],
    "ane-property-names": [ "maxresbw" ],
  }
}
}
```



### 5.5.2. Flow-based Filtered Cost Map Example

This example shows how an ALTO client requests a filtered cost map using the "pid-flows" filter. In this case, the ALTO client receives a sparse cost map, which cuts 50% useless cost values from the full mesh.

```
POST /costmap/multi/filtered HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,
       application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-costmapfilter+json

{
  "cost-type": {
    "cost-mode": "numerical",
    "cost-metric": "routingcost"
  },
  "pid-flows": [
    { "srcs": ["PID1"], "dsts": ["PID2", "PID3"] },
    { "srcs": ["PID3"], "dsts": ["PID4"] }
  ]
}

HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: application/alto-costmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "my-default-network-map",
        "tag": "75ed013b3cb58f896e839582504f622838ce670f"
      }
    ],
    "cost-type": {
      "cost-mode": "numerical",
      "cost-metric": "hopcount"
    }
  },
  "cost-map": {
    "PID1": { "PID2": 6, "PID3": 2 },
    "PID3": { "PID4": 1 }
  }
}
```

## 5.5.3. Flow-based Endpoint Cost Service Example #1

This example shows how the ALTO client requests endpoint cost using "flow-based-filter" and extended endpoint addresses. In this case, the ALTO client specifies tcp socket address to get more accurate path cost.

```
POST /endpointcost/lookup HTTP/1.1
Host: alto.example.com
Accept: application/alto-endpointcost+json,
       application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-endpointcostparams+json

{
  "cost-type": {
    "cost-mode": "numerical",
    "cost-metric": "hopcount"
  },
  "endpoint-flows": [
    { "srcs": ["ipv4:192.0.2.2"],
      "dsts": ["ipv4:192.0.2.89", "tcp:cdn1.example.com:21"] },
    { "srcs": ["tcp:203.0.113.45:54321"],
      "dsts": ["tcp:cdn1.example.com:21"] }
  ]
}
```

```
HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: application/alto-endpointcost+json
```

```
{
  "meta": {
    "cost-type": {
      "cost-mode": "numerical",
      "cost-metric": "routingcost"
    }
  },
  "endpoint-cost-map": {
    "ipv4:192.0.2.2": {
      "ipv4:192.0.2.89": 100,
      "tcp:cdn1.example.com:21": 20
    },
    "tcp:203.0.113.45:54321": {
      "tcp:cdn1.example.com:21": 80
    }
  }
}
```

## 5.5.4. Flow-based Endpoint Cost Service Example #2

This example shows the integration of the path vector extension and the flow-based query. And in this example, the ALTO client specifies the flow from "tcp6:203.0.113.45:54321" to "tcp6:group1.example.com:21" is multicast. So the ALTO server will expose the destination IP as a multicast group IP, and find the multicast destinations "fe80::40e:9594:da3d:34b" and "fe80::826:daff:feb8:1bb". Then the ALTO server will append the cost for the shared tree into the "endpoint-cost-map".

```
POST /pathvector/lookup 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"
  },
  "endpoint-flows": [
    { "srcs": ["ipv4:192.0.2.2"],
      "dsts": ["tcp:192.0.2.89:21",
               "tcp:cdn1.example.com:21"] },
    { "srcs": ["tcp6:203.0.113.45:54321"],
      "dsts": ["tcp6:group1.example.com:21"],
      "flow-spec-announce": {
        "transmission-type": "multicast" } }
  ],
  "ane-property-names": ["maxresbw"]
}
```

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

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

```
{
  "meta": {
    "vtags": {
```

```

        "resource-id": "path-vector-endpoint-cost.ecs",
        "tag": "bb6bb72eafe8f9bdc4f335c7ed3b10822a391cef"
    },
    "cost-type": {
        "cost-mode": "array",
        "cost-metric": "ane-path"
    }
},
"endpoint-cost-map": {
    "ipv4:192.0.2.2": {
        "tcp:192.0.2.89:21": [ "ane:S1", "ane:D1" ],
        "tcp:cdn1.example.com:21": [ "ane:S1", "ane:D2", "ane:D3" ]
    },
    "tcp6:203.0.113.45:54321": {
        "tcp6:group1.example.com:21": [ "ane:S2", "ane:D3" ]
    },
    "tcp6:group1.example.com:21": {
        "tcp6:[fe80::40e:9594:da3d:34b]:21": [ "ane:G1" ],
        "tcp6:[fe80::826:daff:feb8:1bb]:21": [ "ane:G2" ],
    }
}
}

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

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "path-vector-endpoint-cost.ecs",
        "tag": "bb6bb72eafe8f9bdc4f335c7ed3b10822a391cef"
      }
    ]
  },
  "property-map": {
    "ane:S1": { "maxresbw": 100 },
    "ane:S2": { "maxresbw": 100 },
    "ane:D1": { "maxresbw": 150 },
    "ane:D2": { "maxresbw": 80 },
    "ane:D3": { "maxresbw": 150 },
    "ane:G1": { "maxresbw": 100 },
    "ane:G2": { "maxresbw": 100 }
  }
}

```

## 6. Security Considerations

As discussed in Section 15.4 of [RFC7285], an ALTO server or a third party who is able to intercept the flow-based cost query messages MAY store and process the obtained information in order to analyze user behaviors and communication patterns. Since flow-based cost queries MAY potentially provide more accurate information, an ALTO client should be cognizant about the trade-off between redundancy and privacy.

## 7. IANA Considerations

This document defines new address types to be registered to an existing ALTO registry, and a new registry for their compatible address types.

### 7.1. ALTO Address Type Registry

This document defines several new address types to be registered to "ALTO Address Type Registry", listed in Table 1.

Identifier	Address Encoding	Prefix Encoding	Mapping to/from IPv4/v6
eth	See Section 3.2.1	None	Mapping to/from IPv4 by [RFC0903] and [RFC0826]; Mapping to/from IPv6 by [RFC3122] and [RFC4861]
domain	See Section 3.2.2	None	Mapping to/from IPv4 by [RFC1034]
domain6	See Section 3.2.2	None	Mapping to/from IPv6 by [RFC3596]
tcp	See Section 3.2.3	None	No mapping
tcp6	See Section 3.2.4	None	No mapping
udp	See Section 3.2.3	None	No mapping
udp6	See Section 3.2.4	None	No mapping

Table 1: ALTO Address Type Registry

## 7.2. ALTO Address Type Compatibility Registry

This document proposes to create a new registry called "ALTO Address Type Compatibility Registry", whose purpose is stated in Section 3.3.

The compatible address type identifiers of the ones registered in the ALTO Address Type Registry are listed in Table 2.

Identifier	Compatible Identifiers
eth	ipv4, ipv6
domain	eth, ipv4
domain6	eth, ipv6
tcp	eth, ipv4, domain
tcp6	eth, ipv6, domain6
udp	eth, ipv4, domain
udp6	eth, ipv6, domain6

Table 2: ALTO Address Type  
Compatibility Registry

The entry of an address type identifier SHOULD only include the identifiers registered before it. The compatibility between address types are bidirectional. For example, although "eth" does not register "tcp" as its compatible identifier, an ALTO server MUST recognize them as compatible because "eth" is registered as a compatible identifier of "tcp".

Any new ALTO address type identifier registered after this document MUST register their compatible identifiers in this registry simultaneously.

### 7.3. ALTO Flow-specific Announcement Registry

This document proposes to create a new registry called "ALTO Flow-specific Announcement Registry", whose purpose is stated in Section 4. An initial registry entry is also documented as shown in Table 3.

Type	Interpretation
transmission-type	See Section 4.2

Table 3: ALTO Flow-specific  
Announcement Registry

## 8. References

### 8.1. Normative References

- [EUI48] IEEE, ., "Guidelines for use of a 48-bit Extended Unique Identifier (EUI-48)", 2012.
- [EUI64] IEEE, ., "Guidelines for use of a 64-bit Extended Unique Identifier (EUI-64)", 2012.
- [I-D.ietf-alto-path-vector]  
Gao, K., Randriamasy, S., Yang, Y., and J. Zhang, "ALTO Extension: Path Vector", Work in Progress, Internet-Draft, draft-ietf-alto-path-vector-10, 9 March 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-alto-path-vector-10.txt>>.
- [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>>.

### 8.2. Informative References

- [RFC0826] Plummer, D., "An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, DOI 10.17487/RFC0826, November 1982, <<https://www.rfc-editor.org/info/rfc826>>.
- [RFC0903] Finlayson, R., Mann, T., Mogul, J.C., and M. Theimer, "A Reverse Address Resolution Protocol", STD 38, RFC 903, DOI 10.17487/RFC0903, June 1984, <<https://www.rfc-editor.org/info/rfc903>>.
- [RFC1034] Mockapetris, P.V., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [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>>.
- [RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997,  
<<https://www.rfc-editor.org/info/rfc2181>>.
- [RFC2732] Hinden, R., Carpenter, B., and L. Masinter, "Format for Literal IPv6 Addresses in URL's", RFC 2732, DOI 10.17487/RFC2732, December 1999,  
<<https://www.rfc-editor.org/info/rfc2732>>.
- [RFC3122] Conta, A., "Extensions to IPv6 Neighbor Discovery for Inverse Discovery Specification", RFC 3122, DOI 10.17487/RFC3122, June 2001,  
<<https://www.rfc-editor.org/info/rfc3122>>.
- [RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", STD 88, RFC 3596, DOI 10.17487/RFC3596, October 2003,  
<<https://www.rfc-editor.org/info/rfc3596>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007,  
<<https://www.rfc-editor.org/info/rfc4861>>.

#### Appendix A. Acknowledgment

The authors would like to thank Dawn Chen, Haizhou Du, Sabine Randriamasy and Wendy Roome for their fruitful discussions and feedback on this document. Shawn Lin also gave substantial review feedback and suggestions on the protocol design.

#### Appendix B. Change Logs

Note to Editor: Please remove this section prior to publication.

This section records the change logs of the draft updates.

Changes since -06 versions:

- \* Clarify the type of "flow-spec-announcement" in both the request and the response
- \* Modify examples to be compatible with the latest path vector document

- \* Add a new registry for flow-specific announcements

- \* Fix some typos

Changes since -05 versions:

- \* Add flow-specific information announcement in the flow-based filter.
- \* Modify examples and add descriptions to Make them clear.
- \* Rename the address type "Domain Name" to "Internet Domain Name" to distinguish it with the "Domain Name" in the unified properties draft.

Changes since older versions:

Changes since -04 revision:

- \* Improve the clarity of the document by explicitly stating the problems.
- \* Keep only "flow" in the terminology section.
- \* Move section 6 "Advanced Flow-based Query" out of this document.
- \* Change "ALTO Address Type Conflicts Registry" to "ALTO Address Type Compatibility Registry".

Since -03 revision:

- \* Remove some irrelevant content from the draft.
- \* Improve the description of the new endpoint address type identifier registry. And add a new registry to declare the conflicting address type identifiers.

Since -02 revision:

- \* Change "EndpointURI" to "AddressType::EndpointAddr" for consistency.
- \* Replace "Cost Confidence" by "Cost Statistics" for compatibility.

Since -01 revision:

- \* Define the basic flow-based query extensions for Filtered Cost Map and Endpoint Cost service. The basic flow-based query is downward

compatible with the legacy ALTO service. It does not introduce any new media types.

- \* Move the service of media-type "application/alto-flowcost+json" to the advanced flow-based query extension. It will ask ALTO server to support the new media type.

Since -00 revision:

- \* Change the schema of "pid-flows" and "endpoint-flows" fields from pair list to pair mesh list.

#### Authors' Addresses

Jingxuan Jensen Zhang  
China  
201804  
Shanghai  
4800 Caoan Road  
Tongji University

Email: jingxuan.n.zhang@gmail.com

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

Email: kaigao@scu.edu.cn

Junzhuo Wang

Qiao Xiang  
Yale University  
51 Prospect Street  
New Haven, CT  
United States of America

Email: qiao.xiang@cs.yale.edu

Yang Richard Yang  
Yale University

51 Prospect Street  
New Haven, CT  
United States of America

Email: yry@cs.yale.edu

ALTO WG  
Internet-Draft  
Intended status: Standards Track  
Expires: January 5, 2018

K. Gao  
Tsinghua University  
X. Wang  
Tongji University  
Q. Xiang  
Tongji/Yale University  
C. Gu  
Tongji University  
Y. Yang  
Yale University  
G. Chen  
Huawei  
July 4, 2017

A Recommendation for Compressing ALTO Path Vectors  
draft-gao-alto-routing-state-abstraction-06.txt

Abstract

The path vector extension [I-D.ietf-alto-path-vector] has extended the original ALTO protocol [RFC7285] with the ability to represent a more detailed view of the network, containing not only end-to-end metrics but also information about shared bottlenecks.

However, the view computed by straw man algorithms can contain redundant information and result in unnecessary communication overhead. The situation gets even worse when certain ALTO extensions are enabled, for example, the incremental update extension [I-D.ietf-alto-incr-update-sse] which continuously pushes data changes to ALTO clients. Redundant information can trigger unnecessary updates.

In this document, an algorithm is described which can effectively reduce the redundancy in the network view while still providing the same information as in the original path vectors. The algorithm is fully compatible with the path vector extension and has several by-products which can be leveraged by other extensions to achieve better performances.

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 5, 2018.

## Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Changes Since Version -03, -04 and -05 . . . . .	4
3. Terminology . . . . .	5
4. Compressing Path Vectors . . . . .	5
5. Equivalent Transformation Algorithm . . . . .	7
5.1. Parameters and Variables . . . . .	7
5.2. Algorithm Description . . . . .	8
6. Recommended Redundancy Check Algorithm . . . . .	9
6.1. Parameters and Variables . . . . .	10
6.2. Algorithm Description . . . . .	10
7. Reducing Unnecessary Incremental Updates . . . . .	11
8. Extension for Customized Input Parameters . . . . .	11
8.1. Parameters and Variables . . . . .	11
8.2. Algorithm Description . . . . .	12

8.3. ALTO Extension for Client Constraint Map . . . . .	12
8.4. Examples . . . . .	14
8.5. Compatibility . . . . .	20
9. Security Considerations . . . . .	20
10. IANA Considerations . . . . .	21
11. Acknowledgments . . . . .	21
12. References . . . . .	21
12.1. Normative References . . . . .	21
12.2. Informative References . . . . .	21
Authors' Addresses . . . . .	22

## 1. Introduction

The path vector extension [I-D.ietf-alto-path-vector] has extended the base ALTO protocol [RFC7285] with the ability to present more complex network views than the simple abstraction used by Cost Map or Endpoint Cost Service. This has enabled ALTO clients to query more sophisticated information such as shared bottlenecks, by which ALTO clients can schedule their flows properly to avoid congestion and to better utilize the network resources.

A straw man approach, especially in the context of Software Defined Networking (SDN) where network providers have a global view, can compute the path vectors by retrieving the paths for all requested flows and returning the links on those paths as abstract network elements. However, this approach has several drawbacks:

- o The resultant network view may lead to privacy leaks. Since the paths constitute a sub-graph of the global view, they may contain sensitive information without further processing.
- o The resultant network view may contain redundant information. The path vector information is primarily used to avoid network bottlenecks. Thus, if a link cannot become the bottleneck, as demonstrated in Section 4, it is considered as redundant. Redundant links not only increase the communication overhead of the path vector extension, but also trigger false-positive data change events when the incremental update extension is activated.

This document describes an algorithm that identifies redundant abstract network elements and reduces them as much as possible. The algorithm, namely the "equivalent transformation" algorithm, can be integrated with any implementation of the path vector extension as a post-processing step. As the name suggests, this algorithm essentially conducts "equivalent" transformations on the original path vectors, removes redundant information and obtains a more compact view.

This extension is fully compatible with the path vector extension and can be optionally turned on and off without affecting the correctness of responses. A crucial part of the equivalent transformation algorithm is how to find redundant abstract network elements. By tuning the redundancy check algorithm, one can make different trade-off decisions between efficiency and privacy. A reference implementation of redundancy check algorithm is also described in this document.

Furthermore, the redundancy check algorithm can generate filters for incremental updates of path vector queries. It can also accept customized parameters from ALTO clients to achieve even better compression results.

This document is organized as follows. Section 4 gives a concrete example to demonstrate the importance of compressing path vectors. Section 5 gives the equivalent transformation algorithm and Section 6 introduces a reference implementation of redundancy check algorithm. Section 7 discusses how to generate filters for ALTO incremental update services and Section 8 introduces an optional extension which allows ALTO clients to share certain information and further improves the performance of equivalent transformation. Finally, Section 9 and Section 10 discuss security and IANA considerations.

## 2. Changes Since Version -03, -04 and -05

In early versions of this draft, a lot of contents are shared with the path vector draft. From version -04, the authors have adjusted the structure and target this document as a supplement of the path vector extension with

1. the equivalent transformation algorithm which compresses original the path vectors to provide a more compact network view,
2. a reference implementation of the redundancy check algorithm which provides near-optimal results, and
3. the client constraint map extension which allows ALTO clients to optionally provide additional client-side information to help further reduce the communication overhead.

The document also discusses how the algorithms and extension introduced here can cooperate with existing working group drafts, such as Incremental Updates, Multi-Cost and Cost Calendar.

The latest version (-06) also fixed some minor issues in -04 and -05.



### 3. Terminology

This document uses the same terms as in [I-D.ietf-alto-path-vector].

### 4. Compressing Path Vectors

We use the example shown in Figure 1 to demonstrate the importance of compressing path vectors. The network has 6 switches (sw1 to sw6) forming a dumbbell topology. Switches sw1/sw3 provide access on one side, sw2/sw4 provide access on the other side, and sw5/sw6 form the backbone. End hosts eh1 to eh4 are connected to access switches sw1 to sw4 respectively. Assume that the bandwidth of each link is 100 Mbps, and that the network is abstracted with 4 PIDs each representing a host at one access switch.

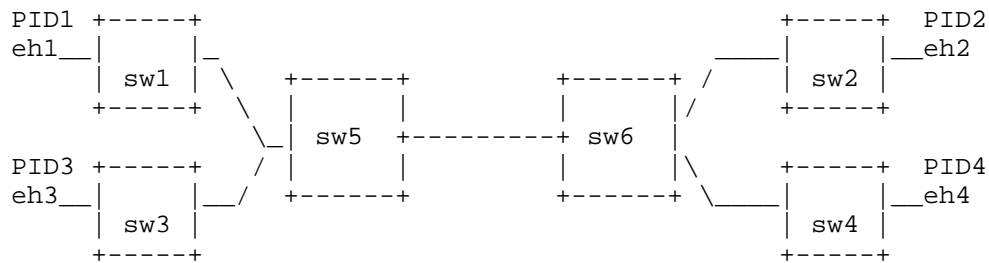


Figure 1: Raw Network Topology

Link	Description
link1	sw1 <==> sw5
link2	sw2 <==> sw6
link3	sw3 <==> sw5
link4	sw4 <==> sw6
link5	sw5 <==> sw6

Table 1: Description of the Links

Consider an application which schedules the traffic consisting of two flows, eh1 -> eh2 and eh3 -> eh4. The application can query the path vectors and a straw man implementation will return all 5 links (abstract network elements) as shown in Figure 2.

```

path vectors:
  eh1: [ eh2: [ane:l1, ane:l5, ane:l2]]
  eh3: [ eh4: [ane:l3, ane:l5, ane:l4]]

abstract network element property map:
  ane:l1 : 100 Mbps
  ane:l2 : 100 Mbps
  ane:l3 : 100 Mbps
  ane:l4 : 100 Mbps
  ane:l5 : 100 Mbps

```

Figure 2: Path Vectors Returned by Straw Man Implementation

The resultant path vectors represent the following linear constraints on the available bandwidth for the two flows:

```

bw(eh1->eh2)          <= 100 Mbps (ane:l1)
bw(eh1->eh2)          <= 100 Mbps (ane:l2)
                        bw(eh3->eh4) <= 100 Mbps (ane:l3)
                        bw(eh3->eh4) <= 100 Mbps (ane:l4)
bw(eh1->eh2) + bw(eh3->eh4) <= 100 Mbps (ane:l5)

```

Figure 3: Linear Constraints Represented by the Path Vectors

It can be seen that the constraints of ane:l1 and ane:l2 are exactly the same, and so are ane:l3 and ane:l4. Intuitively, we can replace ane:l1 and ane:l2 with a new abstract network element "ane:1", and similarly replace ane:l3 and ane:l4 with "ane:2". The new path vectors are shown in Figure 4.

```

path vectors:
  eh1: [ eh2: [ane:1, ane:l5]]
  eh3: [ eh4: [ane:2, ane:l5]]

abstract network element property map:
  ane:1 : 100 Mbps
  ane:2 : 100 Mbps
  ane:l5 : 100 Mbps

```

Figure 4: Path Vectors after Merging ane:l1/ane:l2 and ane:l3/ane:l4

Taking a deeper look at Figure 3, it can be seen that constraints of ane:1 (ane:l1/ane:l2) and ane:2 (ane:l3/ane:l4) can be implicitly derived from that of ane:l5. Thus, these constraints are considered redundant and the path vectors in Figure 4 can be further reduced. We replace ane:l5 with a new "ane:3" and the new path vectors are shown in Figure 5.

```
path vectors:
  eh1: [ eh2: [ane:3]]
  eh3: [ eh4: [ane:3]]

abstract network element property map:
  ane:3 : 100 Mbps
```

Figure 5: Path Vectors after Removing Redundant Elements

It is clear that the new path vectors (Figure 5) are much more compact than the original path vectors (Figure 2) but they contain just as much information. Meanwhile, the application can hardly infer anything about the original topology with the compact path vectors.

To reduce the communication overhead and improve the privacy protection of the path vector extension, an algorithm is described in this document to efficiently compute the compact path vectors.

## 5. Equivalent Transformation Algorithm

This section describes the path vector compression algorithm, namely the "equivalent transformation" algorithm.

### 5.1. Parameters and Variables

The equivalent transformation algorithm accepts 3 parameters: the original path vectors *P*, the corresponding abstract network element property map *M*, and a redundancy check algorithm *R(P, M)*.

**Original path vectors *P*:** The original path vectors *P* MUST have the format of a cost map or an endpoint cost map, where each cost value is a JSONArray of abstract network element names, as defined in [I-D.ietf-alto-path-vector].

**Abstract network element property map *M*:** The abstract network element property map *M* MUST contain all the abstract network elements whose names are included in the original path vectors *P*. It MUST contain at least one valid ALTO cost type which is supported by the corresponding path vector service, but MUST NOT contain ordinal values. Unless it is specifically defined in another extension, the cost values MUST follow the associative addition rule, e.g.  $\text{cost}(\text{ane1}) + \text{cost}(\text{ane2}) = \text{cost}(\text{ane1} \circ \text{ane2}) = \text{cost}(\text{ane2}) + \text{cost}(\text{ane1})$  where  $\text{ne1} \circ \text{ne2}$  represents a virtual "ane-path" consisting of *ane1* and *ane2*. One exception is bandwidth, where the "addition" is actually a "minimum" function.

Redundancy check algorithm R(P, M): The redundancy check algorithm R(P, M) MUST accept the two parameters P and M as specified above. It MUST return a list of abstract network element names, representing those whose corresponding constraints are redundant.

In addition to the parameters mentioned above, the algorithm also maintains the following variables.

Temporary path vectors P0: P0 store the temporary values after each step of equivalent transformation.

Temporary abstract network element property map M0: M0 stores the temporary value of an abstract network element property map after each step of equivalent transformation.

Reverse abstract network element map RM: RM is a map whose key is an abstract network element name with the value being a set of endpoint/PID pairs.

Redundant abstract network element set S: S contains the result of the redundancy check algorithm R(P, M).

## 5.2. Algorithm Description

The equivalent transformation consists of the following steps:

1. When the algorithm starts execution, it sets  $P0 = P$  and  $M0 = M$
2. For each abstract network element name "n", find the set of endpoint/PID pairs  $\{(a, b)\}$  where "n" appears in the path vector of  $a \rightarrow b$  in P0. Put the resultant set of endpoint/PID pairs in RM with "n" as the key.
3. Group RM by the value sets, e.g. put all (n, v) which have the same set of endpoint/PID pairs in the same group. It is guaranteed that each abstract network element name only appears once in one group. Now use the groups to construct a partition of all abstract network element names, where each partition contains all the abstract network element names from a single group. Each partition is associated with a unique ID, which follows the format of an abstract network element name as defined in [I-D.ietf-alto-path-vector].
4. For each endpoint/PID pair in P0, replace the abstract network element names with their group IDs. Construct an empty abstract network element property map M1. For each group, create an abstract network element property entry "e" where each abstract network element property is the "sum" of the abstract network

element property values in M0 of all abstract network elements in the group. Put "e" in M1 with the group ID as the key and also the abstract network element name. Replace M0 with M1.

5. Pass P0 and M0 to redundancy check algorithm  $R(P,M)$ , and store the result in S.
6. If only bandwidth is contained in M0, go to 7. Otherwise, go to 8.
7. For each endpoint/PID pair, remove the abstract network element names in S from the path vectors. Remove the entries in M0 whose keys are in S. Go to 10.
8. Construct an empty abstract network element property map M1. For each abstract network element property entry in M0, if the abstract network element name is not in S, put the entry in M1. For each abstract network element name "n" in S, find the corresponding set of endpoint/PID pairs in RM. For each pair, replace "n" in the corresponding path vector to a new unique abstract network element name and put an entry in M1 whose key is the new abstract network element name while the value being the value of "n" in M0. Replace M0 with M1.
9. Repeat steps 2-4 and go to 10.
10. Create a virtual abstract network element "n" with a unique abstract network element name and sufficiently large bandwidth value. For each endpoint/PID pair in P, if the path vector is an empty set (this only happens when only bandwidth is requested), put the name of "n" in the path vector and add "n" to M0.
11. Return P0 as the path vector response and M0 as the corresponding abstract network element property map.

The term "sum" in step 4 is in quotes because the exact meaning depends on the property types. As stated earlier, the values MUST NOT be ordinal and MUST follow the associative addition rule unless specifically defined in a later extension. This document defines one exception -- bandwidth -- whose addition operator is the "minimum" function, which satisfies the associative addition rule.

#### 6. Recommended Redundancy Check Algorithm

In this section, an algorithm is described as a reference implementation of the redundancy check algorithm  $R(P, M)$ .

### 6.1. Parameters and Variables

The algorithm takes two parameters: the path vectors P and the corresponding abstract network element property map M.

Path vectors P: As specified in Section 5.1.

Abstract network element property map M: As specified in Section 5.1.

In addition to the parameters, this algorithm also maintains the following variable.

Set of linear constraints C: The set of linear constraints which can be derived from P and M.

### 6.2. Algorithm Description

The algorithm consists of the following steps:

1. If the abstract network element properties in M do not include bandwidth, return the set of all abstract network elements.
2. Construct the set of linear constraints, C. For each endpoint/PID pair, define a variable "x\_i" with a unique ID "i". Construct RM as specified in step 3 of Section 5.2. For each abstract network element "n", find the corresponding set of endpoint/PID pairs "p\_n" in RM. Construct a linear constraint "c\_n: A\_n X <= b\_n". The left hand side is the sum of all the variables "{x\_i}" whose coefficient "a\_i" is 1 if the associated pair is in "p\_n" and otherwise 0. The right hand side is the bandwidth of "n". Put "c\_n" in C.
3. For each "c\_i: A\_i X <= b\_i" in C, construct a new linear programming problem:
 
$$\max A_i X,$$
 where  $A_j X \leq b_j$ , j is not equal to i.
4. Solve this linear programming problem, let the maximum value be "z". If "z <= b\_i", this constraint is redundant. Otherwise the constraint is NOT redundant.
5. Repeat steps 3-4 until the redundancy of all abstract network elements in the original path vectors has been identified. Return the set of abstract network elements whose corresponding linear constraints are redundant.

## 7. Reducing Unnecessary Incremental Updates

This section describes how an ALTO server implementation can use the results in the redundancy check algorithm described in Section 6 to filter unnecessary incremental updates.

Consider the example in Section 4 where the bandwidth of link1 ( $s1 \leftrightarrow s5$ ) has increased from 100 Mbps to 150 Mbps. Straw man approaches may push incremental updates to ALTO clients without considering how the value changes. On the other hand, this link is not included in the path vectors after equivalent transformation, and one can conclude from the redundancy check algorithm Section 6.2 that it can only be non-redundant if " $b_i < z \leq 100$  Mbps". Since the new " $b_i$ " is 150 Mbps, this condition does not hold, i.e., link1 is still redundant in the updated path vector. In that case, ALTO server MAY NOT push the incremental updates.

However, this filter only works for redundant abstract network elements, e.g. " $z \leq b_i$ ". In all other cases, the path vectors have to be recomputed to guarantee equivalence.

## 8. Extension for Customized Input Parameters

This section introduces an optional extension which can be leveraged by ALTO clients to help reduce the communication overhead of path vector services. In order to do so, a revised version of Section 6 must be introduced.

### 8.1. Parameters and Variables

The algorithm takes three parameters: the path vectors  $P$ , the corresponding abstract network element property map  $M$ , and client constraint map  $CM$ .

Path vectors  $P$ : Same as in Section 6.1.

Abstract network element property map  $M$ : Same as in Section 6.1.

Client constraint map  $CM$ : Client constraint map  $CM$  has the same format as a cost map, where the first key represents the source endpoint address/PID name, the second key represents the destination endpoint address/PID name, and the value is a non-negative float number representing the upper bound bandwidth value for the given endpoint/PID pair.

The algorithm also maintains the following variables:

Set of linear constraints  $C$ : The same as  $C$  in Section 6.1.

Set of client constraints CC: The set of linear constraints which can be derived from CM.

## 8.2. Algorithm Description

The algorithm consists of the following steps:

1. The same as step 1 in Section 6.2.
2. The same as step 2 in Section 6.2.
3. For each endpoint/PID pair in CM, assume the value is a non-negative number "u". If the pair is also in the original path vector, construct a linear constraint "cc: x\_i <= u" and add it to CC where "x\_i" is the variable associated with the same pair in step 2.
4. For each "c\_i: A\_i X <= b\_i" in C, construct a linear programming problem:

$$\begin{array}{ll} \max & A_i X \\ \text{where} & A_j X \leq b_j \text{ in } C, j \text{ is not equal to } i \\ & x_j \leq u_j \text{ in } CC \end{array}$$

5. The same as step 4 in Section 6.2.
6. The same as step 5 in Section 6.2.

Clearly, the feasible region for each linear programming problem in step 4 is smaller or equal to the one in Section 6.2. Thus, each abstract network element has a higher chance of being redundant.

## 8.3. ALTO Extension for Client Constraint Map

This section describes the extensions needed to enable client constraint map.

Any ALTO resource/service that supports client constraint map MUST also support the path vector extension and accept the cost type whose cost mode is "array" and cost metric "ane-path". This extension does not change the specifications on "media types", "HTTP methods", "uses", and "response".

### 8.3.1. Capabilities

The client constraint map extension requires a new capability field "client-constraint-map" in the IRD.



```
object {  
  JSONString cost-type-names<1..*>;  
  [JSONBool client-constraint-map];  
  ... capabilities defined by other extensions  
} ClientConstraintMapCapabilities;
```

cost-type-names: As defined in Section 11.3.2.4 of [RFC7285].

client-constraint-map: If present and the value is "true", it means the resource/service accepts client constraint map in the parameters. Otherwise, the client MUST assume the server does not support this extension and MUST NOT include client constraint map in input parameters.

### 8.3.2. Accept Input Parameters

For filtered cost map with client constraint map extension, it MUST accept the following parameters:

```
object {  
  [CostType cost-type];  
  [PIDFilter pids];  
  [ClientConstraintPIDMap client-constraint-map];  
  ... input parameters defined by other extensions  
} ReqFilteredCostMap;
```

```
object {  
  PIDName -> ClientConstraintPIDGroup;  
} ClientConstraintPIDMap;
```

```
object {  
  PIDName -> JSONNumber;  
} ClientConstraintPIDGroup;
```

cost-type: As defined in Section 11.3.2.3 in [RFC7285]. It MUST have the cost mode "array" and cost metric "one-path" if the field "client-constraint-map" is present. Otherwise, the ALTO server MUST return an error with error code "E\_INVALID\_FIELD\_VALUE".

pids: As defined in Section 11.3.2.3 in [RFC7285].

client-constraint-map: The client constraint map MUST have the format as a JSON object "ClientConstraintPIDMap". All the PID names in the client constraint map MUST also be included in "pids", otherwise the ALTO server MUST return an error with error code "E\_INVALID\_FIELD\_VALUE". If the value for a given PID pair is 0, ALTO server MUST not include this pair in the path vector.

Similarly, the input parameters for endpoint cost services with client constraint map MUST have the following format:

```
object {  
  [CostType cost-type;]  
  EndpointFilter endpoints;  
  [ClientConstraintEndpointMap client-constraint-map;]  
  ... input parameters defined by other extensions  
} ReqEndpointCostMap;  
  
object {  
  TypedEndpointAddr -> ClientConstraintEndpointGroup;  
} ClientConstriantEndpointMap;  
  
object {  
  TypedEndpointAddr -> JSONNumber;  
} ClientConstraintEndpointGroup;
```

cost-type: Same as above.

endpoints: As defined in Section 11.5.1.3 of [RFC7285].

client-constraint-map: The client constraint map contained in the parameter MUST have the format of a JSON object "ClientConstriantEndpointMap". All the endpoint addresses MUST also appear in the "endpoints", otherwise the ALTO server MUST return an error with an error code "E\_INVALID\_FIELD\_VALUE". If the value for a given endpoint pair is 0, ALTO server MUST NOT included this pair in the path vector.

#### 8.4. Examples

This section contains a series of examples for the client constraint map extension.

##### 8.4.1. Information Resource Directory Example

```
{
  "meta": {
    "cost-types": {
      "pv-ane": {
        "cost-mode": "array",
        "cost-metric": "ane-path"
      }
    }
  },
  "resource": {
    "default-network-map": {
      "uri": "http://alto.example.com/networkmap",
      "media-type": "application/alto-networkmap+json"
    },
    "filtered-multi-cost-map": {
      "uri": "http://alto.example.com/costmap/filtered",
      "media-type": "application/alto-costmap+json",
      "accepts": "application/alto-costmapfilter+json",
      "uses": ["default-network-map"],
      "capabilities": {
        "cost-type-names": ["pv-ane"],
        "property-map": "default-prop-map",
        "client-constraint-map": true
      }
    },
    "default-endpoint-cost-map": {
      "uri": "http://alto.example.com/endpointcost/lookup",
      "media-type": "application/alto-endpointcostmap+json",
      "accepts": "application/alto-endpointcostparams+json",
      "capabilities": {
        "cost-type-names": ["pv-ane"],
        "client-constraint-map": true
      }
    },
    "default-prop-map": {
      "uri": "http://alto.example.com/default-prop-map",
      "media-type": "application/alto-propmap+json",
      "accepts": "application/alto-propmapparams+json",
      "capabilities": {
        "domain-types": ["ane"],
        "prop-types": [ "availbw" ]
      }
    }
  }
}
```

#### 8.4.2. Filtered Cost Map Example

Assume we use the example in Section 4 and PID1-PID4 are mapped to eh1-eh4 respectively.

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

```
{
  "cost-type": {
    "cost-mode": "array",
    "cost-metric": "ane-path"
  },
  "pids": {
    "srcs": ["PID1", "PID3"],
    "dsts": ["PID2", "PID4"]
  },
  "client-constraint-map": {
    "PID1": { "PID2": 40, "PID4": 0 },
    "PID3": { "PID2": 50, "PID4": 50 }
  }
}
```

```
HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: multipart/related; boundary=31415926
```

```
--31415926
Content-Type: application/alto-costmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "default-network-map",
        "tag": "75ed013b3cb58f896e839582504f622838ce670f"
      }
    ],
    "vtag": {
      "resource-id": "cost-map-pv",
      "tag": "27612897acf278ffu3287c284dd28841da78213",
      "query-id": "query-cost-map-pv-276128"
    },
    "cost-type": {
      "cost-mode": "array",
      "cost-metric": "ane-path"
    }
  },
  "cost-map": {
    "PID1": {
      "PID2": [ "ane:1" ]
    },
    "PID3": {
      "PID2": [ "ane:1" ],
      "PID4": [ "ane:1" ]
    }
  }
}
```

```
--31415926
Content-Type: application/alto-propmap+json

{
  "property-map": {
    "ane:1": { "availbw": 100 }
  }
}
```

--31415926--

Since the bandwidth of all links is 100 Mbps, it can be easily concluded that only link5 can potentially become a bottleneck with the given client constraints. Thus, only one abstract network element is returned.

#### 8.4.3. Endpoint Cost Service Example

Assume we use the example in Section 4 and eh1-eh4 are associated with IP addresses 192.0.2.1-192.0.2.4 respectively.

```
POST /endpointcost/lookup HTTP/1.1
Host: alto.example.com
Accept: 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": ["ipv4:192.0.2.1", "ipv4:192.0.2.3"],
    "dsts": ["ipv4:192.0.2.2", "ipv4:192.0.2.4"]
  },
  "client-constraint-map": {
    "ipv4:192.0.2.1": { "ipv4:192.0.2.2": 40, "ipv4:192.0.2.4": 0 },
    "ipv4:192.0.2.3": { "ipv4:192.0.2.2": 50, "ipv4:192.0.2.4": 50 }
  }
}

HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: application/alto-endpointcost+json
```

```

{
  "meta": {
    "vtag": {
      "resource-id": "default-prop-map",
      "tag": "a911354dfd1ef6555bfe7af07d3af0bfebe7c8a9",
      "query-id": "query-ecs-a91135"
    },
    "cost-type": {
      "cost-mode": "array",
      "cost-metric": "ane-path"
    }
  },
  "endpoint-cost-map": {
    "ipv4:192.0.2.1": {
      "ipv4:192.0.2.2": [ "ane:1" ]
    },
    "ipv4:192.0.2.3": {
      "ipv4:192.0.2.2": [ "ane:1" ],
      "ipv4:192.0.2.4": [ "ane:1" ]
    }
  }
}

```

Since the bandwidth for all links is 100 Mbps, it can be easily concluded that only link5 can potentially become a bottleneck with the given client constraints. Thus, only one abstract network element is returned.

In this example, the abstract network element property map is not attached so the client SHOULD send another request to fetch the details about the abstract network elements.

```

POST /default-prop-map HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json, application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-propmapparams+json

```

```

{
  "query-id": "query-ecs-a91135",
  "entities": [ "ane:1" ],
  "properties": [ "availbw" ]
}

```

```
HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: application/alto-propmap+json
```

```
{
  "property-map": {
    "ane:1": { "availbw": 100 }
  }
}
```

## 8.5. Compatibility

The client constraint map is fully compatible with the path vector extension. For other extensions such as multi-cost [I-D.ietf-alto-multi-cost] and cost calendar [I-D.ietf-alto-cost-calendar], the input parameters **MUST** still follow the definition of "client-constraint-map" but can adjust the requirements for other fields.

Since the client constraint map extension is fully compatible with the path vector extension, it does not alter the compatibility with other extensions such as multi-cost and cost calendar.

## 9. Security Considerations

This document does not introduce any privacy or security issue on ALTO servers not already present in the base ALTO protocol or in the path vector extension.

The algorithms specified in this document can even help protect the privacy of network providers by conducting irreversible transformations on the original path vector.

The client constraint map extension defined in Section 8.3 can potentially leak client-side information to ALTO servers. ALTO client implementations **MUST** take information security into consideration when using this extension, for example, only activating this extension when the ALTO server is considered a trusted party.

ALTO clients can also obfuscate the information contained in a request, for example, providing larger values than actual upper bounds. Such obfuscation will not affect the correctness of the response, but can potentially affect the reduction effect of client constraint map.



## 10. IANA Considerations

This document does not define any new media type or introduce any new IANA consideration.

## 11. Acknowledgments

The authors would like to thank Dr. Qiao Xiang, Mr. Jingxuan Zhang (Tongji University), Prof. Jun Bi (Tsinghua University) and Dr. Andreas Voellmy (Yale University) for their early engagement and discussions.

## 12. References

### 12.1. 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, <<http://www.rfc-editor.org/info/rfc2119>>.

### 12.2. Informative References

- [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-02 (work in progress), April 2016.
- [I-D.ietf-alto-multi-cost]  
Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost ALTO", draft-ietf-alto-multi-cost-10 (work in progress), April 2017.
- [I-D.ietf-alto-path-vector]  
Bernstein, G., Chen, S., Gao, K., Lee, Y., Roome, W., Scharf, M., Yang, Y., and J. Zhang, "ALTO Extension: Path Vector Cost Mode", draft-ietf-alto-path-vector-00 (work in progress), May 2017.

[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, <<http://www.rfc-editor.org/info/rfc7285>>.

#### Authors' Addresses

Kai Gao  
Tsinghua University  
30 Shuangqinglu Street  
Beijing 100084  
China

Email: [gaok12@mails.tsinghua.edu.cn](mailto:gaok12@mails.tsinghua.edu.cn)

Xin (Tony) Wang  
Tongji University  
4800 CaoAn Road  
Shanghai 210000  
China

Email: [xinwang2014@hotmail.com](mailto:xinwang2014@hotmail.com)

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

Email: [qiao.xiang@cs.yale.edu](mailto:qiao.xiang@cs.yale.edu)

Chen Gu  
Tongji University  
4800 CaoAn Road  
Shanghai 210000  
China

Email: [gc19931011jy@gmail.com](mailto:gc19931011jy@gmail.com)

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

Email: [yry@cs.yale.edu](mailto:yry@cs.yale.edu)

G. Robert Chen  
Huawei  
Nanjing  
China

Email: [chenguohai@huawei.com](mailto:chenguohai@huawei.com)

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 4, 2018

S. Randriamasy  
Nokia Bell Labs  
R. Yang  
Yale University  
Q. Wu  
Huawei  
L. Deng  
China Mobile  
N. Schwan  
Thales Deutschland  
July 3, 2017

ALTO Cost Calendar  
draft-ietf-alto-cost-calendar-02

Abstract

The goal of Application-Layer Traffic Optimization (ALTO) is to bridge the gap between network and applications by provisioning network related information in order to allow applications to make network informed decisions. The present draft extends the ALTO cost information so as to broaden the decision possibilities of applications to not only decide 'where' to connect to, but also 'when'. This is useful to applications that need to schedule their data transfers and connections and have a degree of freedom to do so. ALTO guidance to schedule application traffic can also efficiently help for load balancing and resources efficiency. Besides, the ALTO Cost Calendar allows to schedule the ALTO requests themselves and thus to save a number of ALTO transactions.

This draft proposes new capabilities and attributes on filtered cost maps and endpoint costs enabling an ALTO Server to provide "Cost Calendars". These capabilities are applicable to time-sensitive ALTO metrics. With ALTO Cost Calendars, an ALTO Server exposes ALTO Cost Values in JSON arrays where each value corresponds to a given time interval. The time intervals as well as other Calendar attributes are specified in the IRD and ALTO Server responses.

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 4, 2018.

## Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Overview of ALTO Cost Calendars . . . . .	4
2.1. ALTO Cost Calendar information features . . . . .	5
2.2. ALTO Calendar design characteristics . . . . .	6
2.2.1. ALTO Cost Calendar for all cost modes . . . . .	6
2.2.2. Compatibility with legacy ALTO Clients . . . . .	7
3. ALTO Calendar specification: IRD extensions . . . . .	7
3.1. Calendar attributes in the IRD resources capabilities . .	7
3.2. Calendars in a delegate IRD . . . . .	9
3.3. Example IRD with ALTO Cost Calendars . . . . .	9
4. ALTO Calendar specification: Service Information Resources .	12
4.1. Calendar extensions for Filtered Cost Maps . . . . .	12
4.1.1. Calendar extensions in Filtered cost map requests . .	13
4.1.2. Calendar extensions in Filtered Cost map responses .	13

4.1.3. Use case and example: FCM with a bandwidth Calendar .	15
4.2. Calendar extensions in the Endpoint Cost Map Service . .	17
4.2.1. Calendar specific input in Endpoint cost map requests . . . . .	17
4.2.2. Calendar attributes in the Endpoint Cost Map response	17
4.2.3. Use case and example: ECS with a routingcost Calendar	18
4.2.4. Use case and example: ECS with a multi-cost calendar for routingcost and latency . . . . .	21
5. IANA Considerations . . . . .	23
5.1. Information for IANA on proposed Cost Types . . . . .	23
5.2. Information for IANA on proposed Endpoint Properties . .	23
6. Acknowledgements . . . . .	23
7. References . . . . .	23
7.1. Normative References . . . . .	23
7.2. Informative References . . . . .	24
Authors' Addresses . . . . .	24

## 1. Introduction

IETF is currently standardizing the ALTO protocol which aims at providing guidance to overlay applications needing to select one or several hosts from a set of candidates able to provide a desired resource. This guidance is based on parameters that affect performance and efficiency of the data transmission between the hosts such as the topological distance. The goal of ALTO is to improve the Quality of Experience (QoE) in the application while optimizing resource usage in the underlying network infrastructure.

The ALTO protocol in [RFC7285] specifies a Network Map which defines groupings of endpoints in provider-defined network regions (called PIDs). The Cost Map Service, Endpoint Cost Service (ECS) and Endpoint Ranking Service then provide ISP-defined costs and rankings for connections among the specified endpoints and PIDs and thus incentives for application clients to connect to ISP preferred locations, e.g. to reduce their costs. ALTO intentionally avoids provisioning realtime information as explained in the ALTO Problem Statement [RFC5693] and ALTO Requirements [RFC5693]. Thus the current Cost Map and Endpoint Cost Service are providing, for a given Cost Type, exactly one path cost value. Applications have to query one of these two services to retrieve the currently valid cost values. They therefore need to plan their ALTO information requests according to their own estimation of the frequency of cost value change.

With [RFC7285], an ALTO client should interpret the returned costs as those at the query moment. However, Network costs can fluctuate, e.g. due to diurnal patterns of traffic demand or planned events such as network maintenance, holidays or highly publicized events. Providing network costs for only the current time thus may not be

sufficient, in particular for applications that can schedule their traffic in a span of time, for example by deferring backup to night during traffic trough.

In case the ALTO Cost value changes are predicable over a certain period of time and the application does not require immediate data transfer, it can save time to get the whole set of cost values over this period in one single ALTO response. Using this set to schedule data transfers allows optimising the network resources usage and QoE. ALTO Clients and Servers can also minimize their workload by reducing and accordingly scheduling their data exchanges.

This document extends RFC7285 to allow an ALTO server to provide network costs for a given duration of time. A sequence of network costs across a time span for a given pair of network locations is named an "ALTO Cost Calendar". The Filtered Cost Map Service and Endpoint Cost Service are extended to provide Cost Calendars. In addition to this functional ALTO enhancement, we expect to further gain on storage and on the wire data exchange by gathering multiple Cost Values for one Cost Type into one single ALTO Server response.

In this draft an "ALTO Cost Calendar" is specified by information resources capabilities that are applicable to time-sensitive ALTO metrics. An ALTO Cost Calendar exposes ALTO Cost Values in JSON arrays where each value corresponds to a given time interval. The time intervals as well as other Calendar attributes are specified in the IRD and in the Server response to allow the ALTO Client to interpret the received ALTO values. Last, the proposed extensions for ALTO Calendars are applicable to any Cost Mode and they ensure backwards compatibility with legacy ALTO clients.

In the rest of this document, Section 2 provides the design characteristics. Sections 3 and 4 define the formal specifications for the IRD and the information resources. Section 5 provides non-normative use cases to illustrate the usage of cost calendars. IANA considerations and security considerations will be completed in further versions.

## 2. Overview of ALTO Cost Calendars

An ALTO Cost calendar provided by the ALTO Server provides 2 information items:

- o an array of values for a given metric, where each value corresponds to a time interval, where the value array can sometimes be a cyclic pattern that repeats a certain number of times.

- o attributes describing the time scope of the calendar such as the size and number of the intervals and the date of the starting point of the calendar, allowing an ALTO Client to properly interpret the values.

An ALTO Cost Calendar can be used like a "time table" to figure out the best time to schedule data transfers and also to proactively manage application traffic given predictable events such as flash crowds, traffic intensive holidays and network maintenance. It may be viewed as a synthetic abstraction of real measurements that can be historic or be a prediction for upcoming time periods.

Most likely, the ALTO Cost Calendar would be used for the Endpoint Cost Service, assuming that a limited set of feasible Endpoints for a non-real time application is already identified, that they do not need to be accessed immediately and that their access can be scheduled within a given time period. The Filtered Cost Map service is also applicable as long as the size of the Map allows it.

## 2.1. ALTO Cost Calendar information features

The Calendar attributes are provided in the IRD and in ALTO Server responses. The IRD announces attributes with dateless values in its information resources capabilities, where as attributes with time dependent values are provided in the "meta" of Server responses. The ALTO Cost Calendar attributes provide the following information:

- o attributes to interpret the time scope of the Calendar value array:
  - \* generic time zone,
  - \* applicable time interval for each calendar value: combining numbers and time units to reflect for example: 1 hour, 2 minutes, 10 seconds, 1 week, 1 month,
  - \* duration of the Calendar: e.g. the number of intervals provided in the calendar.
- o "calendar-start-date": specifying when the calendar starts, that is to which date the first value of the cost calendar is applicable.
- o "repeated": an optional attribute indicating for how many iterations the provided calendar will have the same values. The server may use it to allow the client to schedule its next request and thus save its own workload by avoiding to process useless requests.



## 2.2. ALTO Calendar design characteristics

The protocol extension placeholders for an ALTO Calendar are: the IRD, the ALTO requests and responses for Cost calendars.

Extensions are designed to be light and ensure backwards compatibility with base protocol ALTO Clients and with other extensions. It uses section 8.3.7 "Parsing of Unknown Fields" of RFC7285 that writes: "Extensions may include additional fields within JSON objects defined in this document. ALTO implementations MUST ignore unknown fields when processing ALTO messages."

The calendar-specific capabilities are integrated in the information resources of the IRD and in the "meta" member of ALTO responses to Cost Calendars requests. A calendar and its capabilities are associated with a given information resource and within this information resource with a given cost type. This design has several advantages:

- o it does not introduce a new mode,
- o it does not introduce new media types,
- o it allows an ALTO Server to offer calendar capabilities on a cost type, with attributes values adapted to each information resource.

The Applicable Calendared information resources are:

- o the Filtered Cost Map,
- o the Endpoint Cost Map.

The ALTO Server can choose in which frequency it provides cost Calendars to ALTO Clients. It may either provide calendar updates starting at the request date, or carefully schedule its updates so as to take profit from a potential repetition/periodicity of calendar values.

### 2.2.1. ALTO Cost Calendar for all cost modes

Calendars are well-suited for values encoded in the 'numerical' mode. However, Calendars can also represent any metric considered as time-varying by an ALTO Server. For example, types of Cost values such as JSONBool can also be expressed as calendars, as states may be "true" or "false" depending on given time periods or likewise, values represented by strings, such as "medium", "high", "low", "blue", "open" .

Note also that a Calendar is applicable as well to time-varying metrics provided in the 'ordinal' mode, if these values are time-varying and their update is carefully managed by the ALTO Server.

#### 2.2.2. Compatibility with legacy ALTO Clients

The ALTO protocol extensions for Cost Calendars have been defined so as to ensure that Calendar capable ALTO Servers can provide legacy ALTO Clients with legacy information resources as well. That is a legacy ALTO Client can request resources and receive responses as specified in RFC7285.

A Calendar-aware ALTO Server MUST implement the base protocol specified in RFC7285.

When a metric is available as a calendar, it MUST be available as a single value as well.

For compatibility with legacy ALTO Clients specified in RFC7285, calendared information resources are not applicable for full Cost Maps for the following reason: a legacy ALTO client would receive a Calendared Cost Map via an HTTP 'GET' command. As specified in section 8.3.7 of RFC7285, it will ignore the Calendar Attributes indicated in the "meta" of the responses. Therefore, lacking information on calendar attributes, it will not be able to correctly interpret and process the values of the received array of calendar cost values.

Therefore, calendared information resources MUST be requested via the Filtered Cost Map Service or the Endpoint Cost Service, using a POST method.

### 3. ALTO Calendar specification: IRD extensions

The Calendar attributes in the IRD information resources capabilities carry constant dateless values. A calendar is associated with an information resource rather than a cost type. For example, a Server can provide a "routingcost" calendar for the Filtered Cost Map Service at a granularity of one day and a "routingcost" calendar for the Endpoint Cost service at a finer granularity but for a limited number of endpoints.

#### 3.1. Calendar attributes in the IRD resources capabilities

When for an applicable resource, an ALTO Server provides a Cost Calendar for a given Cost Type, it MUST indicate this in the IRD capabilities of this resource, by an object of type

'CalendarAttributes', associated with this Cost Type and specified below.

The capabilities of a Calendar aware information resource entry have a member named "calendar-attributes" which is an array of objects of type CalendarAttributes. It is necessary to use an array because of resources such as Filtered Cost Map and Endpoint Cost Map, for which the member "cost-type-names" is an array of 1 or more values.

A member "calendar-attributes" MUST appear only once for each applicable cost type name of a resource entry. If "calendar-attributes" are specified several times for a same "cost-type-name" in the capabilities of a resource entry, the ALTO client SHOULD ignore any calendar capabilities on this "cost-type-name" for this resource entry.

```
CalendarAttributes calendar-attributes <1..*>;
```

```
object{
  JSONString    cost-type-names <1..*>;
  JSONString    time-interval-size;
  JSONNumber    number-of-intervals;
} CalendarAttributes;
```

- o "cost-type-name":

- \* An array of one or more elements indicating the cost-type-names in the IRD entry to which the capabilities apply.

- o "time-interval-size":

- \* is the duration of an ALTO calendar time interval, expressed as a time unit appended to the number of these units. The time unit, ranges from "second" to "year". The number is encoded with an integer. Example values are: "5 minute" , "2 hour", meaning that each calendar value applies on a time interval that lasts respectively 5 minutes and 2 hours.

- o "number-of-intervals":

- \* the integer number of values of the cost calendar array, at least equal to 1.

- Attribute "cost-type-name" , if used, provides a better readability to the calendar attributes specified in the IRD and avoids confusion with calendar attributes of other cost-types.

- Multiplying Attributes 'time-interval-size' and 'number-of-intervals' provides the duration of the provided calendar. For example an ALTO Server may provide a calendar for ALTO values changing every 'time-interval-size' equal to 5 minutes. If 'number-of-intervals' has the value 12, then the duration of the provided calendar is "1 hour".

### 3.2. Calendars in a delegate IRD

One option to clarify IRD resources is that a "root" ALTO Server implementing base protocol resources delegates "specialized" information resources such as the ones providing Cost Calendars to another ALTO Server running in a subdomain specified with its URI in the "root" ALTO Server. This option is described in Section 9.2.4 "Delegation using IRDs" of RFC7285.

This document provides an example, where a "root" ALTO Server runs in a domain called "alto.example.com". It delegates the announcement of Calendars capabilities to an ALTO Server running in a subdomain called "custom.alto.example.com". The location of the "delegate Calendar IRD" is assumed to be indicated in the "root" IRD by the resource entry: "custom-calendared-resources".

Another advantage is that some Cost Types for some resources may be more advantageous as Cost Calendars and it makes few sense to get them as a single value. For example, Cost Types with predictable and frequently changing values, calendared in short time intervals such as a minute.

### 3.3. Example IRD with ALTO Cost Calendars

The cost types in this example are either specified in the base ALTO protocol or may be proposed in other drafts see [draft-ietf-alto-performance-metrics]. In this example, the available cost metrics are indicated in the "meta" field by cost type names "num-routingcost", "num-latency", "num-pathbandwidth" and "string-quality-status". Metrics "routingcost", 'latency' and 'bandwidthscore' are available in the "numerical" Cost Mode. Metric "quality-status" is available in the "string" Cost Mode.

The example IRD includes 2 particular URIs providing calendars:

- o "http://custom.alto.example.com/calendar/costmap/filtered": a filtered cost map in which calendar capabilities are indicated for cost type names: "num-routingcost", "num-pathbandwidth" and "string-service-status",

- o "http://custom.alto.example.com/calendar/endpointcost/lookup": an endpoint cost map in which in which calendar capabilities are indicated for cost type names: "num-routingcost", "num-latency", "num-pathbandwidth", "string-service-status".

The design of the Calendar capabilities allows that some calendars on a cost type name are available in several information resources with different Calendar Attributes. This is the case for calendars on "num-routingcost", "num-pathbandwidth" and "string-service-status", available in both the Filtered Cost map and Endpoint Cost map service, but with different time interval sizes for "num-pathbandwidth" and "string-service-status".

```
GET /calendars-directory HTTP/1.1
Host: custom.alto.example.com
Accept: application/alto-directory+json,application/alto-error+json
-----
```

```
HTTP/1.1 200 OK
Content-Length: [TODO]
Content-Type: application/alto-directory+json
```

```
{
  "meta" : {
    "cost-types": {
      "num-routingcost": {
        "cost-mode" : "numerical",
        "cost-metric" : "routingcost"
      },
      "num-latency": {
        "cost-mode" : "numerical",
        "cost-metric": "latency"
      },
      "num-pathbandwidth": {
        "cost-mode" : "numerical",
        "cost-metric": "bandwidthscore",
      },
      "string-qual-status": {
        "cost-mode" : "string",
        "cost-metric": "quality-status",
      }
    },
    ... other meta ...
  },

  "resources" : {
    "filtered-cost-map-calendar" : {
      "uri" : "http://custom.alto.example.com/calendar/costmap/filtered",
      "media-type" : "application/alto-costmap+json",
```

```

    "accepts" : "application/alto-costmapfilter+json",
    "capabilities" : {
      "cost-constraints" : true,
      "cost-type-names" : [ "num-routingcost", "num-pathbandwidth",
                           "string-service-status" ],
      "calendar-attributes" : [
        { "cost-type-names" : [ "num-routingcost", "num-pathbandwidth" ]
          ,
            "time-interval-size" : "1 hour",
            "number-of-intervals" : 24
          },
        { "cost-type-names" : "string-service-status",
          "time-interval-size" : "30 minute",
          "number-of-intervals" : 48
        }
      ] // end calendar-attributes
    "uses": [ "my-default-network-map" ]
  }
},

"endpoint-cost-calendar-map" : {
  "uri" : "http://custom.alto.example.com/calendar/endpointcost/lookup",
  "media-types" : [ "application/alto-endpointcost+json" ],
  "accepts" : [ "application/alto-endpointcostparams+json" ],
  "capabilities" : {
    "cost-constraints" : true,
    "cost-type-names" : [ "num-routingcost", "num-latency",
                        "num-pathbandwidth", "string-service-status" ],
    "calendar-attributes" : [
      { "cost-type-names" : "num-routingcost",
        "time-interval-size" : "1 hour",
        "number-of-intervals" : 24
      },
      { "cost-type-names" : "latency",
        "time-interval-size" : "5 minute",
        "number-of-intervals" : 12
      },
      { "cost-type-names" : "num-pathbandwidth",
        "time-interval-size" : "1 minute",
        "number-of-intervals" : 60
      },
      { "cost-type-names" : "string-service-status",
        "time-interval-size" : "2 minute",
        "number-of-intervals" : 30
      }
    ]
    "uses": [ "my-default-network-map" ]
  } // ECM capab
} //info resource N

```

```
} // ressources
```

In this example IRD, for the filtered cost map service:

- o the Calendar for 'num-routingcost' and 'num-pathbandwidth' is an array of 24 values each provided on a time interval lasting 1 hour.
- o the Calendar for "string-service-status": "is an array of 48 values each provided on a time interval lasting 30 minutes.

For the endpoint cost map service:

- o the Calendar for 'num-routingcost': is an array of 24 values each provided on a time interval lasting 1 hour.
- o the Calendar for 'latency': is an array of 12 values each provided on a time interval lasting 5 minutes.
- o the Calendar for 'num-pathbandwidth': is an array of 60 values each provided on a time interval lasting 1 minute.
- o the Calendar for "string-service-status": "is an array of 30 values each provided on a time interval lasting 2 minutes.

#### 4. ALTO Calendar specification: Service Information Resources

This section documents the individual information resources defined to provide the Calendared information services defined in this document.

The reference time zone for the provided time values is GMT because the option chosen to express the time format is the HTTP header fields format:

Date: Tue, 15 Nov 2014 08:12:31 GMT

##### 4.1. Calendar extensions for Filtered Cost Maps

A legacy ALTO client requests and gets filtered cost map responses as specified in RFC7285.

#### 4.1.1.1. Calendar extensions in Filtered cost map requests

The input parameters of a "legacy" request for a filtered cost map, defined by object ReqFilteredCostMap in section 11.3.2 of RFC7285, are augmented with one additional member.

A Calendar-aware ALTO client requesting a Calendar on a given Cost Type for a Filtered Cost Map resource having Calendar capabilities MUST add the following field to its input parameters:

```
JSONBoolean    calendared<1..*>;
```

This field is an array of 1 to N boolean values, where N is the number of requested metrics. Each boolean value indicates whether or not the ALTO Server should provide the values for this Cost Type as a calendar. The array MUST contain exactly N boolean values, otherwise the server returns an error.

This field MUST NOT be specified if member "calendar-attributes" is not present for this information resource.

If this field is not present, it MUST be assumed to have only values equal to "false".

A Calendar-aware ALTO client supporting single cost type values, as specified in RFC7285, MUST provide an array of 1 element:

```
"calendared" : [true];
```

A Calendar-aware ALTO client that is also Multi-Cost aware MUST provide an array of N values set to "true" or "false", depending whether it wants the applicable Cost Type values as a single or calendared value.

#### 4.1.1.2. Calendar extensions in Filtered Cost map responses

The calendared costs are JSONArrays instead of JSONNumbers for the legacy ALTO implementation. All arrays have a number of values equal to 'number-of-intervals'.

The "meta" field of a Calendared Filtered Cost map response MUST include at least:

- o if the ALTO Client supports cost values for one Cost Type at a time only: the "meta" fields specified in RFC 7285 for these information service responses:

- \* "dependent-vtags ",



- \* "cost-type" field.
- o if the ALTO Client supports cost values for several Cost Types at a time, as specified in [draft-ietf-alto-multi-cost] : the "meta" fields specified in [draft-ietf-alto-multi-cost] for these information service responses:
  - \* "dependent-vtags ",
  - \* "cost-type" field with value set to '{}', for backwards compatibility with RFC7285.
  - \* "multi-cost-types" field.
- o If the client request does not provide member "calendared" or if it provides it with a value equal to 'false', for all the requested Cost Types, then the ALTO Server response is exactly as specified in RFC 7285 [ID-alto-protocol] and [draft-ietf-alto-multi-cost].
- o If the value of member "calendared" is equal to 'false' for a given requested Cost Type, the ALTO Server must return, for these Cost Types, a single cost value as specified in RFC 7285.

In addition, the "meta" field of a Calendared Filtered Cost map response MUST include the member "calendar-response-attributes" for the requested information resource, together with the values provided by the ALTO Server for these attributes. This member is an array of objects of type "CalendarResponseAttributes", defined as follows:

CalendarResponseAttributes calendar-response-attributes <1..\*>;

```
object{
  JSONString    cost-type-names;
  JSONString    calendar-start-time;
  JSONString    time-interval-size;
  JSONNumber    number-of-intervals;
  [JSONNumber   repeated;]          [OPTIONAL]
} CalendarResponseAttributes;
```

Object CalendarResponseAttributes has the following attributes:

- o "cost-type-names": member indicating the cost-type-names to which the capabilities apply.
- o "calendar-start-time": indicates the date at which the first value of the calendar applies. By default, the value provided for the

"calendar-start-time" attribute SHOULD be no later than the request date.

- o "time-interval-size": as specified in section "Calendar attributes in the IRD resources capabilities",
- o "number-of-intervals": as specified in section "Calendar attributes in the IRD resources capabilities",
- o "repeated": is an optional field provided for Calendars. It is an integer N greater or equal to '1' that indicates how many iterations of the calendar value array starting at the date indicated by "calendar-start-time" have the same values. The number N includes the provided iteration.

Using the member "repeated" helps minimizing on the wire data exchange: by providing it, an ALTO Server will avoid unnecessary processing of requests for Calendars with unchanged values while it allows ALTO Clients to save their resources as well.

For example: if the "calendar-start-time" member has value "Mon, 30 Jun 2014 at 00:00:00 GMT" and if the value of member "repeated" is equal to 4, it means that the calendar values are the same values on Monday, Tuesday, Wednesday and Thursday. The ALTO Client thus may use the same calendar for the next 4 duration periods following "calendar-start-time".

#### 4.1.3. Use case and example: FCM with a bandwidth Calendar

An example of non-real time information that can be provisioned in a 'calendar' is the expected path bandwidth. While the transmission rate can be measured in real time by end systems, the operator of a data center is in the position of formulating preferences for given paths, at given time periods for example to avoid traffic peaks due to diurnal usage patterns. In this example, we assume that an ALTO Client requests a bandwidth calendar as specified in the IRD to schedule its bulk data transfers as described in the use cases.

In the example IRD, calendars for cost type name "num-pathbandwidth" are available for the information resources: "filtered-cost-calendar-map" and "endpoint-cost-calendar-map". The ALTO Client requests a calendar for "num-pathbandwidth" via a POST request for a filtered cost map.

We suppose in this example that the ALTO Client sends its request on Tuesday July 1st 2014 at 13:15

```
POST /calendar/costmap/filtered HTTP/1.1
Host: alto.example.com
Content-Length: [TODO]
Content-Type: application/alto-costmapfilter+json
Accept: application/alto-costmap+json,application/alto-error+json

{
  "cost-type" : { "cost-mode" : "numerical", "cost-metric" : "bandwidthscore" },
  "calendared" : [true],

  "pids" : {
    "srcs" : [ "PID1", "PID2" ],
    "dsts" : [ "PID1", "PID2", "PID3" ]
  }
}

HTTP/1.1 200 OK
Content-Length: [TODO]
Content-Type: application/alto-costmap+json

{
  "meta" : {
    "dependent-vtags" : [...],
    "cost-type" : { "cost-mode" : "numerical", "cost-metric" : "bandwidthscore"
  },
  "calendar-response-attributes" : [
    "calendar-start-time" : Tue, 1 Jul 2014 13:00:00 GMT,
    "time-interval-size" : "2 hour",
    "numb-intervals" : 12
  ]
},

  "cost-map" : {
    "PID1" : { "PID1": [v1,v2, ... v12],
               "PID2": [v1,v2, ... v12],
               "PID3": [v1,v2, ... v12] },
    "PID2" : { "PID1": [v1,v2, ... v12],
               "PID2": [v1,v2, ... v12],
               "PID3": [v1,v2, ... v12] }
  }
}
```

#### 4.2. Calendar extensions in the Endpoint Cost Map Service

This document extends the Endpoint Cost Service, as defined in {11.5.1} of [RFC7285], by adding new input parameters and capabilities, and by returning JSONArrays instead of JSONNumbers as the cost values. The media type {11.5.1.1} and HTTP method {11.5.1.2} are unchanged.

##### 4.2.1. Calendar specific input in Endpoint cost map requests

The extensions to the requests for calendared Endpoint Cost Maps are the same as for the Filtered Cost Map Service, specified in section XXXX of this draft.

The ReqEndpointCostMap object for a Calendared ECM request will have the following format:

```
object {  
  [CostType cost-type;]  
  [CostType multi-cost-types<1..*>;]  
  [JSONBoolean    calendared<1..*>;]  
  EndpointFilter endpoints;  
} ReqEndpointCostMap;  
  
object {  
  [TypedEndpointAddr srcs<0..*>;]  
  [TypedEndpointAddr dsts<0..*>;]  
} EndpointFilter;
```

##### 4.2.2. Calendar attributes in the Endpoint Cost Map response

The "meta" field of a Calendared Endpoint Cost map response MUST include at least:

- o if the ALTO Client supports cost values for one Cost Type at a time only: the "meta" fields specified in {11.5.1.6} of RFC 7285 for the Endpoint Cost response:
  - \* "cost-type" field.
- o if the ALTO Client supports cost values for several Cost Types at a time, as specified in [draft-ietf-alto-multi-cost] : the "meta" fields specified in [draft-ietf-alto-multi-cost] for the the Endpoint Cost response:
  - \* "cost-type" field with value set to '{}', for backwards compatibility with RFC7285.

\* "multi-cost-types" field.

If the client request does not provide member "calendared" or if it provides it with a value equal to 'false', for all the requested Cost Types, then the ALTO Server response is exactly as specified in RFC 7285 [ID-alto-protocol] and [draft-ietf-alto-multi-cost].

If the ALTO client provides member "calendared" in the input parameters with a value equal to 'true' for given requested Cost Types, the "meta" member of a Calendared Endpoint Cost Map response MUST include, for these Cost Types, the same additional member "calendar-response-attributes", as specified for the Filtered Cost Map Service. The Server response is thus changed as follows, w.r.t RFC 7285 and [draft-ietf-alto-multi-cost]:

- o the "meta" member has one additional field "CalendarResponseAttributes", as specified for the Filtered Cost Map Service,
- o the calendared costs are JSONArrays instead of JSONNumbers for the legacy ALTO implementation. All arrays have a number of values equal to 'number-of-intervals'.

If the value of member "calendared" is equal to 'false' for a given requested Cost Type, the ALTO Server must return, for these Cost Types, a single cost value as specified in RFC 7285.

#### 4.2.3. Use case and example: ECS with a routingcost Calendar

Let us assume an Application Client is located in an end system with limited resources and having an access to the network that is either intermittent or provides an acceptable quality in limited but predictable time periods. Therefore, it needs to both schedule its resources greedy networking activities and its ALTO transactions.

The Application Client has the choice to trade content or resources with a set of Endpoints and needs to decide with which one it will connect and at what time. For instance, the Endpoints are spread in different time-zones, or have intermittent access. In this example, the 'routingcost' is assumed to be time-varying, with values provided as ALTO Calendars.

The ALTO Client associated with the Application Client queries an ALTO Calendar on 'routingcost' and will get the Calendar covering the 24 hours time period "containing" the date and time of the ALTO client request.

For Cost Type 'num-routingcost', the solicited ALTO Server has defined 3 different daily patterns each represented by a Calendar, to cover the week of Monday June 30th at 00:00 to Sunday July 6th 23:59:

- C1 for Monday, Tuesday, Wednesday, Thursday, (week days)
- C2 for Saturday, Sunday, (week end)
- C3 for Friday (maintenance outage on July 4, 2014 from 02:00:00 GMT to 04:00:00 GMT, or big holiday such as New Year evening).

In the following example, the ALTO Client sends its request on Tuesday July 1st 2014 at 13:15.

```
POST /calendar/endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: [TODO]
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

{
  "cost-type" : {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
  "calendared" : [true],
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv4:203.0.113.45",
      "ipv6:2000::1:2345:6789:abcd"
    ]
  }
}

HTTP/1.1 200 OK
Content-Length: [TODO]
Content-Type: application/alto-endpointcost+json

{
  "meta" : {
    "cost-type" : {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
    "calendar-response-attributes" : [
      { "calendar-start-time" : Mon, 30 Jun 2014 00:00:00 GMT,
        "time-interval-size" : "1 hour",
        "numb-intervals" : 24,
        "repeated": 4 }
    ],
  } // end meta

  "endpoint-cost-map" : {
    "ipv4:192.0.2.2": {
      "ipv4:192.0.2.89" : [v1, v2, ... v24],
      "ipv4:198.51.100.34" : [v1, v2, ... v24],
      "ipv4:203.0.113.45" : [v1, v2, ... v24],
      "ipv6:2000::1:2345:6789:abcd" : [v1, v2, ... v24]
    }
  }
}
```

When the Client gets the Calendar for "routingcost", it sees that the "calendar-start-time" is Monday at 00h00 GMT and member "repeated" is equal to '4'. It understands that the provided values are valid until Thursday included and will only need to get a Calendar update on Friday.

#### 4.2.4. Use case and example: ECS with a multi-cost calendar for routingcost and latency

In this example, it is assumed that the ALTO Server implements multi-cost capabilities, as specified in [draft-ietf-alto-multi-cost]. That is, an ALTO client can request and receive values for several cost types in one single transaction. An illustrating use case is a path selection done on the basis of 2 metrics: routing cost and latency.

As in the previous example, the IRD indicates that the ALTO Server provides "routingcost" Calendars in terms of 24 time intervals of 1 hour each.

For metric "latency", the IRD indicates that the ALTO Server provides Calendars in terms of 12 time intervals values lasting each 5 minutes.

In the following example transaction, the ALTO Client sends its request on Tuesday July 1st 2014 at 13:15.

```
POST calendar/endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: [TODO]
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

{
  "cost-type" : {},
  "multi-cost-types" : [
    { "cost-mode" : "numerical", "cost-metric" : "routingcost" },
    { "cost-mode" : "numerical", "cost-metric" : "latency" }
  ],
  "calendared" : [true, true],
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv4:203.0.113.45",
      "ipv6:2000::1:2345:6789:abcd"
    ]
  }
}
```



```
}
}
```

HTTP/1.1 200 OK

Content-Length: [TODO]

Content-Type: application/alto-endpointcost+json

```
{
  "meta" : {
    "multi-cost-types" : [
      { "cost-mode" : "numerical", "cost-metric" : "routingcost" },
      { "cost-mode" : "numerical", "cost-metric" : "latency" }
    ],
    "calendar-response-attributes" : [
      { "cost-type-name" : "num-routingcost",
        "calendar-start-time" : "Mon, 30 Jun 2014 00:00:00 GMT",
        "time-interval-size" : "1 hour",
        "numb-intervals" : 24,
        "repeated": 4 },
      { "cost-type-name" : "num-latency",
        "calendar-start-time" : "Tue, 1 Jul 2014 13:00:00 GMT",
        "time-interval-size" : "5 minute",
        "numb-intervals" : 12 }
    ],
  } // end meta

  "endpoint-cost-map" : {
    "ipv4:192.0.2.2" : {
      "ipv4:192.0.2.89" : [[r1, r2, ... r24], [l1, l2, ... l12]],
      "ipv4:198.51.100.34" : [[r1, r2, ... r24], [l1, l2, ... l12]],
      "ipv4:203.0.113.45" : [[r1, r2, ... r24], [l1, l2, ... l12]],
      "ipv6:2000::1:2345:6789:abcd" : [[r1, r2, ... r24], [l1, l2, ... l12]]
    }
  }
}
```

When receiving the response, the client sees that the calendar values for 'routing cost' are repeated for 4 iterations. Therefore, in its next requests until the routing cost calendar is expected to change, the client will only need to request a calendar for "latency".

Without the ALTO Calendar extensions, the ALTO client would have no clue on the dynamicity of the metric value change and would spend needless time requesting values at an inappropriate pace. In addition, without the Multi-Cost ALTO capabilities, the ALTO client

would duplicate this waste of time as it would need to send one request per cost metric.

## 5. IANA Considerations

Information for the ALTO Endpoint property registry maintained by the IANA and related to the new Endpoints supported by the acting ALTO server. These definitions will be formulated according to the syntax defined in Section on "ALTO Endpoint Property Registry" of [RFC7285]

Information for the ALTO Cost Type Registry maintained by the IANA and related to the new Cost Types supported by the acting ALTO server. These definitions will be formulated according to the syntax defined in Section on "ALTO Cost Type Registry" of [RFC7285],

### 5.1. Information for IANA on proposed Cost Types

When a new ALTO Cost Type is defined, accepted by the ALTO working group and requests for IANA registration MUST include the following information, detailed in Section 11.2: Identifier, Intended Semantics, Security Considerations.

### 5.2. Information for IANA on proposed Endpoint Properties

Likewise, an ALTO Endpoint Property Registry could serve the same purposes as the ALTO Cost Type registry. Application to IANA registration for Endpoint Properties would follow a similar process.

## 6. Acknowledgements

Many thanks to Diego Lopez, He Peng, Haibin Song, Dawn Chan, Li Geng and Yichen Qian for fruitful discussions and review feedback.

## 7. References

### 7.1. 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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, DOI 10.17487/RFC5693, October 2009, <<http://www.rfc-editor.org/info/rfc5693>>.

## 7.2. Informative References

- [draft-ietf-alto-multi-cost]  
S. Randriamasy, W. Roome, N. Schwan, "Multi-Cost ALTO (work in progress), draft-ietf-alto-multi-cost", September 2016.
- [draft-ietf-alto-performance-metrics]  
Q. Wu, Y. Yang, Y. Lee, D. Dhody, S. Randriamasy, "ALTO Performance Cost Metrics (work in progress)", September 2016.
- [draft-yang-alto-topology]  
Y. Yang, "ALTO Topology Considerations (work in progress)", July 2013.
- [ID-alto-protocol]  
R. Alimi, R. Penno, Y. Yang, Eds., "ALTO Protocol, RFC 7285", September 2014.
- [RFC7285] R. Alimi, R. Yang, R. Penno, Eds., "ALTO Protocol", September 2014.
- [sdnrg] "Software Defined Network Research Group,  
<http://trac.tools.ietf.org/group/irtf/trac/wiki/sdnrg>".
- [slides-88-alto-5-topology]  
G. Bernstein, Y. Lee, Y. Yang, , "ALTO Topology Service: Use Cases, Requirements and Framework (presentation slides IETF88 ALTO WG session),  
<http://tools.ietf.org/agenda/88/slides/slides-88-alto-5.pdf>", November 2013.

## Authors' Addresses

Sabine Randriamasy  
Nokia Bell Labs  
Route de Villejust  
NOZAY 91460  
FRANCE

Email: [Sabine.Randriamasy@nokia-bell-labs.com](mailto:Sabine.Randriamasy@nokia-bell-labs.com)

Richard Yang  
Yale University  
51 Prospect st  
New Haven, CT 06520  
USA

Email: yry@cs.yale.edu

Qin Wu  
Huawei  
101 Software Avenue, Yuhua District  
Nanjing, Jiangsu 210012  
China

Email: sunseawq@huawei.com

Lingli Deng  
China Mobile  
China

Email: denglingli@chinamobile.com

Nico Schwan  
Thales Deutschland

Email: nico.schwan@thalesgroup.com

ALTO WG  
Internet-Draft  
Intended status: Standards Track  
Expires: September 21, 2020

W. Roome  
Nokia Bell Labs  
Y. Yang  
Yale University  
March 20, 2020

ALTO Incremental Updates Using Server-Sent Events (SSE)  
draft-ietf-alto-incr-update-sse-22

Abstract

The Application-Layer Traffic Optimization (ALTO) [RFC7285] protocol provides network related information, called network information resources, to client applications so that clients can make informed decisions in utilizing network resources. This document presents a mechanism to allow an ALTO server to push updates to ALTO clients, to achieve two benefits: (1) updates can be incremental, in that if only a small section of an information resource changes, the ALTO server can send just the changes; and (2) updates can be immediate, in that the ALTO server can send updates as soon as they are available.

Requirements Language

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 21, 2020.

## Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Terms . . . . .	4
3. Background . . . . .	6
3.1. Incremental Encoding: JSON Merge Patch . . . . .	6
3.1.1. JSON Merge Patch Encoding . . . . .	6
3.1.2. JSON Merge Patch ALTO Messages . . . . .	7
3.2. Incremental Encoding: JSON Patch . . . . .	10
3.2.1. JSON Patch Encoding . . . . .	11
3.2.2. JSON Patch ALTO Messages . . . . .	11
3.3. Multiplexing and Server Push: HTTP/2 . . . . .	13
3.4. Server Push: Server-Sent Event . . . . .	14
4. Overview of Approach and High-level Protocol Message Flow . .	15
4.1. Update Stream Service Message Flow . . . . .	16
4.2. Stream Control Service Message Flow . . . . .	17
4.3. Service Announcement and Management Message Flow . . . .	18
5. Update Messages: Data Update and Control Update Messages . .	19
5.1. Generic ALTO Update Message Structure . . . . .	19
5.2. ALTO Data Update Message . . . . .	19
5.3. ALTO Control Update Message . . . . .	21
6. Update Stream Service . . . . .	22
6.1. Media Type . . . . .	22
6.2. HTTP Method . . . . .	22
6.3. Capabilities . . . . .	22
6.4. Uses . . . . .	23
6.5. Request: Accept Input Parameters . . . . .	23
6.6. Response . . . . .	25
6.7. Additional Requirements on Update Stream Service . . . .	27
6.7.1. Event Sequence Requirements . . . . .	27
6.7.2. Cross-Stream Consistency Requirements . . . . .	27
6.7.3. Multipart Update Requirements . . . . .	28
6.8. Keep-Alive Messages . . . . .	28

7.	Stream Control Service . . . . .	29
7.1.	URI . . . . .	29
7.2.	Media Type . . . . .	30
7.3.	HTTP Method . . . . .	30
7.4.	IRD Capabilities & Uses . . . . .	30
7.5.	Request: Accept Input Parameters . . . . .	30
7.6.	Response . . . . .	31
8.	Examples . . . . .	32
8.1.	Example: IRD Announcing Update Stream Services . . . . .	32
8.2.	Example: Simple Network and Cost Map Updates . . . . .	35
8.3.	Example: Advanced Network and Cost Map Updates . . . . .	38
8.4.	Example: Endpoint Property Updates . . . . .	41
8.5.	Example: Multipart Message Updates . . . . .	45
9.	Operation and Processing Considerations . . . . .	47
9.1.	Considerations for Choosing Data Update Messages . . . . .	47
9.2.	Considerations for Client Processing Data Update Messages . . . . .	48
9.3.	Considerations for Updates to Filtered Cost Maps . . . . .	49
9.4.	Considerations for Updates to Ordinal Mode Costs . . . . .	50
9.5.	Considerations for SSE Text Formatting and Processing . . . . .	50
10.	Security Considerations . . . . .	51
10.1.	Update Stream Server: Denial-of-Service Attacks . . . . .	51
10.2.	ALTO Client: Update Overloading or Instability . . . . .	52
10.3.	Stream Control: Spoofed Control Requests and Information Breakdown . . . . .	52
11.	Requirements on Future ALTO Services to Use this Design . . . . .	52
12.	IANA Considerations . . . . .	53
12.1.	application/alto-updatestreamparams+json Media Type . . . . .	53
12.2.	application/alto-updatestreamcontrol+json Media Type . . . . .	54
13.	Contributors . . . . .	55
14.	Acknowledgments . . . . .	55
15.	Appendix: Design Decision: Not Allowing Stream Restart . . . . .	56
16.	References . . . . .	57
16.1.	Normative References . . . . .	57
16.2.	Informative References . . . . .	57
	Authors' Addresses . . . . .	58

## 1. Introduction

The Application-Layer Traffic Optimization (ALTO) [RFC7285] protocol provides network related information called network information resources to client applications so that clients may make informed decisions in utilizing network resources. For example, an ALTO server provides network and cost maps, where a network map partitions the set of endpoints into a manageable number of sets each defined by a Provider-Defined Identifier (PID), and a cost map provides directed costs between PIDs. Given network and cost maps, an ALTO client can obtain costs between endpoints by first using the network map to get the PID for each endpoint, and then using the cost map to get the

costs between those PIDs. Such costs can be used by the client to choose communicating endpoints with low network costs.

The ALTO protocol defines only an ALTO client pull model, without defining a mechanism to allow an ALTO client to obtain updates to network information resources, other than by periodically re-fetching them. In settings where an information resource may be large but only parts of it may change frequently (e.g., some entries of a cost map), complete re-fetching can be inefficient.

This document presents a mechanism to allow an ALTO server to push incremental updates to ALTO clients. Integrating server-push and incremental updates provides two benefits: (1) updates can be small, in that if only a small section of an information resource changes, the ALTO server can send just the changes; and (2) updates can be immediate, in that the ALTO server can send updates as soon as they are available.

While primarily intended to provide updates to GET-mode network and cost maps, the mechanism defined in this document can also provide updates to POST-mode ALTO services, such as the ALTO endpoint property and endpoint cost services. The mechanism can also support new ALTO services to be defined by future extensions, but a future service needs to satisfy requirements specified in Section 11.

The rest of this document is organized as follows. Section 3 gives background on the basic techniques used in this design: (1) JSON merge patch and JSON patch to allow incremental update; and (2) Server-Sent Events (SSE) [SSE] to allow server push. With the background, Section 4 gives a non-normative overview of the design. Section 5 defines individual messages in an update stream. Section 6 defines the update stream service; Section 7 defines the stream control service; Section 8 gives several examples to illustrate the two types of services. Section 9 describes operation and processing considerations by both ALTO servers and clients; Section 15 discusses a design feature that is not supported; Section 10 discusses security issues; Section 11 and Section 12 review the requirements for future ALTO services to use SSE and IANA considerations, respectively.

## 2. Terms

Besides the terminologies as defined in [RFC7285], this document also uses additional terminologies defined as follows:

**Update Stream:** A reliable, in-order HTTP/1.x compatible connection between an ALTO client and an ALTO server so that the server can push a sequence of update messages using [SSE] to the client.



**Update Stream Server:** This document refers to an ALTO server providing an update stream as an ALTO update stream server, or update stream server for short. Note that the ALTO server mentioned in this document refers to a general server that provides various kinds of services; it can be an update stream server or stream control server (see below); it can also be a server providing ALTO Information Resource Directory (IRD).

**Update Message:** A message that is either a data update message or a control update message.

**Data Update Message:** An update message that is for a single ALTO information resource and sent from the update stream server to the ALTO client when the resource changes. A data update message can be either a full-replacement message or an incremental-change message. Full replacement is a shorthand for a full-replacement message, and incremental change is a shorthand for an incremental-change message.

**Full Replacement:** A data update message for a resource that encodes the content of the resource in its original ALTO encoding.

**Incremental Change:** An data update message that specifies only the difference between the new content and the previous version. An incremental change can be encoded using either JSON merge patch or JSON patch in this document.

**Stream Control Service:** A service that provides an HTTP URI so that the ALTO client of an update stream can use it to send stream control requests to the ALTO server on the addition or removal of resources receiving update messages from the update stream. The ALTO server creates a new stream control resource for each update stream instance, assigns a unique URI to it, and sends the URI to the client as the first event in the stream. (Note that the Stream Control Service in ALTO has no association with the similarly named Stream Control Transmission Protocol [RFC4960].)

**Stream Control:** A shorthand for stream control service.

**Stream Control Server:** An ALTO server providing the stream control service.

**Substream-ID:** An ALTO client can assign a unique substream-id when requesting the addition of a resource receiving update messages from an update stream. The server puts the substream-id in each update event for that resource. Substream-id allows a client to use one update stream to receive updates to multiple requests for the same resource (i.e., with the same resource-id in an ALTO IRD), for example, for a POST-mode resource with different input parameters.

**Data-ID:** A subfield of the 'event' field of [SSE] to identify the ALTO data (object) to be updated. For an ALTO resource returning a multipart response, the data-id to identify the data (object) is the substream-id, in addition to the content-id of the object in the multipart response. The data-id of a single part response is just the substream-id.

**Control Update Message:** An update message for the update stream server to notify the ALTO client of related control information of the update stream. A control update message may be triggered by an internal event at the server, such as server overloading and hence the update stream server will no longer send updates for an information resource, or as a result of a client sending a request through the stream control service. The first message of an update stream is a control update message and provides the URI using which the ALTO client can send stream control requests to the stream control server.

### 3. Background

The design requires two basic techniques: encoding of incremental changes and server push. For incremental changes, existing techniques include JSON merge patch and JSON patch; this design uses both. For server push, existing techniques include HTTP/2 and [SSE]; this design adopts some design features of HTTP/2 but uses [SSE] as the basic server-push design. The rest of this section gives a non-normative summary of JSON merge patch, JSON patch, HTTP/2 and [SSE].

#### 3.1. Incremental Encoding: JSON Merge Patch

To avoid always sending complete data, a server needs mechanisms to encode incremental changes, and JSON merge patch is one mechanism. [RFC7396] defines the encoding of incremental changes (called JSON merge patch objects) to be used by the HTTP PATCH method [RFC5789]. This document adopts from [RFC7396] only the JSON merge patch object encoding and does not use the HTTP PATCH method, as the updates are sent as events, instead of HTTP methods; also the updates are server-to-client in the updates, and PATCH semantics is more for client-to-server. Below is a non-normative summary of JSON merge patch objects; see [RFC7396] for the normative definition.

##### 3.1.1. JSON Merge Patch Encoding

Informally, a JSON merge patch message consists of a JSON merge patch object (referred to as a patch in [RFC7396]), which defines how to transform one JSON value into another using a recursive merge patch algorithm. Specifically, the patch is computed by treating two JSON values (first one being the original, and the second being the

updated) as trees of nested JSON objects (dictionaries of name-value pairs), where the leaves are values (e.g., JSON arrays, strings, numbers) other than JSON objects and the path for each leaf is the sequence of keys leading to that leaf. When the second tree has a different value for a leaf at a path, or adds a new leaf, the patch has a leaf, at that path, with the new value. When a leaf in the first tree does not exist in the second tree, the JSON merge patch tree has a leaf with a JSON "null" value. Hence, in the patch, null as the value of a name/value pair will delete the element with "name" in the original JSON value. The patch does not have an entry for any leaf that has the same value in both versions. See the MergePatch pseudocode at the beginning of Section 2 of [RFC7396] for the formal specification of how to apply a given patch. As a result, if all leaf values are simple scalars, JSON merge patch is a quite efficient representation of incremental changes. It is less efficient when leaf values are arrays, because JSON merge patch replaces arrays in their entirety, even if only one entry changes.

### 3.1.2. JSON Merge Patch ALTO Messages

To provide both examples of JSON merge patch and a demonstration of the feasibility of applying JSON merge patch to ALTO, the sections below show the application of JSON merge patch to two key ALTO messages.

#### 3.1.2.1. JSON Merge Patch Network Map Messages

Section 11.2.1.6 of [RFC7285] defines the format of an ALTO network map message. Assume a simple example ALTO message sending an initial network map:

```

{
  "meta" : {
    "vtag": {
      "resource-id" : "my-network-map",
      "tag" : "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
    }
  },
  "network-map" : {
    "PID1" : {
      "ipv4" : [ "192.0.2.0/24", "198.51.100.0/25" ]
    },
    "PID2" : {
      "ipv4" : [ "198.51.100.128/25" ]
    },
    "PID3" : {
      "ipv4" : [ "0.0.0.0/0" ],
      "ipv6" : [ "::/0" ]
    }
  }
}

```

Consider the following JSON merge patch update message, which (1) adds an ipv4 prefix "203.0.113.0/25" and an ipv6 prefix "2001:db8:8000::/33" to "PID1", (2) deletes "PID2", and (3) assigns a new "tag" to the network map:

```

{
  "meta" : {
    "vtag" : {
      "tag" : "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
    }
  },
  "network-map": {
    "PID1" : {
      "ipv4" : [ "192.0.2.0/24", "198.51.100.0/25",
                  "203.0.113.0/25" ],
      "ipv6" : [ "2001:db8:8000::/33" ]
    },
    "PID2" : null
  }
}

```

Applying the JSON merge patch update to the initial network map is equivalent to the following ALTO network map:

```

{
  "meta" : {
    "vtag": {
      "resource-id" : "my-network-map",
      "tag" : "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
    }
  },
  "network-map" : {
    "PID1" : {
      "ipv4" : [ "192.0.2.0/24", "198.51.100.0/25",
                  "203.0.113.0/25" ],
      "ipv6" : [ "2001:db8:8000::/33" ]
    },
    "PID3" : {
      "ipv4" : [ "0.0.0.0/0" ],
      "ipv6" : [ "::/0" ]
    }
  }
}

```

### 3.1.2.2. JSON Merge Patch Cost Map Messages

Section 11.2.3.6 of [RFC7285] defines the format of an ALTO cost map message. Assume a simple example ALTO message for an initial cost map:

```

{
  "meta" : {
    "dependent-vtags" : [
      { "resource-id": "my-network-map",
        "tag": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
      }
    ],
    "cost-type" : {
      "cost-mode" : "numerical",
      "cost-metric": "routingcost"
    },
    "vtag": {
      "resource-id" : "my-cost-map",
      "tag" : "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
    }
  },
  "cost-map" : {
    "PID1": { "PID1": 1, "PID2": 5, "PID3": 10 },
    "PID2": { "PID1": 5, "PID2": 1, "PID3": 15 },
    "PID3": { "PID1": 20, "PID2": 15 }
  }
}

```

The following JSON merge patch message updates the example cost map so that (1) the "tag" field of the cost map is updated, (2) the cost of PID1->PID2 is 9 instead of 5, (3) the cost of PID3->PID1 is no longer available, and (4) the cost of PID3->PID3 is defined as 1.

```
{
  "meta" : {
    "vtag": {
      "tag": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"
    }
  }
  "cost-map" : {
    "PID1" : { "PID2" : 9 },
    "PID3" : { "PID1" : null, "PID3" : 1 }
  }
}
```

Hence applying the JSON merge patch to the initial cost map is equivalent to the following ALTO cost map:

```
{
  "meta" : {
    "dependent-vtags" : [
      { "resource-id": "my-network-map",
        "tag": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
      }
    ],
    "cost-type" : {
      "cost-mode" : "numerical",
      "cost-metric": "routingcost"
    },
    "vtag": {
      "resource-id": "my-cost-map",
      "tag": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"
    }
  },
  "cost-map" : {
    "PID1": { "PID1": 1, "PID2": 9, "PID3": 10 },
    "PID2": { "PID1": 5, "PID2": 1, "PID3": 15 },
    "PID3": { "PID2": 15, "PID3": 1 }
  }
}
```

### 3.2. Incremental Encoding: JSON Patch

### 3.2.1. JSON Patch Encoding

One issue of JSON merge patch is that it does not handle array changes well. In particular, JSON merge patch considers an array as a single object and hence can only replace an array in its entirety. When the change is to make a small change to an array such as the deletion of an element from a large array, whole-array replacement is inefficient. Consider the example in Section 3.1.2.1. To add a new entry to the `ipv4` array for `PID1`, the server needs to send a whole new array. Another issue is that JSON merge patch cannot change a value to be null, as the JSON merge patch processing algorithm (`MergePatch` in Section 3.1.1) interprets a null as a removal instruction. On the other hand, some ALTO resources can have null values, and it is possible that the update will want to change the new value to be null.

JSON patch [RFC6902] can address the preceding issues. It defines a set of operators to modify a JSON object. See [RFC6902] for the normative definition.

### 3.2.2. JSON Patch ALTO Messages

To provide both examples of JSON patch and a demonstration of the difference between JSON patch and JSON merge patch, the sections below show the application of JSON patch to the same updates shown in Section 3.1.2.

#### 3.2.2.1. JSON Patch Network Map Messages

First consider the same update as in Section 3.1.2.1 for the network map. Below is the encoding using JSON patch:

```
[
  {
    "op": "replace",
    "path": "/meta/vtag/tag",
    "value": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
  },
  {
    "op": "add",
    "path": "/network-map/PID1/ipv4/2",
    "value": "203.0.113.0/25"
  },
  {
    "op": "add",
    "path": "/network-map/PID1/ipv6",
    "value": ["2001:db8:8000::/33"]
  },
  {
    "op": "remove",
    "path": "/network-map/PID2"
  }
]
```

#### 3.2.2.2. JSON Patch Cost Map Messages

Compared with JSON merge patch, JSON patch does not encode cost map updates efficiently. Consider the cost map update shown in Section 3.1.2.2, the encoding using JSON patch is:



```
[
  {
    "op": "replace",
    "path": "/meta/vtag/tag",
    "value": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"
  },
  {
    "op": "replace",
    "path": "/cost-map/PID1/PID2",
    "value": 9
  },
  {
    "op": "remove",
    "path": "/cost-map/PID3/PID1"
  },
  {
    "op": "replace",
    "path": "/cost-map/PID3/PID3",
    "value": 1
  }
]
```

### 3.3. Multiplexing and Server Push: HTTP/2

HTTP/2 ([RFC7540]) provides two related features: multiplexing and server push. In particular, HTTP/2 allows a client and a server to multiplex multiple HTTP requests and responses over a single TCP connection. The requests and responses can be interleaved on a block (frame) by block (frame) basis, by indicating the requests and responses in HTTP/2 messages, avoiding the head-of-line blocking problem encountered with HTTP/1.1. To achieve the same goal, this design introduces substream-id to allow a client to receive updates to multiple resources. HTTP/2 also provides a Server Push facility, to allow a server to send asynchronous updates.

Despite the two features of HTTP/2, this design chooses an HTTP/1.x-compatible design for the simplicity of HTTP/1.x. An HTTP/2 based design may more likely need to be implemented using a more complex HTTP/2 client library. In such a case, one approach for using Server Push for updates is for the update stream server to send each data update message as a separate Server Push item and let the client apply those updates as they arrive. An HTTP/2 client library may not necessarily inform a client application when the server pushes a resource. Instead, the library might cache the pushed resource, and only deliver it to the client when the client explicitly requests that URI. Further, it is more likely that an HTTP/2 based design may encounter issues with a proxy between the client and the server, in that Server Push is optional and can be

disabled by any proxy between the client and the server. This is not a problem for the intended use of Server Push: eventually the client will request those resources, so disabling Server Push just adds a delay. But this means that Server Push is not suitable for resources which the client does not know to request.

Thus this design leaves an HTTP/2 based design as a future work and focuses on ALTO updates on HTTP/1.x and [SSE].

### 3.4. Server Push: Server-Sent Event

Server-Sent Events (SSE) is a technique which can work with HTTP/1.1. The following is a non-normative summary of SSE; see [SSE] for its normative definition.

SSE enable a server to send new data to a client by "server-push". The client establishes an HTTP ([RFC7230], [RFC7231]) connection to the server and keeps the connection open. The server continually sends messages. Each message has one or more lines, where a line is terminated by a carriage-return immediately followed by a new-line, a carriage-return not immediately followed by a new-line, or a new-line not immediately preceded by a carriage-return. A message is terminated by a blank line (two line terminators in a row).

Each line in a message is of the form "field-name: string value". Lines with a blank field-name (that is, lines which start with a colon) are ignored, as are lines which do not have a colon. The protocol defines three field names: event, id, and data. If a message has more than one "data" line, the value of the data field is the concatenation of the values on those lines. There can be only one "event" and "id" line per message. The "data" field is required; the others are optional.

Figure 1 is a sample SSE stream, starting with the client request. The server sends three events and then closes the stream.

```
(Client request)
GET /stream HTTP/1.1
Host: example.com
Accept: text/event-stream

(Server response)
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/event-stream

event: start
id: 1
data: hello there

event: middle
id: 2
data: let's chat some more ...
data: and more and more and ...

event: end
id: 3
data: goodbye
```

Figure 1: A Sample SSE stream.

#### 4. Overview of Approach and High-level Protocol Message Flow

With the preceding background, this section now gives a non-normative overview of the update mechanisms and message flow to be defined in later sections of this document. Figure 2 gives the main components and overall message flow.

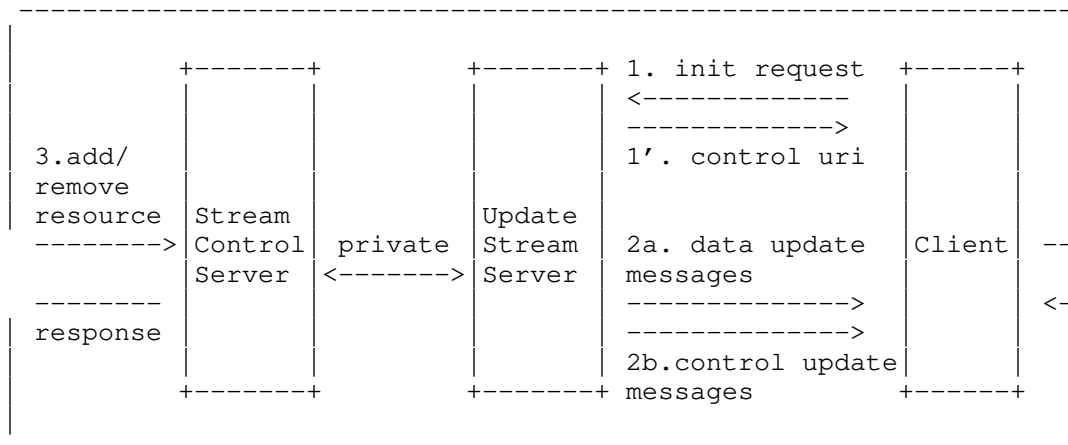


Figure 2: ALTO SSE Architecture and Message Flow.

#### 4.1. Update Stream Service Message Flow

The building block of the update mechanism defined in this document is the update stream service (defined in Section 6), where each update stream service is a POST-mode service that provides update streams.

Note that the lines of the format **"\*\* ... \*\*"** are used to describe message flows in this section and the following sections.

**\*\* Initial request: client -> update server \*\*:**

When an ALTO client requests an update stream service, the ALTO client establishes a persistent connection to the update stream server and submits an initial update-stream request (defined in Section 6.5), creating an update stream. This initial request creating the update stream is labeled "1. init request" in Figure 2.

An update stream can provide updates to both GET-mode resources, such as ALTO network and cost maps, and POST-mode resources, such as ALTO endpoint property service. Also, to avoid creating too many update streams, this design allows an ALTO client to use one update stream to receive updates to multiple requests. In particular, the client may request to receive updates for the same resource but with different parameters for a POST-mode resource, in addition to being able to consolidate updates for multiple resources into a single stream. The updates for each request is called a substream, and hence, the update server needs an identifier to indicate the

substream when sending an update. To achieve this goal, the client assigns a unique substream-id when requesting updates to a resource in an update stream, and the server puts the substream-id in each update.

**\*\* Data updates: update server -> client \*\*:**

The objective of an update stream is to continuously push to an ALTO client data value changes to a set of resources, where the set of resources is specified by the ALTO client's requests. This document refers to messages sending such data-value changes as data update messages (defined in Section 5.2). Although an update stream may update one or more requests, each data update message updates only one request and is sent as a Server-Sent Event (SSE), as defined by [SSE]. A data update message is encoded either as a full replacement or as an incremental change. A full replacement uses the JSON message format defined by the ALTO protocol. There can be multiple encodings for incremental changes. The current design supports incremental changes using JSON merge patch ([RFC7396]) or JSON patch ([RFC6902]) to describe the changes of the resource. Future documents may define additional mechanisms for incremental changes. The update stream server decides when to send data update messages, and whether to send full replacements or incremental changes. These decisions can vary from resource to resource and from update to update. Since the transport is a HTTP/1.x compatible design, data update messages are delivered reliably and in order, and the lossless, sequential delivery of its messages allows the server to know the exact state of the client to compute the correct incremental updates. Figure 2 shows examples of data update messages (labeled "2a. data update messages") in the overall message flow.

**\*\* Control updates: update server -> client \*\*:**

An update stream can run for a long time, and hence there can be status changes at the update stream server side during the lifetime of an update stream; for example, the update stream server may encounter an error or need to shut down for maintenance. To support robust, flexible protocol design, this document allows the update stream server to send control update messages (defined in Section 5.3) in addition to data update messages to the ALTO client. Figure 2 shows that both data updates and control updates can be sent by the server to the client (labeled "2b. control update messages").

#### 4.2. Stream Control Service Message Flow

**\*\* Stream control: client -> stream control server \*\*:**

In addition to control changes triggered from the update stream server side, in a flexible design, an ALTO client may initiate control changes as well, in particular, by adding or removing ALTO resources receiving updates. An ALTO client initiates such changes using the stream control service (defined in Section 7). Although one may use a design that the client uses the same HTTP connection to send the control requests, it requires stronger server support such as HTTP pipeline. For more flexibility, this document introduces stream control service. In particular, the update stream server of an update stream uses the first message to provide the URI of the stream control service (labeled "1': control uri" in Figure 2).

The ALTO client can then use the URI to ask the stream control server specified in the URI to request the update stream server to (1) send data update messages for additional resources, (2) stop sending data update messages for previously requested resources, or (3) gracefully stop and close the update stream altogether.

#### 4.3. Service Announcement and Management Message Flow

**\*\* Service announcements: IRD server -> client \*\*:**

An update server may provide any number of update stream services, where each update stream may provide updates for a given subset of the ALTO server's resources. An ALTO server's Information Resource Directory (IRD) defines the update stream services and declares the set of resources for which each update stream service provides updates. The ALTO server selects the resource set for each update stream service. It is recommended that if a resource depends on one or more other resource(s) (indicated with the "uses" attribute defined in [RFC7285]), these other resource(s) should also be part of that update stream. Thus the update stream for a cost map should also provide updates for the network map on which that cost map depends.

**\*\* Service management (server) \*\*:**

An ALTO client may request any number of update streams simultaneously. Because each update stream consumes resources on the update stream server, an update stream server may require client authorization and/or authentication, limit the number of open update streams, close inactive streams, or redirect an ALTO client to another update stream server.

## 5. Update Messages: Data Update and Control Update Messages

This section defines the format of update messages sent from the server to the client. It first defines the generic structure of update messages (Section 5.1). It then defines the details of the data update messages (Section 5.2) and the control update messages (Section 5.3). These messages will be used in the next two sections to define the Update Stream Service (Section 6) and the Stream Control Service (Section 7).

### 5.1. Generic ALTO Update Message Structure

Both data update and control update messages from the server to the client have the same basic structure: each message includes a data field to provide data information, which is typically a JSON object; and an event field preceding the data field, to specify the media type indicating the encoding of the data field.

A data update message needs additional information to identify the ALTO data (object) to which the update message applies. To be generic, this document use a data-id to identify the ALTO data (object) to be updated; see below.

Hence, the event field of ALTO update message can include two sub-fields (media-type and data-id), where the two sub-fields are separated by a comma (',', U+002C):

```
media-type [ ',' data-id ]
```

According to Section 4.2 of [RFC6838], the comma character is not allowed in a media-type name. So there is no ambiguous when decoding of the two sub-fields.

Note that an update message does not use the SSE "id" field.

### 5.2. ALTO Data Update Message

A data update message is sent when a monitored resource changes. As discussed in the preceding section, the event field of a data update message includes two sub-fields: 'media-type' and 'data-id'.

The 'media-type' sub-field depends on whether the data update is a complete specification of the identified data, or an incremental patch (e.g., a JSON merge patch or JSON patch), if possible, describing the changes from the last version of the data. This document refers to these as full replacement and incremental change, respectively. The encoding of a full replacement is defined by its defining document (e.g., network and cost map messages by [RFC7285]),

and uses the media type defined in that document. The encoding of JSON merge patch is defined by [RFC7396], with the media type "application/merge-patch+json"; the encoding of JSON patch is defined by [RFC6902], with media type "application/json-patch+json".

The 'data-id' sub-field identifies the ALTO data to which the data update message applies.

First consider the case that the resource containing only a single JSON object. For example, since an ALTO client can request data updates for both a cost map resource (object) and its dependent network map resource (object) in the same update stream, to distinguish the updates, the client assigns a substream-id for each resource receiving data updates. Substream-ids MUST be unique within an update stream, but need not be globally unique. A substream-id is encoded as a JSON string with the same format as that of the type ResourceID (Section 10.2 of [RFC7285]). The type SubstreamID is used in this document to indicate a string of this format. The substream-id of a single JSON object is the 'data-id'.

As an example, assume that the ALTO client assigns substream-id "1" in its request to receive updates to the network map; and substream-id "2" to the cost map. Then the substream-ids are the data-ids indicating which objects will be updated. Figure 3 shows some examples of ALTO data update messages:

```
event: application/alto-networkmap+json,1
data: { ... full network map message ... }

event: application/alto-costmap+json,2
data: { ... full cost map message ... }

event: application/merge-patch+json,2
data: { ... JSON merge patch update for the cost map ... }
```

Figure 3: Examples of ALTO data update messages.

Next consider the case that a resource may include multiple JSON objects. This document considers the case that a resource may contain multiple components (parts) and they are encoded using the media type "multipart/related" [RFC2387]. Each part of this multipart response MUST be an HTTP message including a Content-ID header and a JSON object body. Each component requiring the update stream service (defined in Section 6) MUST be identified by a unique Content-ID to be defined in its defining document.



For a resource using the media type "multipart/related", the 'data-id' sub-field MUST be the concatenation of the substream-id, the '.' separator (U+002E) and the unique Content-ID in order.

### 5.3. ALTO Control Update Message

Control update messages have the media type "application/alto-updatestreamcontrol+json", and the data is of type UpdateStreamControlEvent:

```
object {  
  [String          control-uri;]  
  [SubstreamID     started<1..*>;]  
  [SubstreamID     stopped<1..*>;]  
  [String          description;]  
} UpdateStreamControlEvent;
```

control-uri: the URI providing stream control for this update stream (see Section 7). The server sends a control update message notifying the client of the control-uri. This control update message notifying the control-uri will be sent once and MUST be the first event in an update stream. If the URI value is NULL, the update stream server does not support stream control for this update stream; otherwise, the update stream server provides stream control through the given URI.

started: a list of substream-ids of resources. It notifies the ALTO client that the update stream server will start sending data update messages for each resource listed.

stopped: a list of substream-ids of resources. It notifies the ALTO client that the update stream server will no longer send data update messages for the listed resources. There can be multiple reasons for an update stream server to stop sending data update messages for a resource, including a request from the ALTO client using stream control (Section 6.7.1) or an internal server event.

description: a non-normative, human-readable text providing an explanation for the control event. When an update stream server stops sending data update messages for a resource, it is RECOMMENDED that the update stream server use the description field to provide details. There can be multiple reasons which trigger a "stopped" event; see above. The intention of this field is to provide a human-readable text for the developer and/or the administrator to diagnose potential problems.

## 6. Update Stream Service

An update stream service returns a stream of update messages, as defined in Section 5. An ALTO server's IRD (Information Resource Directory) MAY define one or more update stream services, which ALTO clients use to request new update stream instances. An IRD entry defining an update stream service MUST define the media type, HTTP method, and capabilities & uses as follows.

### 6.1. Media Type

The media type of an ALTO update stream service is "text/event-stream", as defined by [SSE].

### 6.2. HTTP Method

An ALTO update stream service is requested using the HTTP POST method.

### 6.3. Capabilities

The capabilities are defined as an object of type `UpdateStreamCapabilities`:

```
object {
  IncrementalUpdateMediaTypes incremental-change-media-types;
  Boolean                      support-stream-control;
} UpdateStreamCapabilities;

object-map {
  ResourceID -> String;
} IncrementalUpdateMediaTypes;
```

If this update stream can provide data update messages with incremental changes for a resource, the "incremental-change-media-types" field has an entry for that resource-id, and the value is the supported media types of the incremental change separated by commas. Normally this will be "application/merge-patch+json", "application/json-patch+json", or "application/merge-patch+json,application/json-patch+json", because, as described in Section 5, they are the only incremental change types defined by this document. However future extensions may define other types of incremental changes.

When choosing the media-types to encode incremental changes for a resource, the update stream server MUST consider the limitations of the encoding. For example, when a JSON merge patch specifies that the value of a field is null, its semantics is that the field is removed from the target, and hence the field is no longer defined

(i.e., undefined); see the MergePatch algorithm in Section 3.1.1 on how null value is processed. This, however, may not be the intended result for the resource, when null and undefined have different semantics for the resource. In such a case, the update stream server MUST choose JSON patch over JSON merge patch, if JSON patch is indicated as a capability of the update stream server; If the the server does not support JSON patch to handle such a case, the server then need to send a full replacement.

The "support-stream-control" field specifies whether the given update stream supports stream control. If "support-stream-control" field is "true", the update stream server will use the stream control specified in this document; else, the update stream server may use other mechanisms to provide the same functionality as stream control.

#### 6.4. Uses

The "uses" attribute MUST be an array with the resource-ids of every resource for which this update stream can provide updates. Each resource specified in the "uses" MUST support full replacement: the update stream server can always send full replacement, and the ALTO client MUST accept full replacement.

This set may be any subset of the ALTO server's resources, and may include resources defined in linked IRDs. However, it is RECOMMENDED that the ALTO server selects a set that is closed under the resource dependency relationship. That is, if an update stream's "uses" set includes resource R1, and resource R1 depends on ("uses") resource R0, then the update stream's "uses" set SHOULD include R0 as well as R1. For example, an update stream for a cost map SHOULD also provide updates for the network map upon which that cost map depends.

#### 6.5. Request: Accept Input Parameters

An ALTO client specifies the parameters for the new update stream by sending an HTTP POST body with the media type "application/alto-updatestreamparams+json". That body contains a JSON Object of type UpdateStreamReq, where:

```
object {  
  [AddUpdatesReq    add;]  
  [SubstreamID      remove<0...*>;]  
} UpdateStreamReq;  
  
object-map {  
  SubstreamID -> AddUpdateReq;  
} AddUpdatesReq;  
  
object {  
  ResourceID    resource-id;  
  [JSONString   tag;]  
  [Boolean      incremental-changes;]  
  [Object       input;]  
} AddUpdateReq;
```

**add:** specifies the resources (and the parameters for the resources) for which the ALTO client wants updates. In the scope of the same update stream, the ALTO client **MUST** assign a substream-id that is unique in the scope of the update stream (Section 5.2) for each entry, and use those substream-ids as the keys in the "add" field.

**resource-id:** the resource-id of an ALTO resource, and **MUST** be in the update stream's "uses" list (Section 6.4). If the resource-id is a GET-mode resource with a version tag (or "vtag"), as defined in Section 6.3 and Section 10.3 of [RFC7285], and the ALTO client has previously retrieved a version of that resource from the update stream server, the ALTO client **MAY** set the "tag" field to the tag part of the client's version of that resource. If that version is not current, the update stream server **MUST** send a full replacement before sending any incremental changes, as described in Section 6.7.1. If that version is still current, the update stream server **MAY** omit the initial full replacement.

**incremental-changes:** the ALTO client specifies whether it is willing to receive incremental changes from the update stream server for this substream. If the "incremental-changes" field is "true", the update stream server **MAY** send incremental changes for this substream. In this case, the client **MUST** support all incremental methods from the set announced in the server's capabilities for this resource; see Section 6.3 for server's announcement of potential incremental methods. If a client does not support all incremental methods from the set announced in the server's capabilities, the client can set "incremental-changes" to "false", and the update stream server then **MUST NOT** send incremental changes for that substream. The default value

for "incremental-changes" is "true", so to suppress incremental changes, the ALTO client MUST explicitly set "incremental-changes" to "false". An alternative design of incremental-changes control is a more fine-grained control, by allowing a client to select a subset of incremental methods from the set announced in the server's capabilities. But this alternative design is not adopted in this document, because it adds complexity to the server, which is more likely to be the bottleneck. Note that the ALTO client cannot suppress full replacement. When the ALTO client sets "incremental-changes" to "false", the update stream server MUST send a full replacement instead of an incremental change to the ALTO client. The update stream server MAY wait until more changes are available, and send a single full replacement with those changes. Thus an ALTO client which declines to accept incremental changes may not get updates as quickly as an ALTO client which does.

input: If the resource is a POST-mode service which requires input, the ALTO client MUST set the "input" field to a JSON Object with the parameters that the resource expects.

remove: it is used in update stream control requests (Section 7), and is not allowed in the update stream request. The update stream server SHOULD ignore this field if it is included in the request.

If a request has any errors, the update stream server MUST NOT create an update stream. Also, the update stream server will send an error response to the ALTO client as specified in Section 6.6.

## 6.6. Response

If the update stream request has any errors, the update stream server MUST return an HTTP "400 Bad Request" to the ALTO client. The body part of the HTTP response is the JSON object defined in Section 8.5.2 in [RFC7285]. Hence, an ALTO error response has the format:

```
HTTP/1.1 400 Bad Request
Content-Length: 131
Content-Type: application/alto-error+json
Connection: Closed
```

```
{
  "meta":{
    "code": "E_INVALID_FIELD_VALUE",
    "field": "add/my-network-map/resource-id",
    "value": "my-networkmap/#"
  }
}
```

Note that "field" and "value" are optional fields. If the "value" field exists, the "field" field MUST exist.

- o If an update stream request does not have an "add" field specifying one or more resources, the error code of the error message MUST be E\_MISSING\_FIELD and the "field" field SHOULD be "add". The update stream server MUST close the stream without sending any events.
- o If the "resource-id" field is invalid, or is not associated with the update stream, the error code of the error message MUST be E\_INVALID\_FIELD\_VALUE; the "field" field SHOULD be the full path of the "resource-id" field and the "value" field SHOULD be the invalid resource-id. If there are more than one invalid resource-ids, the update stream server SHOULD pick one and return it. The update stream server MUST close the stream (i.e., TCP connection) without sending any events.
- o If the resource is a POST-mode service which requires input, the client MUST set the "input" field to a JSON Object with the parameters that that resource expects. If the "input" field is missing or invalid, the update stream server MUST return the same error response that that resource would return for missing or invalid input (see [RFC7285]). In this case, the update stream server MUST close the update stream without sending any events. If the input for several POST-mode resources are missing or invalid, the update stream server MUST pick one and return it.

The response to a valid request is a stream of update messages. Section 5 defines the update messages, and [SSE] defines how they are encoded into a stream.

An update stream server SHOULD send updates only when the underlying values change. However, it may be difficult for an update stream

server to guarantee that in all circumstances. Therefore a client MUST NOT assume that an update message represents an actual change.

## 6.7. Additional Requirements on Update Stream Service

### 6.7.1. Event Sequence Requirements

- o The first event MUST be a control update message with the URI of the update stream control service (see Section 7) for this update stream. Note that the value of the control-uri can be "null", indicating that there is no control stream service.
- o As soon as possible after the ALTO client initiates the connection, the update stream server checks the "tag" field for each added update request. If the "tag" field is not specified in an added update request, the update stream server MUST first send a full replacement for the request. If the "tag" field is specified, the client can accept incremental changes, and the server can compute an incremental update based on the "tag" (the server needs to ensure that for a POST resource with input, the "tag" should indicate the correct result for different inputs), the update stream server MAY omit the initial full replacement.
- o If this update stream provides updates for resource-ids R0 and R1, and if R1 depends on R0, then the update stream server MUST send the update for R0 before sending the related updates for R1. For example, suppose an update stream provides updates to a network map and its dependent cost maps. When the network map changes, the update stream server MUST send the network map update before sending the cost map updates.
- o When the ALTO client uses the stream control service to stop updates for one or more resources (Section 7), the ALTO client MUST send a stream control request. The update stream server MUST send a control update message whose "stopped" field has the substream-ids of all stopped resources.

### 6.7.2. Cross-Stream Consistency Requirements

If multiple ALTO clients create multiple update streams from the same update stream resource, and with the same update request parameters (i.e., same resource, same input), the update stream server MUST send the same updates to all of them. However, the update stream server MAY pack data items into different patch events, as long as the net result of applying those updates is the same.

For example, suppose two different ALTO clients create two different update streams for the same cost map, and suppose the update stream

server processes three separate cost point updates with a brief pause between each update. The server **MUST** send all three new cost points to both clients. But the update stream server **MAY** send a single patch event (with all three cost points) to one ALTO client, while sending three separate patch events (with one cost point per event) to the other ALTO client.

A update stream server **MAY** offer several different update stream resources that provide updates to the same underlying resource (that is, a resource-id may appear in the "uses" field of more than one update stream resource). In this case, those update stream resources **MUST** return the same update.

### 6.7.3. Multipart Update Requirements

This design allows any valid media type for full replacement. Hence, it supports ALTO resources using multipart to contain multiple JSON objects. This realizes the push benefit, but not the incremental encoding benefit of SSE.

JSON patch and merge patch provide the incremental encoding benefit but can be applied to only a single JSON object. If an update stream service supports a resource providing a multipart media type, which we refer to as a multipart resource, then the update stream service needs to handle the issue that the message of a full multipart resource can include multiple JSON objects. To address the issue, when an update stream service specifies that it supports JSON patch or merge patch incremental updates for a multipart resource, the service **MUST** ensure that (1) each part of a multipart message is a single JSON object, (2) each part is specified by a static content-id in the initial full message, (3) each data update event applies to only one part; and (4) each data update specifies substream-id.content-id as the 'event' field of the event, to identify the part to be updated.

### 6.8. Keep-Alive Messages

In an SSE stream, any line which starts with a colon (U+003A) character is a comment, and an ALTO client **MUST** ignore that line ([SSE]). As recommended in [SSE], an update stream server **SHOULD** send a comment line (or an event) every 15 seconds to prevent ALTO clients and proxy servers from dropping the HTTP connection. Note that although TCP also provides a Keep-alive function, the interval between TCP Keep-alive messages can depend on the OS configuration and varies. The preceding recommended SSE keep-alive allows the SSE client to detect the status of the update stream server with more certainty.



## 7. Stream Control Service

A stream control service allows an ALTO client to remove resources from the set of resources that are monitored by an update stream, or add additional resources to that set. The service also allows an ALTO client to gracefully shut down an update stream.

When an update stream server creates a new update stream, and if the update stream server supports stream control for the update stream, the update stream server creates a stream control service for that update stream. An ALTO client uses the stream control service to remove resources from the update stream instance, or to request updates for additional resources. An ALTO client cannot obtain the stream control service through the IRD. Instead, the first event that the update stream server sends to the ALTO client has the URI for the associated stream control service (see Section 5.3).

Each stream control request is an individual HTTP request. The ALTO client MAY send multiple stream control requests to the stream control server using the same HTTP connection.

### 7.1. URI

The URI for a stream control service, by itself, MUST uniquely specify the update stream instance which it controls. The stream control server MUST NOT use other properties of an HTTP request, such as cookies or the client's IP address, to determine the update stream. Furthermore, an update stream server MUST NOT reuse a control service URI once the associated update stream has been closed.

The ALTO client MUST evaluate a relative control URI reference [RFC3986] (for example, a URI reference without a host, or with a relative path) in the context of the URI used to create the update stream. The stream control service's host MAY be different from the update stream's host.

It is expected that there is an internal mechanism to map a stream control URI to the unique update stream instance to be controlled. For example, the update stream service may assign a unique, internal stream id to each update stream instance. However, the exact mechanism is left to the update stream service provider.

To prevent an attacker from forging a stream control URI and sending bogus requests to disrupt other update streams, the service should consider two security issues. First, if http, not https, is used, the stream control URI can be exposed to an on-path attacker. To address this issue, in a setting where the path from the server to

the client can traverse such an attacker, the server SHOULD use https. Second, even without direct exposure, an off-path attacker may guess valid stream control URIs. To address this issue, the server SHOULD choose stream control URIs with enough randomness, to make guessing difficult; the server SHOULD introduce mechanisms that detect repeated guesses indicating an attack (e.g., keeping track of the number of failed stream control attempts); please see <https://www.w3.org/TR/capability-urls/> .

## 7.2. Media Type

An ALTO stream control response does not have a specific media type.

## 7.3. HTTP Method

An ALTO update stream control resource is requested using the HTTP POST method.

## 7.4. IRD Capabilities & Uses

None (Stream control services do not appear in the IRD).

## 7.5. Request: Accept Input Parameters

A stream control service accepts the same input media type and input parameters as the update stream service (Section 6.5). The only difference is that a stream control service also accepts the "remove" field.

If specified, the "remove" field is an array of substream-ids the ALTO client previously added to this update stream. An empty "remove" array is equivalent to a list of all currently active resources; the update stream server responds by removing all resources and closing the stream.

An ALTO client MAY use the "add" field to add additional resources. The ALTO client MUST assign a unique substream-id to each additional resource. Substream-ids MUST be unique over the lifetime of this update stream: an ALTO client MUST NOT reuse a previously removed substream-id. The processing of an "add" resource is the same as discussed in Section 6.5 and Section 6.7.

If a request has any errors, the update stream server MUST NOT add or remove any resources from the associated update stream. Also, the stream control server will return an error response to the client as specified in Section 7.6.

## 7.6. Response

The stream control server MUST process the "add" field before the "remove" field. If the request removes all active resources without adding any additional resources, the update stream server MUST close the update stream. Thus an update stream cannot have zero resources.

If the request has any errors, the stream control server MUST return an HTTP "400 Bad Request" to the ALTO client. The body part of the HTTP response is the JSON object defined in Section 8.5.2 in [RFC7285]. An error response has the same format as specified in Section 6.6. Detailed error code and error information are specified as below.

- o If the "add" request does not satisfy the requirements in Section 6.5, the stream control server MUST return the ALTO error message defined in Section 6.6.
- o If any substream-id in the "remove" field was not added in a prior request, the error code of the error message MUST be `E_INVALID_FIELD_VALUE`; the "field" field SHOULD be "remove" and the "value" field SHOULD be an array of the invalid substream-ids. Thus it is illegal to "add" and "remove" the same substream-id in the same request. However, it is legal to remove a substream-id twice. To support the preceding checking, the update stream server MUST keep track of previously-used-but-now-closed substream-ids.
- o If any substream-id in the "add" field has been used before in this stream, the error code of the error message MUST be `E_INVALID_FIELD_VALUE`, the "field" field SHOULD be "add" and the "value" field SHOULD be an array of invalid substream-ids.
- o If the request has a non-empty "add" field and a "remove" field with an empty list of substream-ids (to replace all active resources with a new set, the client MUST explicitly enumerate the substream-ids to be removed), the error code of the error message MUST be `E_INVALID_FIELD_VALUE`; the "field" field SHOULD be "remove" and the "value" field SHOULD be an empty array.

If the request is valid but the associated update stream has been closed then the stream control server MUST return an HTTP "404 Not Found".

If the request is valid and the stream control server successfully processes the request without error, the stream control server should return either an HTTP "202 Accepted" response or an HTTP "204 No Content" response. The difference is that for the latter case, the

stream control server is sure that the update stream server has also processed the request. Regardless of 202 or 204 HTTP response, the final updates of related resources will be notified by the update stream server using its control update message(s), due to the modular design.

## 8. Examples

### 8.1. Example: IRD Announcing Update Stream Services

Below is an example IRD announcing three update stream services. The first, which is named "update-my-costs", provides updates for the network map, the "routingcost" and "hopcount" cost maps, and a filtered cost map resource. The second, which is named "update-my-prop", provides updates to the endpoint properties service. The third, which is named "update-my-pv", provides updates to a non-standard ALTO service returning a multipart response.

Note that in the "update-my-costs" update stream shown in the example IRD, the update stream server uses JSON patch for network map, and it uses JSON merge patch to update the other resources. Also, the update stream will only provide full replacements for "my-simple-filtered-cost-map".

Also, note that this IRD defines two filtered cost map resources. They use the same cost types, but "my-filtered-cost-map" accepts cost constraint tests, while "my-simple-filtered-cost-map" does not. To avoid the issues discussed in Section 9.3, the update stream provides updates for the second, but not the first.

This IRD also announces a non-standard ALTO service, which is named "my-pv". This service accepts an extended endpoint cost request as an input and returns a multipart response including an endpoint cost resource and a property map resource. This document does not rely on any other design details of this new service. In this document, the "my-pv" service is only used to illustrate how the update stream service provides updates to an ALTO resource returning a multipart response.

```
"my-network-map": {
  "uri": "https://alto.example.com/networkmap",
  "media-type": "application/alto-networkmap+json",
},
"my-routingcost-map": {
  "uri": "https://alto.example.com/costmap/routingcost",
  "media-type": "application/alto-costmap+json",
  "uses": ["my-networkmap"],
  "capabilities": {
```

```
    "cost-type-names": ["num-routingcost"]
  },
  "my-hopcount-map": {
    "uri": "https://alto.example.com/costmap/hopcount",
    "media-type": "application/alto-costmap+json",
    "uses": ["my-networkmap"],
    "capabilities": {
      "cost-type-names": ["num-hopcount"]
    }
  },
  "my-filtered-cost-map": {
    "uri": "https://alto.example.com/costmap/filtered/constraints",
    "media-type": "application/alto-costmap+json",
    "accepts": "application/alto-costmapfilter+json",
    "uses": ["my-networkmap"],
    "capabilities": {
      "cost-type-names": ["num-routingcost", "num-hopcount"],
      "cost-constraints": true
    }
  },
  "my-simple-filtered-cost-map": {
    "uri": "https://alto.example.com/costmap/filtered/simple",
    "media-type": "application/alto-costmap+json",
    "accepts": "application/alto-costmapfilter+json",
    "uses": ["my-networkmap"],
    "capabilities": {
      "cost-type-names": ["num-routingcost", "num-hopcount"],
      "cost-constraints": false
    }
  },
  "my-props": {
    "uri": "https://alto.example.com/properties",
    "media-type": "application/alto-endpointprops+json",
    "accepts": "application/alto-endpointpropparams+json",
    "capabilities": {
      "prop-types": ["priv:ietf-bandwidth"]
    }
  },
  "my-pv": {
    "uri": "https://alto.example.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" ]
    }
  }
```

```
    },
    "update-my-costs": {
      "uri": "https://alto.example.com/updates/costs",
      "media-type": "text/event-stream",
      "accepts": "application/alto-updatestreamparams+json",
      "uses": [
        "my-network-map",
        "my-routingcost-map",
        "my-hopcount-map",
        "my-simple-filtered-cost-map"
      ],
      "capabilities": {
        "incremental-change-media-types": {
          "my-network-map": "application/json-patch+json",
          "my-routingcost-map": "application/merge-patch+json",
          "my-hopcount-map": "application/merge-patch+json"
        },
        "support-stream-control": true
      }
    },
    "update-my-props": {
      "uri": "https://alto.example.com/updates/properties",
      "media-type": "text/event-stream",
      "uses": [ "my-props" ],
      "accepts": "application/alto-updatestreamparams+json",
      "capabilities": {
        "incremental-change-media-types": {
          "my-props": "application/merge-patch+json"
        },
        "support-stream-control": true
      }
    },
    "update-my-pv": {
      "uri": "https://alto.example.com/updates/pv",
      "media-type": "text/event-stream",
      "uses": [ "my-pv" ],
      "accepts": "application/alto-updatestreamparams+json",
      "capabilities": {
        "incremental-change-media-types": {
          "my-pv": "application/merge-patch+json"
        },
        "support-stream-control": true
      }
    }
  }
```

## 8.2. Example: Simple Network and Cost Map Updates

Given the update streams announced in the preceding example IRD, the section below shows an example of an ALTO client's request and the update stream server's immediate response, using the update stream resource "update-my-costs". In the example, the ALTO client requests updates for the network map and "routingcost" cost map, but not for the "hopcount" cost map. The ALTO client uses the ALTO server's resource-ids as the substream-ids. Because the client does not provide a "tag" for the network map, the update stream server must send a full replacement for the network map as well as for the cost map. The ALTO client does not set "incremental-changes" to "false", so it defaults to "true". Thus, the update stream server will send patch updates for the cost map and the network map.

```
POST /updates/costs HTTP/1.1
Host: alto.example.com
Accept: text/event-stream,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 155

{ "add": {
  "my-network-map": {
    "resource-id": "my-network-map"
  },
  "my-routingcost-map": {
    "resource-id": "my-routingcost-map"
  }
}
}
```

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

event: application/alto-updatestreamcontrol+json
data: {"control-uri":
data: "https://alto.example.com/updates/streams/3141592653589"}

event: application/alto-networkmap+json,my-network-map
data: {
data:   "meta" : {
data:     "vtag": {
data:       "resource-id" : "my-network-map",
data:       "tag" : "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
data:     }
data:   },
data:   "network-map" : {
```

```

data:      "PID1" : {
data:      "ipv4" : [ "192.0.2.0/24", "198.51.100.0/25" ]
data:      },
data:      "PID2" : {
data:      "ipv4" : [ "198.51.100.128/25" ]
data:      },
data:      "PID3" : {
data:      "ipv4" : [ "0.0.0.0/0" ],
data:      "ipv6" : [ "::/0" ]
data:      }
data:    }
data:  }
data: }

event: application/alto-costmap+json,my-routingcost-map
data: {
data:   "meta" : {
data:     "dependent-vtags" : [{
data:       "resource-id": "my-network-map",
data:       "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
data:     }],
data:     "cost-type" : {
data:       "cost-mode" : "numerical",
data:       "cost-metric": "routingcost"
data:     },
data:     "vtag": {
data:       "resource-id" : "my-routingcost-map",
data:       "tag" : "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
data:     }
data:   },
data:   "cost-map" : {
data:     "PID1": { "PID1": 1, "PID2": 5, "PID3": 10 },
data:     "PID2": { "PID1": 5, "PID2": 1, "PID3": 15 },
data:     "PID3": { "PID1": 20, "PID2": 15 }
data:   }
data: }

```

After sending those events immediately, the update stream server will send additional events as the maps change. For example, the following represents a small change to the cost map. PID1->PID2 is changed to 9 from 5, PID3->PID1 is no longer available and PID3->PID3 is now defined as 1:



```

event: application/merge-patch+json,my-routingcost-map
data: {
data:   "meta" : {
data:     "vtag": {
data:       "tag": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"
data:     }
data:   },
data:   "cost-map": {
data:     "PID1" : { "PID2" : 9 },
data:     "PID3" : { "PID1" : null, "PID3" : 1 }
data:   }
data: }

```

As another example, the following represents a change to the network map: an ipv4 prefix "203.0.113.0/25" is added to PID1. It triggers changes to the cost map. The update stream server chooses to send an incremental change for the network map and send a full replacement instead of an incremental change for the cost map:

```

event: application/json-patch+json,my-network-map
data: {
data:   {
data:     "op": "replace",
data:     "path": "/meta/vtag/tag",
data:     "value" : "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
data:   },
data:   {
data:     "op": "add",
data:     "path": "/network-map/PID1/ipv4/2",
data:     "value": "203.0.113.0/25"
data:   }
data: }

event: application/alto-costmap+json,my-routingcost-map
data: {
data:   "meta" : {
data:     "vtag": {
data:       "tag": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"
data:     }
data:   },
data:   "cost-map" : {
data:     "PID1": { "PID1": 1, "PID2": 3, "PID3": 7 },
data:     "PID2": { "PID1": 12, "PID2": 1, "PID3": 9 },
data:     "PID3": { "PID1": 14, "PID2": 8 }
data:   }
data: }

```

### 8.3. Example: Advanced Network and Cost Map Updates

This example is similar to the previous one, except that the ALTO client requests updates for the "hopcount" cost map as well as the "routingcost" cost map and provides the current version tag of the network map, so the update stream server is not required to send the full network map data update message at the beginning of the stream. In this example, the client uses the substream-ids "net", "routing" and "hops" for those resources. The update stream server sends the stream control URI and the full cost maps, followed by updates for the network map and cost maps as they become available:

```
POST /updates/costs HTTP/1.1
Host: alto.example.com
Accept: text/event-stream,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 244

{ "add": {
  "net": {
    "resource-id": "my-network-map",
    "tag": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
  },
  "routing": {
    "resource-id": "my-routingcost-map"
  },
  "hops": {
    "resource-id": "my-hopcount-map"
  }
}
```

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

event: application/alto-updatestreamcontrol+json
data: {"control-uri":
data: "https://alto.example.com/updates/streams/2718281828459"}

event: application/alto-costmap+json, routing
data: { ... full routingcost cost map message ... }

event: application/alto-costmap+json, hops
data: { ... full hopcount cost map message ... }

(pause)

event: application/merge-patch+json, routing
data: {"cost-map": {"PID2" : {"PID3" : 31}}}

event: application/merge-patch+json, hops
data: {"cost-map": {"PID2" : {"PID3" : 4}}}
```

If the ALTO client wishes to stop receiving updates for the "hopcount" cost map, the ALTO client can send a "remove" request on the stream control URI:

```
POST /updates/streams/2718281828459 HTTP/1.1
Host: alto.example.com
Accept: text/plain,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 24

{
  "remove": [ "hops" ]
}
```

```
HTTP/1.1 204 No Content
Content-Length: 0
```

(stream closed without sending data content)

The update stream server sends a "stopped" control update message on the original request stream to inform the ALTO client that updates are stopped for that resource:

```
event: application/alto-updatestreamcontrol+json
data: {
data:   "stopped": ["hops"]
data: }
```

Below is an example of an invalid stream control request. The "remove" field of the request includes an undefined substream-id and the stream control server will return an error response to the ALTO client.

```
POST /updates/streams/2718281828459 HTTP/1.1
Host: alto.example.com
Accept: text/plain,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 31
{
  "remove": [ "properties" ]
}

HTTP/1.1 400 Bad Request
Content-Length: 89
Content-Type: application/alto-error+json

{
  "meta":{
    "code": "E_INVALID_FIELD_VALUE",
    "field": "remove",
    "value": "properties"
  }
}
```

If the ALTO client no longer needs any updates, and wishes to shut the update stream down gracefully, the client can send a "remove" request with an empty array:

```
POST /updates/streams/2718281828459 HTTP/1.1
Host: alto.example.com
Accept: text/plain,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 17
```

```
{
  "remove": [ ]
}
```

```
HTTP/1.1 204 No Content
Content-Length: 0
```

(stream closed without sending data content)

The update stream server sends a final control update message on the original request stream to inform the ALTO client that all updates are stopped and then closes the stream:

```
event: application/alto-updatestreamcontrol+json
data: {
  data: "stopped": ["net", "routing"]
  data: }
```

(server closes stream)

#### 8.4. Example: Endpoint Property Updates

As another example, here is how an ALTO client can request updates for the property "priv:ietf-bandwidth" for one set of endpoints and "priv:ietf-load" for another. The update stream server immediately sends full replacements with the property values for all endpoints. After that, the update stream server sends data update messages for the individual endpoints as their property values change.

```
POST /updates/properties HTTP/1.1
Host: alto.example.com
Accept: text/event-stream
Content-Type: application/alto-updatestreamparams+json
Content-Length: 511
```

```
{ "add": {
  "props-1": {
    "resource-id": "my-props",
    "input": {
      "properties" : [ "priv:ietf-bandwidth" ],
      "endpoints" : [
        "ipv4:198.51.100.1",
        "ipv4:198.51.100.2",
        "ipv4:198.51.100.3"
      ]
    }
  },
  "props-2": {
    "resource-id": "my-props",
    "input": {
      "properties" : [ "priv:ietf-load" ],
      "endpoints" : [
        "ipv6:2001:db8:100::1",
        "ipv6:2001:db8:100::2",
        "ipv6:2001:db8:100::3"
      ]
    }
  }
}
```

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

event: application/alto-updatestreamcontrol+json
data: {"control-uri":
data: "https://alto.example.com/updates/streams/1414213562373"}

event: application/alto-endpointprops+json,props-1
data: { "endpoint-properties": {
data:     "ipv4:198.51.100.1" : { "priv:ietf-bandwidth": "13" },
data:     "ipv4:198.51.100.2" : { "priv:ietf-bandwidth": "42" },
data:     "ipv4:198.51.100.3" : { "priv:ietf-bandwidth": "27" }
data:   } }

event: application/alto-endpointprops+json,props-2
data: { "endpoint-properties": {
data:     "ipv6:2001:db8:100::1" : { "priv:ietf-load": "8" },
data:     "ipv6:2001:db8:100::2" : { "priv:ietf-load": "2" },
data:     "ipv6:2001:db8:100::3" : { "priv:ietf-load": "9" }
data:   } }

  (pause)

event: application/merge-patch+json,props-1
data: { "endpoints-properties":
data:   {"ipv4:198.51.100.1" : {"priv:ietf-bandwidth": "3"}}
data: }

  (pause)

event: application/merge-patch+json,props-2
data: { "endpoint-properties":
data:   {"ipv6:2001:db8:100::3" : {"priv:ietf-load": "7"}}
data: }
```

If the ALTO client needs the "priv:ietf-bandwidth" property and the "priv:ietf-load" property for additional endpoints, the ALTO client can send an "add" request on the stream control URI:

```
POST /updates/streams/1414213562373" HTTP/1.1
Host: alto.example.com
Accept: text/plain,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 448
```

```
{ "add": {
  "props-3": {
    "resource-id": "my-props",
    "input": {
      "properties" : [ "priv:ietf-bandwidth" ],
      "endpoints" : [
        "ipv4:198.51.100.4",
        "ipv4:198.51.100.5"
      ]
    }
  },
  "props-4": {
    "resource-id": "my-props",
    "input": {
      "properties" : [ "priv:ietf-load" ],
      "endpoints" : [
        "ipv6:2001:db8:100::4",
        "ipv6:2001:db8:100::5"
      ]
    }
  }
}
```

```
HTTP/1.1 204 No Content
Content-Length: 0
```

(stream closed without sending data content)

The update stream server sends full replacements for the two new resources, followed by incremental changes for all four requests as they arrive:



```
event: application/alto-endpointprops+json,props-3
data: { "endpoint-properties": {
data:   "ipv4:198.51.100.4" : { "priv:ietf-bandwidth": "25" },
data:   "ipv4:198.51.100.5" : { "priv:ietf-bandwidth": "31" },
data: } }
```

```
event: application/alto-endpointprops+json,props-4
data: { "endpoint-properties": {
data:   "ipv6:2001:db8:100::4" : { "priv:ietf-load": "6" },
data:   "ipv6:2001:db8:100::5" : { "priv:ietf-load": "4" },
data: } }
```

(pause)

```
event: application/merge-patch+json,props-3
data: { "endpoint-properties":
data:   {"ipv4:198.51.100.5" : {"priv:ietf-bandwidth": "15"}}
data: }
```

(pause)

```
event: application/merge-patch+json,props-2
data: { "endpoint-properties":
data:   {"ipv6:2001:db8:100::2" : {"priv:ietf-load": "9"}}
data: }
```

(pause)

```
event: application/merge-patch+json,props-4
data: { "endpoint-properties":
data:   {"ipv6:2001:db8:100::4" : {"priv:ietf-load": "3"}}
data: }
```

## 8.5. Example: Multipart Message Updates

This example shows how an ALTO client can request a non-standard ALTO service returning a multipart response. The update stream server immediately sends full replacements of the multipart response. After that, the update stream server sends data update messages for the individual parts of the response as the ALTO data (object) in each part changes.

```
POST /updates/pv HTTP/1.1
Host: alto.example.com
Accept: text/event-stream
Content-Type: application/alto-updatestreamparams+json
Content-Length: 382
```

```
{
  "add": {
    "ecspvsub1": {
      "resource-id": "my-pv",
      "input": {
        "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" ]
        },
        "ane-properties": [ "maxresbw", "persistent-entities" ]
      }
    }
  }
}
```

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

event: application/alto-updatestreamcontrol+json
data: {"control-uri":
data:      "https://alto.example.com/updates/streams/1414"}

event: multipart/related;boundary=example-pv;
      type=application/alto-endpointcost+json,ecspvsubl
data: --example-pv
data: Content-ID: ecsmap
data: Content-Type: application/alto-endpointcost+json
data:
data: { ... data (object) of an endpoint cost map ... }
data: --example-pv
data: Content-ID: propmap
data: Content-Type: application/alto-propmap+json
data:
data: { ... data (object) of a property map ... }
data: --example-pv--

      (pause)

event: application/merge-patch+json,ecspvsubl.ecsmap
data: { ... merge patch for updates of ecspvsubl.ecsmap ... }

event: application/merge-patch+json,ecspvsubl.propmap
data: { ... merge patch for updates of ecspvsubl.propmap ... }
```

## 9. Operation and Processing Considerations

### 9.1. Considerations for Choosing Data Update Messages

The update stream server should be cognizant of the effects of its update schedule, which includes both the choice of timing (i.e., when/what to trigger an update) and the choice of message format (i.e., given an update, send a full replacement or an incremental change). In particular, the update schedule can have effects on both the overhead and the freshness of information. To minimize overhead, the server may choose to batch a sequence of updates for resources that frequently change, by sending cumulative updates or a full replacement after a while. The update stream server should be cognizant that batching reduces the freshness of information. The server should also consider the effect of such delays on client behaviors (see below on client timeout on waiting for updates of dependent resources).

For incremental updates, this design allows both JSON patch and JSON merge patch for incremental changes. JSON merge patch is clearly superior to JSON patch for describing incremental changes to Cost Maps, Endpoint Costs, and Endpoint Properties. For these data structures, JSON merge patch is more space-efficient, as well as simpler to apply; There is no advantage allowing a server to use JSON patch for those resources.

The case is not as clear for incremental changes to network maps.

First, consider small changes such as moving a prefix from one PID to another. JSON patch could encode that as a simple insertion and deletion, while JSON merge patch would have to replace the entire array of prefixes for both PIDs. On the other hand, to process a JSON patch update, the ALTO client would have to retain the indexes of the prefixes for each PID. Logically, the prefixes in a PID are an unordered set, not an array; aside from handling updates, a client has no need to retain the array indexes of the prefixes. Hence to take advantage of JSON patch for network maps, ALTO clients would have to retain additional, otherwise unnecessary, data.

Second, consider more involved changes such as removing half of the prefixes from a PID. JSON merge patch would send a new array for that PID, while JSON patch would have to send a list of remove operations and delete the prefix one by one.

Therefore, each update stream server may decide on its own whether to use JSON merge patch or JSON patch according to the changes in network maps.

## 9.2. Considerations for Client Processing Data Update Messages

In general, when an ALTO client receives a full replacement for a resource, the ALTO client should replace the current version with the new version. When an ALTO client receives an incremental change for a resource, the ALTO client should apply those patches to the current version of the resource.

However, because resources can depend on other resources (e.g., cost maps depend on network maps), an ALTO client MUST NOT use a dependent resource if the resource on which it depends has changed. There are at least two ways an ALTO client can do that. The following paragraphs illustrate these techniques by referring to network and cost map messages, although these techniques apply to any dependent resources.

Note that when a network map changes, the update stream server **MUST** send the network map update message before sending the updates for the dependent cost maps (see Section 6.7.1).

One approach is for the ALTO client to save the network map update message in a buffer and continue to use the previous network map, and the associated cost maps, until the ALTO client receives the update messages for all dependent cost maps. The ALTO client then applies all network and cost map updates atomically.

Alternatively, the ALTO client **MAY** update the network map immediately. In this case, the cost maps using the network map become invalid because they are inconsistent with the current network map; hence, the ALTO client **MUST** mark each such dependent cost map as temporarily invalid and **MUST NOT** use that each such cost map until the ALTO client receives a cost map update message indicating that it is based on the new network map version tag.

The update stream server **SHOULD** send updates for dependent resources (i.e., the cost maps in the preceding example) in a timely fashion. However, if the ALTO client does not receive the expected updates, a simple recovery method is that the ALTO client closes the update stream connection, discards the dependent resources, and reestablishes the update stream. The ALTO client **MAY** retain the version tag of the last version of any tagged resources and give those version tags when requesting the new update stream. In this case, if a version is still current, the update stream server will not re-send that resource.

Although not as efficient as possible, this recovery method is simple and reliable.

### 9.3. Considerations for Updates to Filtered Cost Maps

If an update stream provides updates to a Filtered cost map which allows constraint tests, then an ALTO client **MAY** request updates to a Filtered cost map request with a constraint test. In this case, when a cost changes, the update stream server **MUST** send an update if the new value satisfies the test. If the new value does not, whether the update stream server sends an update depends on whether the previous value satisfied the test. If it did not, the update stream server **SHOULD NOT** send an update to the ALTO client. But if the previous value did, then the update stream server **MUST** send an update with a "null" value, to inform the ALTO client that this cost no longer satisfies the criteria.

An update stream server can avoid having to handle such a complicated behavior by offering update streams only for filtered cost maps which do not allow constraint tests.

#### 9.4. Considerations for Updates to Ordinal Mode Costs

For an ordinal mode cost map, a change to a single cost point may require updating many other costs. As an extreme example, suppose the lowest cost changes to the highest cost. For a numerical mode cost map, only that one cost changes. But for an ordinal mode cost map, every cost might change. While this document allows an update stream server to offer incremental updates for ordinal mode cost maps, update stream server implementors should be aware that incremental updates for ordinal costs are more complicated than for numerical costs, and ALTO clients should be aware that small changes may result in large updates.

An update stream server can avoid this complication by only offering full replacements for ordinal cost maps.

#### 9.5. Considerations for SSE Text Formatting and Processing

SSE was designed for events that consist of relatively small amounts of line-oriented text data, and SSE clients frequently read input one line-at-a-time. However, an update stream sends a full cost map as a single event, and a cost map may involve megabytes, if not tens of megabytes, of text. This has implications that the ALTO client and the update stream server may consider.

First, some SSE client libraries read all data for an event into memory, and then present it to the client as a character array. However, a client may not have enough memory to hold the entire JSON text for a large cost map. Hence an ALTO client SHOULD consider using an SSE library which presents the event data in manageable chunks, so the ALTO client can parse the cost map incrementally and store the underlying data in a more compact format.

Second, an SSE client library may use a low level, generic socket read library that stores each line of an event data, just in case the higher level parser may need the line delimiters as part of the protocol formatting. A server sending a complete cost map as a single line may then generate a multi-megabyte data "line", and such a long line may then require complex memory management at the client. It is RECOMMENDED that an update stream server limit the lengths of data lines.

Third, an SSE server may use a library which may put line breaks in places that would have semantic consequences for the ALTO updates;

see Section 11. The update stream server implementation **MUST** ensure that no line breaks are introduced to change the semantics.

## 10. Security Considerations

The Security Considerations (Section 15 of [RFC7285]) of the base protocol fully apply to this extension. For example, the same authenticity and integrity considerations (Section 15.1 of [RFC7285]) still fully apply; the same considerations for the privacy of ALTO users (Section 15.4 of [RFC7285]) also still fully apply.

The additional services (addition of update streams and stream control URIs) provided by this extension extend the attack surface described in Section 15.1.1 of [RFC7285]. Below we discuss the additional risks and their remedies.

### 10.1. Update Stream Server: Denial-of-Service Attacks

Allowing persistent update stream connections enables a new class of Denial-of-Service attacks.

For the update stream server, an ALTO client might create an unreasonable number of update stream connections, or add an unreasonable number of substream-ids to one update stream.

To avoid these attacks on the update stream server, the server **SHOULD** choose to limit the number of active streams and reject new requests when that threshold is reached. An update stream server **SHOULD** also choose to limit the number of active substream-ids on any given stream, or limit the total number of substream-ids used over the lifetime of a stream, and reject any stream control request which would exceed those limits. In these cases, the update stream server **SHOULD** return the HTTP status "503 Service Unavailable".

It is important to note that the preceding approach are not the only possibilities. For example, it may be possible for the update stream server to use somewhat more clever logic involving IP reputation, rate-limiting, and compartmentalizing the overall threshold into smaller thresholds that apply to subsets of potential clients.

While the preceding techniques prevent update stream DoS attacks from disrupting an update stream server's other services, it does make it easier for a DoS attack to disrupt the update stream service. Therefore an update stream server **MAY** prefer to restrict update stream services to authorized clients, as discussed in Section 15 of [RFC7285].

Alternatively, an update stream server MAY return the HTTP status "307 Temporary Redirect" to redirect the client to another ALTO server which can better handle a large number of update streams.

#### 10.2. ALTO Client: Update Overloading or Instability

The availability of continuous updates can also cause overload for an ALTO client, in particular an ALTO client with limited processing capabilities. The current design does not include any flow control mechanisms for the client to reduce the update rates from the server. Under overloading, the client MAY choose to remove the information resources with high update rates.

Also, under overloading, the client may no longer be able to detect whether an information is still fresh or has become stale. In such a case, the client should be careful in how it uses the information to avoid stability or efficiency issues.

#### 10.3. Stream Control: Spoofed Control Requests and Information Breakdown

An outside party which can read the update stream response, or which can observe stream control requests, can obtain the control URI and use that to send a fraudulent "remove" requests, thus disabling updates for the valid ALTO client. This can be avoided by encrypting the update stream and stream control requests (see Section 15 of [RFC7285]). Also, the update stream server echoes the "remove" requests on the update stream, so the valid ALTO client can detect unauthorized requests.

In general, as the architecture allows the possibility for the update stream server and the stream control server to be different entities, the additional risks should be evaluated and remedied. For example, the private communication path between the servers may be attacked, resulting in a risk of communications breakdown between them, as well as invalid or spoofed messages claiming to be on that private communications path. Proper security mechanisms, including confidentiality, authenticity, and integrity mechanisms should be considered.

#### 11. Requirements on Future ALTO Services to Use this Design

Although this design is quite flexible, it has underlying requirements.

The key requirements are that (1) each data update message is for a single resource; (2) an incremental change can be applied only to a resource that is a single JSON object, as both JSON merge patch and



JSON patch can apply only to a single JSON object. Hence, if a future ALTO resource can contain multiple objects, then either each individual object also has a resource-id or an extension to this design is made.

At the low level encoding level, new line in SSE has its own semantics. Hence, this design requires that resource encoding does not include new lines that can confuse with SSE encoding. In particular, the data update message MUST NOT include "event: " or "data: " at a new line as part of data message.

If an update stream provides updates to a filtered cost map that allows constraint tests, the requirements for such services are stated in Section 9.3.

## 12. IANA Considerations

This document defines two new media-types, "application/alto-updatestreamparams+json", as described in Section 6.5, and "application/alto-updatestreamcontrol+json", as described in Section 5.3. All other media-types used in this document have already been registered, either for ALTO, JSON merge patch, or JSON patch.

### 12.1. application/alto-updatestreamparams+json Media Type

Type name: application

Subtype name: alto-updatestreamparams+json

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type. See [RFC8259].

Security considerations: Security considerations relating to the generation and consumption of ALTO Protocol messages are discussed in Section 10 of [RFCthis] and Section 15 of [RFC7285].

Interoperability considerations: [RFCthis] specifies format of conforming messages and the interpretation thereof.

Published specification: Section 6.5 of [RFCthis].

Applications that use this media type: ALTO servers and ALTO clients either stand alone or are embedded within other applications.

Fragment identifier considerations: n/a

Additional information:

Magic number(s): n/a

File extension(s): [RFCthis] uses the mime type to refer to protocol messages and thus does not require a file extension.

Macintosh file type code(s): n/a

Person & email address to contact for further information: See Authors' Addresses section.

Intended usage: COMMON

Restrictions on usage: n/a

Author: See Authors' Addresses section.

Change controller: Internet Engineering Task Force  
(mailto:iesg@ietf.org).

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

## 12.2. application/alto-updatestreamcontrol+json Media Type

Type name: application

Subtype name: alto-updatestreamcontrol+json

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type. See [RFC8259].

Security considerations: Security considerations relating to the generation and consumption of ALTO Protocol messages are discussed in Section 10 of [RFCthis] and Section 15 of [RFC7285].

Interoperability considerations: [RFCthis] specifies format of conforming messages and the interpretation thereof.

Published specification: Section 5.3 of [RFCthis].

Applications that use this media type: ALTO servers and ALTO clients either stand alone or are embedded within other applications.

Fragment identifier considerations: n/a

Additional information:

Magic number(s): n/a

File extension(s): [RFCthis] uses the mime type to refer to protocol messages and thus does not require a file extension.

Macintosh file type code(s): n/a

Person & email address to contact for further information: See Authors' Addresses section.

Intended usage: COMMON

Restrictions on usage: n/a

Author: See Authors' Addresses section.

Change controller: Internet Engineering Task Force  
(mailto:iesg@ietf.org).

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

### 13. Contributors

Section 2, Section 5.1, Section 5.2 and Section 8.5 of this document are based on contributions from Jingxuan Jensen Zhang, and he is considered an author.

### 14. Acknowledgments

Thank you to Dawn Chen (Tongji University), Shawn Lin (Tongji University) and Xiao Shi (Yale University) for their contributions to an earlier version of this document.

## 15. Appendix: Design Decision: Not Allowing Stream Restart

If an update stream is closed accidentally, when the ALTO client reconnects, the update stream server must resend the full maps. This is clearly inefficient. To avoid that inefficiency, the SSE specification allows an update stream server to assign an id to each event. When an ALTO client reconnects, the ALTO client can present the id of the last successfully received event, and the update stream server restarts with the next event.

However, that mechanism adds additional complexity. The update stream server must save SSE messages in a buffer, in case ALTO clients reconnect. But that mechanism will never be perfect: if the ALTO client waits too long to reconnect, or if the ALTO client sends an invalid id, then the update stream server will have to resend the complete maps anyway.

Furthermore, this is unlikely to be a problem in practice. ALTO clients who want continuous updates for large resources, such as full Network and cost maps, are likely to be things like P2P trackers. These ALTO clients will be well connected to the network; they will rarely drop connections.

Mobile devices certainly can and do drop connections and will have to reconnect. But mobile devices will not need continuous updates for multi-megabyte cost maps. If mobile devices need continuous updates at all, they will need them for small queries, such as the costs from a small set of media servers from which the device can stream the currently playing movie. If the mobile device drops the connection and reestablishes the update stream, the update stream server will have to retransmit only a small amount of redundant data.

In short, using event ids to avoid resending the full map adds a considerable amount of complexity to avoid a situation which is very rare. The complexity is not worth the benefit.

The Update Stream service does allow the ALTO client to specify the tag of the last received version of any tagged resource, and if that is still current, the update stream server need not retransmit the full resource. Hence ALTO clients can use this to avoid retransmitting full network maps. cost maps are not tagged, so this will not work for them. Of course, the ALTO protocol could be extended by adding version tags to cost maps, which would solve the retransmission-on-reconnect problem. However, adding tags to cost maps might add a new set of complications.

## 16. References

### 16.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, BCP 14, March 1997.
- [RFC2387] Levinson, E., "The MIME Multipart/Related Content-type", RFC 2387, BCP 14, August 1998.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", RFC 6838, January 2013.
- [RFC6902] Bryan, P. and M. Nottingham, "JavaScript Object Notation (JSON) Patch", RFC 6902, April 2013.
- [RFC7285] Almi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.
- [RFC7396] Hoffman, P. and J. Snell, "JSON Merge Patch", RFC 7396, October 2014.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [SSE] Hickson, I., "Server-Sent Events (W3C)", W3C Recommendation 03 February 2015, February 2015.

### 16.2. Informative References

- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

- [RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, March 2010.
- [RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.
- [RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.
- [RFC7540] Belshe, M., Peon, R., and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, May 2015.

#### Authors' Addresses

Wendy Roome  
Nokia Bell Labs (Retired)  
124 Burlington Rd  
Murray Hill, NJ 07974  
USA

Phone: +1-908-464-6975  
Email: wendy@wdroome.com

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

Email: yry@cs.yale.edu

ALTO  
Internet-Draft  
Intended status: Experimental  
Expires: 21 September 2022

K. Gao  
Sichuan University  
Y. Lee  
Samsung  
S. Randriamasy  
Nokia Bell Labs  
Y.R. Yang  
Yale University  
J. Zhang  
Tongji University  
20 March 2022

An ALTO Extension: Path Vector  
draft-ietf-alto-path-vector-25

Abstract

This document is an extension to the base Application-Layer Traffic Optimization (ALTO) protocol. It extends the ALTO Cost Map and ALTO Property Map services so that an application can decide which endpoint(s) to connect based on not only numerical/ordinal cost values but also fine-grained abstract information of the paths. This is useful for applications whose performance is impacted by specified components of a network on the end-to-end paths, e.g., they may infer that several paths share common links and prevent traffic bottlenecks by avoiding such paths. This extension introduces a new abstraction called Abstract Network Element (ANE) to represent these components and encodes a network path as a vector of ANEs. Thus, it provides a more complete but still abstract graph representation of the underlying network(s) for informed traffic optimization among endpoints.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 21 September 2022.

## Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	4
2. Requirements Languages . . . . .	6
3. Terminology . . . . .	6
4. Requirements and Use Cases . . . . .	7
4.1. Design Requirements . . . . .	7
4.2. Sample Use Cases . . . . .	10
4.2.1. Exposing Network Bottlenecks . . . . .	11
4.2.2. Resource Exposure for CDN and Service Edge . . . . .	15
5. Path Vector Extension: Overview . . . . .	17
5.1. Abstract Network Element (ANE) . . . . .	18
5.1.1. ANE Entity Domain . . . . .	19
5.1.2. Ephemeral and Persistent ANEs . . . . .	19
5.1.3. Property Filtering . . . . .	20
5.2. Path Vector Cost Type . . . . .	20
5.3. Multipart Path Vector Response . . . . .	21
5.3.1. Identifying the Media Type of the Root Object . . . . .	22
5.3.2. References to Part Messages . . . . .	22
6. Specification: Basic Data Types . . . . .	23
6.1. ANE Name . . . . .	23
6.2. ANE Entity Domain . . . . .	23
6.2.1. Entity Domain Type . . . . .	23
6.2.2. Domain-Specific Entity Identifier . . . . .	23
6.2.3. Hierarchy and Inheritance . . . . .	23
6.2.4. Media Type of Defining Resource . . . . .	23
6.3. ANE Property Name . . . . .	24
6.4. Initial ANE Property Types . . . . .	24
6.4.1. Maximum Reservable Bandwidth . . . . .	24
6.4.2. Persistent Entity ID . . . . .	25
6.4.3. Examples . . . . .	25
6.5. Path Vector Cost Type . . . . .	26



6.5.1.	Cost Metric: ane-path . . . . .	26
6.5.2.	Cost Mode: array . . . . .	27
6.6.	Part Resource ID and Part Content ID . . . . .	27
7.	Specification: Service Extensions . . . . .	27
7.1.	Notations . . . . .	27
7.2.	Multipart Filtered Cost Map for Path Vector . . . . .	28
7.2.1.	Media Type . . . . .	28
7.2.2.	HTTP Method . . . . .	28
7.2.3.	Accept Input Parameters . . . . .	28
7.2.4.	Capabilities . . . . .	29
7.2.5.	Uses . . . . .	30
7.2.6.	Response . . . . .	30
7.3.	Multipart Endpoint Cost Service for Path Vector . . . . .	34
7.3.1.	Media Type . . . . .	34
7.3.2.	HTTP Method . . . . .	34
7.3.3.	Accept Input Parameters . . . . .	34
7.3.4.	Capabilities . . . . .	35
7.3.5.	Uses . . . . .	35
7.3.6.	Response . . . . .	35
8.	Examples . . . . .	39
8.1.	Sample Setup . . . . .	39
8.2.	Information Resource Directory . . . . .	39
8.3.	Multipart Filtered Cost Map . . . . .	42
8.4.	Multipart Endpoint Cost Service Resource . . . . .	43
8.5.	Incremental Updates . . . . .	48
8.6.	Multi-cost . . . . .	50
9.	Compatibility with Other ALTO Extensions . . . . .	52
9.1.	Compatibility with Legacy ALTO Clients/Servers . . . . .	53
9.2.	Compatibility with Multi-Cost Extension . . . . .	53
9.3.	Compatibility with Incremental Update . . . . .	53
9.4.	Compatibility with Cost Calendar . . . . .	53
10.	General Discussions . . . . .	54
10.1.	Constraint Tests for General Cost Types . . . . .	54
10.2.	General Multi-Resource Query . . . . .	54
11.	Security Considerations . . . . .	55
12.	IANA Considerations . . . . .	57
12.1.	ALTO Cost Metric Registry . . . . .	57
12.2.	ALTO Cost Mode Registry . . . . .	58
12.3.	ALTO Entity Domain Type Registry . . . . .	58
12.4.	ALTO Entity Property Type Registry . . . . .	59
12.4.1.	New ANE Property Type: Maximum Reservable Bandwidth . . . . .	59
12.4.2.	New ANE Property Type: Persistent Entity ID . . . . .	60
13.	References . . . . .	60
13.1.	Normative References . . . . .	60
13.2.	Informative References . . . . .	61
Appendix A.	Acknowledgments . . . . .	64
Appendix B.	Revision Logs (To be removed before publication) . . . . .	64

B.1.	Changes since -20 . . . . .	64
B.2.	Changes since -19 . . . . .	65
B.3.	Changes since -18 . . . . .	65
B.4.	Changes since -17 . . . . .	65
B.5.	Changes since -16 . . . . .	65
B.6.	Changes since -15 . . . . .	65
B.7.	Changes since -14 . . . . .	65
B.8.	Changes since -13 . . . . .	66
B.9.	Changes since -12 . . . . .	66
B.10.	Changes since -11 . . . . .	66
B.11.	Changes since -10 . . . . .	66
B.12.	Changes since -09 . . . . .	67
B.13.	Changes since -08 . . . . .	67
B.14.	Changes Since Version -06 . . . . .	67
Authors' Addresses	. . . . .	68

## 1. Introduction

Network performance metrics are crucial to assess the Quality of Experience (QoE) of 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 perceived QoE by better orchestrating their traffic to utilize the resources in the underlying network infrastructure.

Existing ALTO Cost Map (Section 11.2.3 of [RFC7285]) and Endpoint Cost Service (Section 11.5 of [RFC7285]) provide only cost information on an end-to-end path defined by its <source, destination> endpoints: The base protocol [RFC7285] allows the services to expose the topological distances of end-to-end paths, while various extensions have been proposed to extend the capability of these services, e.g., to express other performance metrics [I-D.ietf-alto-performance-metrics], to query multiple costs simultaneously [RFC8189], and to obtain the time-varying values [RFC8896].

While the existing extensions are sufficient for many overlay applications, the QoE of some overlay applications depends not only on the cost information of end-to-end paths, but also on particular components of a network on the paths 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 bottleneck links inside the carrier network as link capacity may impact the rate of data input/output to the job. We refer to such components of a network as Abstract Network Elements (ANE).

Predicting such information can be very complex without the help of ISPs, for example, [BOXOPT] has shown that finding the optimal bandwidth reservation for multiple flows can be NP-hard without further information than whether a reservation succeeds. With proper guidance from the ISP, an overlay application may be able to schedule its traffic for better QoE. In the meantime, it may be helpful as well for ISPs if applications could avoid using bottlenecks or challenging the network with poorly scheduled traffic.

Despite the claimed benefits, ISPs are not likely to expose raw details on their network paths: first for the sake of topology hiding requirement, second because it may increase volume and computation overhead, and last because applications do not necessarily need all the network path details and are likely not able to understand them.

Therefore, it is beneficial for both ISPs and applications if an ALTO server provides ALTO clients with an "abstract network state" that provides the necessary information to applications, while hiding the network complexity and confidential information. An "abstract network state" is a selected set of abstract representations of Abstract Network Elements traversed by the paths between <source, destination> pairs combined with properties of these Abstract Network Elements 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 Abstract Network Elements. Server scalability can also be improved by combining Abstract Network Elements and their properties in a single response.

This document extends [RFC7285] to allow an ALTO server to convey "abstract network state", for paths defined by their <source, destination> pairs. To this end, it introduces a new cost type called "Path Vector" following the cost metric registration specified in [RFC7285] and the updated cost mode registration specified in [I-D.bw-alto-cost-mode]. A Path Vector is an array of identifiers that identifies an Abstract Network Element, which 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 of an ALTO server when providing Path Vector service. 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 involves 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" content type defined in [RFC2387] to return the two maps in a single response.

As a single ISP may not have the knowledge of the full Internet paths between arbitrary endpoints, this document is mainly applicable 1) when there is a single ISP between the requested source and destination PIDs or endpoints, for example, ISP-hosted CDN/edge, tenant interconnection in a single public cloud platform, etc.; or 2) when the Path Vectors are generated from end-to-end measurement data.

## 2. Requirements Languages

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

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

## 3. Terminology

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

**Abstract Network Element (ANE):** An abstract representation for a component in a network that handles data packets and whose properties can potentially have an impact on the end-to-end performance of traffic. An ANE can be a physical device such as a router, a link or an interface, or an aggregation of devices such as a subnetwork or a data center.

The definition of Abstract Network Element is similar to Network Element defined in [RFC2216] in the sense that they both provide an abstract representation of specific components of a network. However, they have different criteria on how these particular components are selected. Specifically, a Network Element requires the components to be capable of exercising QoS control, while Abstract Network Element only requires the components to have an impact on the end-to-end performance.

**ANE Name:** A string that uniquely identifies an ANE in a specific

scope. An ANE can be constructed either statically in advance or on demand based on the requested information. Thus, different ANEs may only be valid within a particular scope, either ephemeral or persistent. Within each scope, an ANE is uniquely identified by an ANE Name, as defined in Section 6.1. Note that an ALTO client must not assume ANEs in different scopes but with the same ANE Name refer to the same component(s) of the network.

**Path Vector:** Path Vector, or ANE Path Vector, refers to a JSON array of ANE Names. It is a generalization of BGP path vector. While standard BGP path vector (Section 5.1.2 of [RFC4271]) specifies a sequence of autonomous systems for a destination IP prefix, the Path Vector defined in this extension specifies a sequence of ANEs either for a source Provider-Defined Identifier (PID) and a destination PID as in the CostMapData (11.2.3.6 in [RFC7285]), or for a source endpoint and a destination endpoint as in the EndpointCostMapData object (Section 11.5.1.6 of [RFC7285]).

**Path Vector resource:** An ALTO information resource (Section 8.1 of [RFC7285]) which supports the extension defined in this document.

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

**Path Vector request:** The POST message sent to an ALTO Path Vector resource.

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

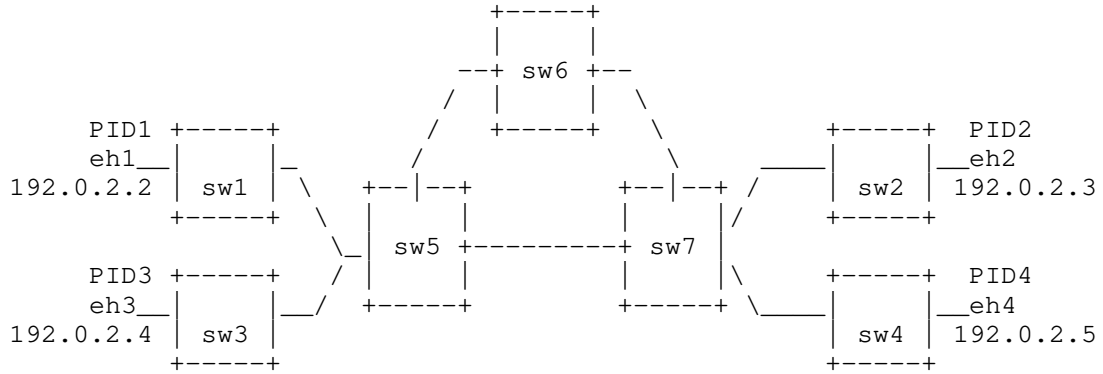
## 4. Requirements and Use Cases

### 4.1. Design Requirements

This section gives an illustrative example of how an overlay application can benefit from the extension defined in this document.

Assume that an application has control over a set of flows, which may go through shared links/nodes and share bottlenecks. The application seeks to schedule the traffic among multiple flows to get better performance. The constraints of feasible rate allocations of those flows will benefit the scheduling. However, Cost Maps as defined in [RFC7285] 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", "sw2", "sw3" and "sw4" are access switches, and sw5-sw7 form the backbone. End hosts eh1 to eh4 are connected to access switches sw1 to sw4 respectively. Assume that the bandwidth of link eh1 -> sw1 and link sw1 -> sw5 is 150 Mbps, and the bandwidth of the other links is 100 Mbps.



$\text{bw}(\text{eh1} \rightarrow \text{sw1}) = \text{bw}(\text{sw1} \rightarrow \text{sw5}) = 150 \text{ Mbps}$   
 $\text{bw}(\text{eh2} \rightarrow \text{sw2}) = \text{bw}(\text{eh3} \rightarrow \text{sw3}) = \text{bw}(\text{eh4} \rightarrow \text{sw4}) = 100 \text{ Mbps}$   
 $\text{bw}(\text{sw1} \rightarrow \text{sw5}) = \text{bw}(\text{sw3} \rightarrow \text{sw5}) = \text{bw}(\text{sw2} \rightarrow \text{sw7}) = \text{bw}(\text{sw4} \rightarrow \text{sw7}) = 100 \text{ Mbps}$   
 $\text{bw}(\text{sw5} \rightarrow \text{sw6}) = \text{bw}(\text{sw5} \rightarrow \text{sw7}) = \text{bw}(\text{sw6} \rightarrow \text{sw7}) = 100 \text{ Mbps}$

Figure 1: Raw Network Topology

The base ALTO topology abstraction of the network is shown in Figure 2. Assume the cost map returns an hypothetical cost type representing the available bandwidth between a source and a destination.



Figure 2: Base Topology Abstraction

Now assume the application wants to maximize the total rate of the traffic among a set of <source, destination> pairs, say "eh1 -> eh2" and "eh1 -> eh4". Let "x" denote the transmission rate of "eh1 -> eh2" and "y" denote the rate of "eh1 -> eh4". The objective function is

$$\max(x + y).$$

With the ALTO Cost Map, the cost between PID1 and PID2 and between PID1 and PID4 will both be 100 Mbps. The client can get a capacity region of

$$\begin{aligned} x &\leq 100 \text{ Mbps,} \\ y &\leq 100 \text{ Mbps.} \end{aligned}$$

With this information, the client may mistakenly think it can achieve a maximum total rate of 200 Mbps. However, this rate is infeasible, as there are only two potential cases:

- \* Case 1: "eh1 -> eh2" and "eh1 -> eh4" take different path segments from "sw5" to "sw7". For example, if "eh1 -> eh2" uses path "eh1 -> sw1 -> sw5 -> sw6 -> sw7 -> sw2 -> eh2" and "eh1 -> eh4" uses path "eh1 -> sw1 -> sw5 -> sw7 -> sw4 -> eh4", then the shared bottleneck links are "eh1 -> sw1" and "sw1 -> sw5". In this case, the capacity region is:

$$\begin{aligned} x &\leq 100 \text{ Mbps} \\ y &\leq 100 \text{ Mbps} \\ x + y &\leq 150 \text{ Mbps} \end{aligned}$$

and the real optimal total rate is 150 Mbps.

- \* Case 2: "eh1 -> eh2" and "eh1 -> eh4" take the same path segment from "sw5" to "sw7". For example, if "eh1 -> eh2" uses path "eh1 -> sw1 -> sw5 -> sw7 -> sw2 -> eh2" and "eh1 -> eh4" also uses path "eh1 -> sw1 -> sw5 -> sw7 -> sw4 -> eh4", then the shared bottleneck link is "sw5 -> sw7". In this case, the capacity region is:

$$\begin{aligned} x &\leq 100 \text{ Mbps} \\ y &\leq 100 \text{ Mbps} \\ x + y &\leq 100 \text{ Mbps} \end{aligned}$$

and the real optimal total rate is 100 Mbps.

Clearly, with more accurate and fine-grained information, the application can gain a better prediction of its traffic and may orchestrate its resources accordingly. However, to provide such information, the network needs to expose abstract information beyond the simple cost map abstraction. In particular:

- \* The ALTO server must expose abstract information about the network paths that are traversed by the traffic between a source and a destination beyond a simple numerical value, which allows the overlay application to distinguish between Cases 1 and 2 and to compute the optimal total rate accordingly.
- \* The ALTO server must allow the client to distinguish the common ANE shared by "eh1 -> eh2" and "eh1 -> eh4", e.g., "eh1 - sw1" and "sw1 - sw5" in Case 1.
- \* The ALTO server must expose abstract information on the properties of the ANEs used by "eh1 -> eh2" and "eh1 -> eh4". For example, an ALTO server can either expose the available bandwidth between "eh1 - sw1", "sw1 - sw5", "sw5 - sw7", "sw5 - sw6", "sw6 - sw7", "sw7 - sw2", "sw7 - sw4", "sw2 - eh2", "sw4 - eh4" in Case 1, or expose 3 abstract elements "A", "B" and "C", which represent the linear constraints that define the same capacity region in Case 1.

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

AR1: An ALTO server must provide the ANEs that are important to assess the QoE of the overlay application on the path of a <source, destination> pair.

AR2: An ALTO server must provide information to identify how ANEs are shared on the paths of different <source, destination> pairs.

AR3: An ALTO server must provide information on the properties that are important to assess the QoE of the application for ANEs.

The extension defined in this document specifies a solution to expose such abstract information.

#### 4.2. Sample Use Cases

While the multiple flow scheduling problem is used to help identify the additional requirements, the extension defined in this document can be applied to a wide range of applications. This section highlights some use cases that are reported.



#### 4.2.1. Exposing Network Bottlenecks

An important use case of the Path Vector extension is to expose network bottlenecks. Applications which need to perform large scale data transfers can benefit from being aware of the resource constraints exposed by this extension even if they have different objectives. One such example is the Worldwide LHC Computing Grid (WLCG), the largest example of a distributed computation collaboration in the research and education world.

Figure 3 illustrates an example of using ALTO Path Vector as an interface between the job optimizer for a data analytics system and the network manager. In particular, we assume the objective of the job optimizer is to minimize the job completion time.

In such a setting, the network-aware job optimizer (e.g., [CLARINET]) takes a query and generates multiple query execution plans (QEP). It can encode the QEPs as Path Vector requests that are sent to an ALTO server. The ALTO server obtains the routing information for the flows in a QEP and finds links, routers, or middleboxes (e.g., a stateful firewall) that can potentially become bottlenecks of the QEP (e.g., see [NOVA] and [G2] for mechanisms to identify bottleneck links under different settings). The resource constraint information is encoded in a Path Vector response and returned to the ALTO client.

With the network resource constraints, the job optimizer may choose the QEP with the optimal job completion time to be executed. It must be noted that the ALTO framework itself does not offer the capability to control the traffic. However, certain network managers may offer ways to enforce resource guarantees, such as on-demand tunnels (e.g., [SWAN]), demand vector (e.g., [HUG], [UNICORN]), etc. The traffic control interfaces and mechanisms are out of the scope of this document.

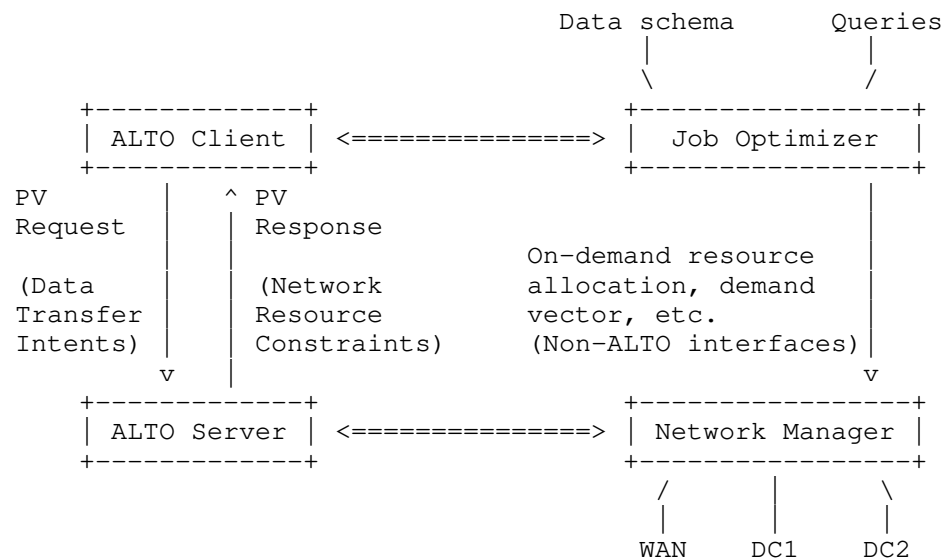


Figure 3: Example Use Case for Data Analytics

Another example is as illustrated in Figure 4. Consider a network consisting of multiple sites and a non-blocking core network, i.e., the links in the core network have sufficient bandwidth that they will not become the bottleneck of the data transfers.

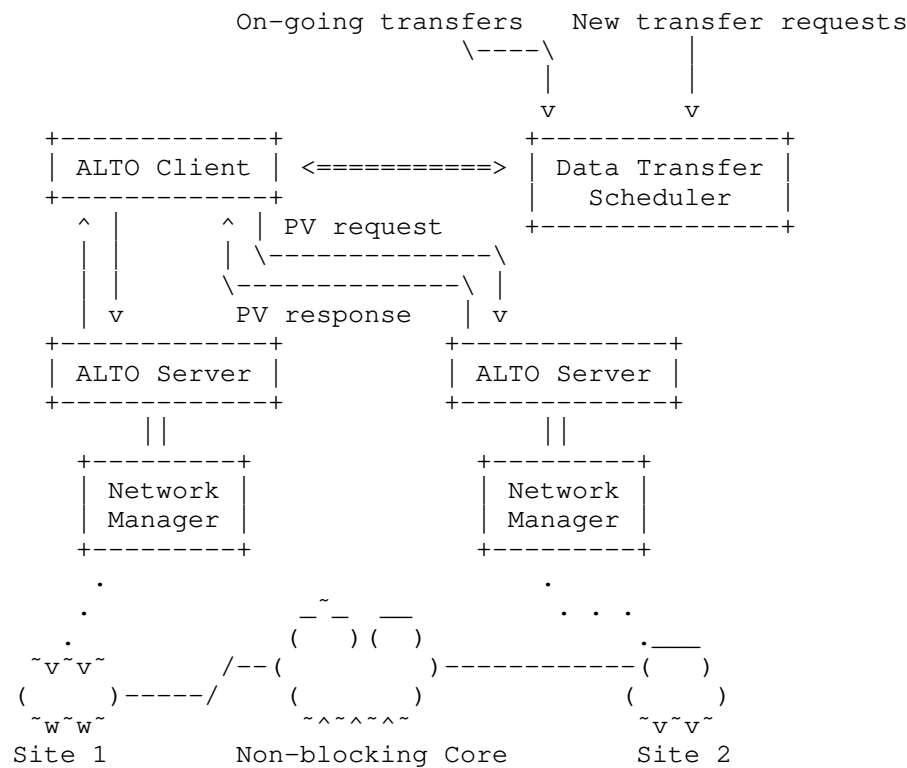
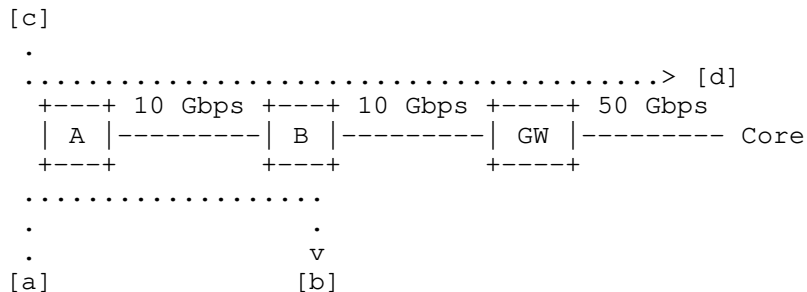


Figure 4: Example Use Case for Cross-site Bottleneck Discovery

Site 1:



Site 2:

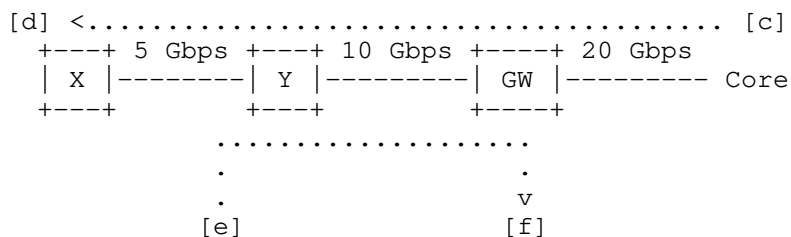


Figure 5: Example: Three Flows in Two Sites

With the Path Vector extension, a site can reveal the bottlenecks inside its own network with necessary information (such as link capacities) to the ALTO client, instead of providing the full topology and routing information, or no bottleneck information at all. The bottleneck information can be used to analyze the impact of adding/removing data transfer flows, e.g., using the [G2] framework. For example, assume hosts "a", "b", "c" are in site 1 and hosts "d", "e", "f" are in site 2, and there are 3 flows in two sites: "a -> b", "c -> d", "e -> f". For these flows, site 1 returns:

```

a: { b: [ane1] },
c: { d: [ane1, ane2, ane3] }

ane1: bw = 10 Gbps (link: A->B)
ane2: bw = 10 Gbps (link: B->GW)
ane3: bw = 50 Gbps (link: GW->Core)

```

and site 2 returns:

```
c: { d: [anei, aneii, aneiii] }  
e: { f: [aneiv] }
```

```
anei: bw = 5 Gbps (link Y->X)  
aneii: bw = 10 Gbps (link GW->Y)  
aneiii: bw = 20 Gbps (link Core->GW)  
aneiv: bw = 10 Gbps (link Y->GW)
```

With the information, the data transfer scheduler can use algorithms such as the theory on bottleneck structure [G2] to predict the potential throughput of the flows.

#### 4.2.2. Resource Exposure for CDN and Service Edge

A growing trend in today's applications (2021) is to bring storage and computation closer to the end users for better QoE, such as Content Delivery Network (CDN), AR/VR, and cloud gaming, as reported in various documents (e.g., [SEREDGE] and [MOWIE]). Internet Service Providers may deploy multiple layers of CDN caches, or more generally service edges, with different latency and available resources including number of CPU cores, memory, and storage.

For example, Figure 6 illustrates a typical edge-cloud scenario where memory is measured in Gigabytes (G) and storage is measured in Terabytes (T). The "on-premise" edge nodes are closest to the end hosts and have the smallest latency, and the site-radio edge node and access central office (CO) have larger latency but more available resources.

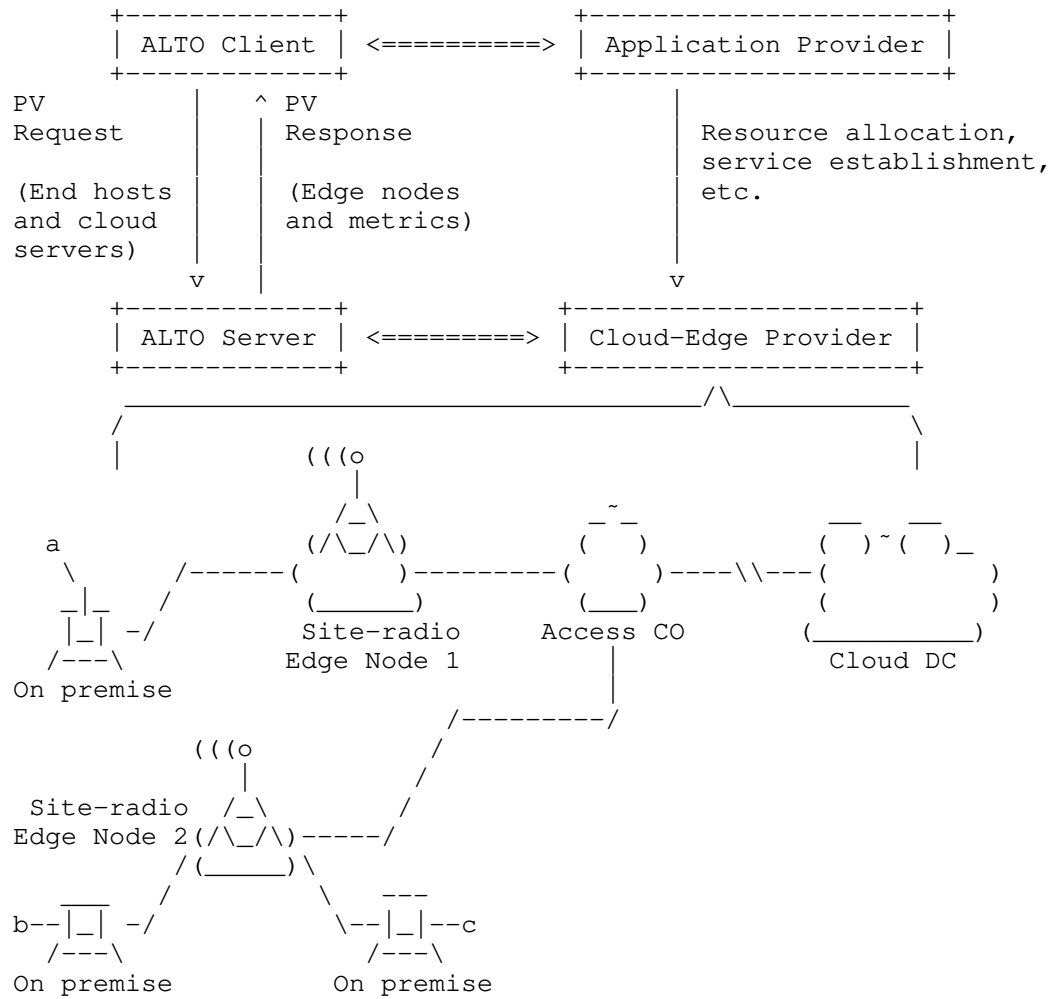


Figure 6: Example Use Case for Service Edge Exposure

```
a: { b: [ane1, ane2, ane3, ane4, ane5],  
      c: [ane1, ane2, ane3, ane4, ane6],  
      DC: [ane1, ane2, ane3] }  
b: { c: [ane5, ane4, ane6], DC: [ane5, ane4, ane3] }  
  
ane1: latency=5ms cpu=2 memory=8G storage=10T  
(on premise, a)  
  
ane2: latency=20ms cpu=4 memory=8G storage=10T  
(Site-radio Edge Node 1)  
  
ane3: latency=100ms cpu=8 memory=128G storage=100T  
(Access CO)  
  
ane4: latency=20ms cpu=4 memory=8G storage=10T  
(Site-radio Edge Node 2)  
  
ane5: latency=5ms cpu=2 memory=8G storage=10T  
(on premise, b)  
  
ane6: latency=5ms cpu=2 memory=8G storage=10T  
(on premise, c)
```

Figure 7: Example Service Edge Query Results

With the extension defined in this document, an ALTO server can selectively reveal the CDNs and service edges that reside along the paths between different end hosts and/or the cloud servers, together with their properties such as capabilities (e.g., storage, GPU) and available Service Level Agreement (SLA) plans. See Figure 7 for an example where the query is made for sources [a, b] and destinations [b, c, DC]. Here each ANE represents a service edge and the properties include access latency, available resources, etc. Note the properties here are only used for illustration purposes and are not part of this extension.

With the service edge information, an ALTO client may better conduct CDN request routing or offload functionalities from the user equipment to the service edge, with considerations on customized quality of experience.

## 5. Path Vector Extension: Overview

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

To satisfy the additional requirements listed in Section 4.1, this extension:

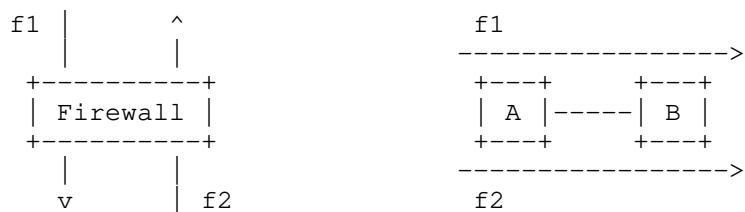
1. introduces the concept of Abstract Network Element (ANE) as the abstraction of components in a network whose properties may have an impact on the end-to-end performance of the traffic handled by those components,
2. extends the Cost Map and Endpoint Cost Service to convey the ANEs traversed by the path of a <source, destination> pair as Path Vectors, and
3. uses the Unified Property Map to convey the association between the ANEs and their properties.

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

#### 5.1. Abstract Network Element (ANE)

This extension introduces ANE as an indirect and network-agnostic way to specify a component or an aggregation of components of a network whose properties have an impact on the end-to-end performance for application traffic between endpoints.

ANEs allow ALTO servers to focus on common properties of different types of network components. For example, the throughput of a flow can be constrained by different components in a network: the capacity of a physical link, the maximum throughput of a firewall, the reserved bandwidth of an MPLS tunnel, etc. See the example below, assume the throughput of the firewall is 100 Mbps and the capacity for link (A, B) is also 100 Mbps, they result in the same constraint on the total throughput of f1 and f2. Thus, they are identical when treated as an ANE.





When an ANE is defined by an ALTO server, it is assigned an identifier by the ALTO server, i.e., a string of type ANEName as specified in Section 6.1, and a set of associated properties.

#### 5.1.1. ANE Entity Domain

In this extension, the associations between ANE and the properties are conveyed in a Unified Property Map. Thus, ANEs must constitute an entity domain (Section 5.1 of [I-D.ietf-alto-unified-props-new]), and each ANE property must be an entity property (Section 5.2 of [I-D.ietf-alto-unified-props-new]).

Specifically, this document defines a new entity domain called "ane" as specified in Section 6.2 and defines two initial properties for the ANE entity domain.

#### 5.1.2. Ephemeral and Persistent ANEs

By design, ANEs are ephemeral and not to be used in further requests to other ALTO resources. More precisely, the corresponding ANE names are no longer valid beyond the scope of a Path Vector response or the incremental update stream for a Path Vector request. Compared with globally unique ANE names, ephemeral ANE has several benefits including better privacy of the ISP's internal structure and more flexible ANE computation.

For example, an ALTO server may define an ANE for each aggregated bottleneck link between the sources and destinations specified in the request. For requests with different sources and destinations, the bottlenecks may be different but can safely reuse the same ANE names. The client can still adjust its traffic based on the information but is difficult to infer the underlying topology with multiple queries.

However, sometimes an ISP may intend to selectively reveal some "persistent" network components which, opposite to being ephemeral, have a longer life cycle. For example, an ALTO server may define an ANE for each service edge cluster. Once a client chooses to use a service edge, e.g., by deploying some user-defined functions, it may want to stick to the service edge to avoid the complexity of state transition or synchronization, and continuously query the properties of the edge cluster.

This document provides a mechanism to expose such network components as persistent ANEs. A persistent ANE has a persistent ID that is registered in a Property Map, together with their properties. See Section 6.2.4 and Section 6.4.2 for more detailed instructions on how to identify ephemeral ANEs and persistent ANEs.

### 5.1.3. Property Filtering

Resource-constrained ALTO clients (see Section 4.1.2 of [RFC7285]) may benefit from the filtering of Path Vector query results at the ALTO server, as an ALTO client may only require a subset of the available properties.

Specifically, the available properties for a given resource are announced in the Information Resource Directory as a new capability called "ane-property-names". The properties selected by a client as being of interest are specified in the subsequent Path Vector queries using the filter called 'ane-property-names'. The response includes and only includes the selected properties for the ANEs in the response.

The "ane-property-names" capability for Cost Map and for Endpoint Cost Service is specified in Section 7.2.4 and Section 7.3.4 respectively. The "ane-property-names" filter for Cost Map and Endpoint Cost Service is specified in Section 7.2.3 and Section 7.3.3 accordingly.

### 5.2. Path Vector Cost Type

For an ALTO client to correctly interpret the Path Vector, this extension specifies a new cost type called the Path Vector cost type.

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 C++ where there existed a JSON array type named JSONArray, a Path Vector will have the type of JSONArray<ANENAME>.

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

The Path Vector cost type is specified in Section 6.5.

### 5.3. Multipart Path Vector Response

For a basic ALTO information resource, a response contains only one type of ALTO resources, e.g., Network Map, Cost Map, or Property Map. Thus, only one round of communication is required: An ALTO client sends a request to an ALTO server, and the ALTO server returns a response, as shown in Figure 8.

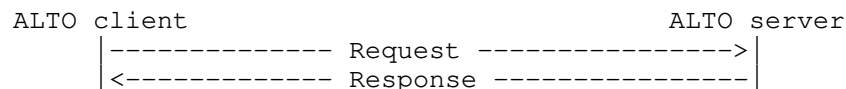


Figure 8: A Typical ALTO Request and Response

The extension defined in this document, on the other hand, involves two types of information resources: Path Vectors conveyed in an `InfoResourceCostMap` (defined in Section 11.2.3.6 of [RFC7285]) or an `InfoResourceEndpointCostMap` (defined in Section 11.5.1.6 of [RFC7285]), and ANE properties conveyed in an `InfoResourceProperties` (defined in Section 7.6 of [I-D.ietf-alto-unified-props-new]).

Instead of two consecutive message exchanges, the extension defined in this document enforces one round of communication. Specifically, the ALTO client must include the source and destination pairs and the requested ANE properties in a single request, and the ALTO server must return a single response containing both the Path Vectors and properties associated with the ANEs in the Path Vectors, as shown in Figure 9. Since the two parts are bundled together in one response message, their orders are interchangeable. See Section 7.2.6 and Section 7.3.6 for details.

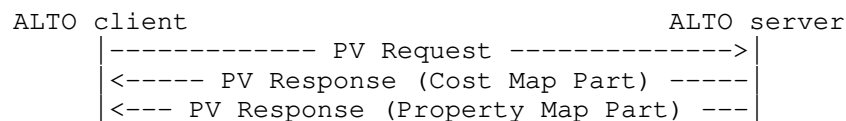


Figure 9: The Path Vector Extension Request and Response

This design is based on the following considerations:

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

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

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

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

ALTO uses media type to indicate the type of an entry in the Information Resource Directory (IRD) (e.g., "application/alto-costmap+json" for Cost Map and "application/alto-endpointcost+json" for Endpoint Cost Service). 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" field in the IRD entry is "multipart/related" with the parameter "type=application/alto-costmap+json"; for an Endpoint Cost Service, the parameter is "type=application/alto-endpointcost+json".

#### 5.3.2. References to Part Messages

As the response of a Path Vector resource is a multipart message with two different parts, it is important that each part can be uniquely identified. Following the designs of [RFC8895], this extension requires that an ALTO server assigns a unique identifier to each part of the multipart response message. This identifier, referred to as a Part Resource ID (See Section 6.6 for details), is present in the part message's "Content-ID" header. By concatenating the Part Resource ID to the identifier of the Path Vector request, an ALTO server/client can uniquely identify the Path Vector Part or the

Property Map part.

## 6. Specification: Basic Data Types

### 6.1. ANE Name

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

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

### 6.2. ANE Entity Domain

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

It must be noted that the term "domain" here does not refer to a network domain. Rather, it is inherited from the "entity domain" defined in Sec 3.2 in [I-D.ietf-alto-unified-props-new] that represents the set of valid entities defined by an ALTO information resource (called the defining information resource).

#### 6.2.1. Entity Domain Type

The Entity Domain Type is "ane".

#### 6.2.2. Domain-Specific Entity Identifier

The entity identifiers are the ANE Names in the associated Property Map.

#### 6.2.3. Hierarchy and Inheritance

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

#### 6.2.4. Media Type of Defining Resource

The defining resource for entity domain type "ane" MUST be a Property Map, i.e., the media type of defining resources is:

application/alto-propmap+json

Specifically, for ephemeral ANEs that appear in a Path Vector response, their entity domain names MUST be exactly ".ane" and the defining resource of these ANEs is the Property Map part of the

multipart response. Meanwhile, for any persistent ANE whose defining resource is a Property Map resource, its entity domain name MUST have the format of "PROPMAP.ane" where PROPMAP is the resource ID of the defining resource. Persistent entities are "persistent" because standalone queries can be made by an ALTO client to their defining resource(s) when the connection to the Path Vector service is closed.

For example, the defining resource of an ephemeral ANE whose entity identifier is ".ane:NET1" is the Property Map part that contains this identifier. The defining resource of a persistent ANE whose entity identifier is "dc-props.ane:DC1" is the Property Map with the resource ID "dc-props".

### 6.3. ANE Property Name

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

### 6.4. Initial ANE Property Types

Two initial ANE property types are specified, "max-reservable-bandwidth" and "persistent-entity-id".

Note that these property types do not depend on any information resource. As such, the EntityPropertyName MUST only have the EntityPropertyType part.

#### 6.4.1. Maximum Reservable Bandwidth

The maximum reservable bandwidth property ("max-reservable-bandwidth") stands for the maximum bandwidth that can be reserved for all the traffic that traverses an ANE. The value MUST be encoded as a non-negative numerical cost value as defined in Section 6.1.2.1 of [RFC7285] and the unit is bit per second (bps). If this property is requested by the ALTO client but not present for an ANE in the server response, it MUST be interpreted as that the property is not defined for the ANE.

This property can be offered in a setting where the ALTO server is part of a network system that provides on-demand resource allocation and the ALTO client is part of a user application. One existing example is [NOVA]: the ALTO server is part of an SDN controller and exposes a list of traversed network elements and associated link bandwidth to the client. The encoding in [NOVA] differs from the Path Vector response defined in this document that the Path Vector part and Property Map part are put in the same JSON object.

In such a framework, the ALTO server exposes resource (e.g., reservable bandwidth) availability information to the ALTO client. How the client makes resource requests based on the information and how the resource allocation is achieved respectively depend on interfaces between the management system and the users or a higher-layer protocol (e.g., SDN network intents or MPLS tunnels), which are out of the scope of this document.

#### 6.4.2. Persistent Entity ID

The persistent entity ID property is the entity identifier of the persistent ANE which an ephemeral ANE presents (See Section 5.1.2 for details). The value of this property is encoded with the format EntityID defined in Section 5.1.3 of [I-D.ietf-alto-unified-props-new].

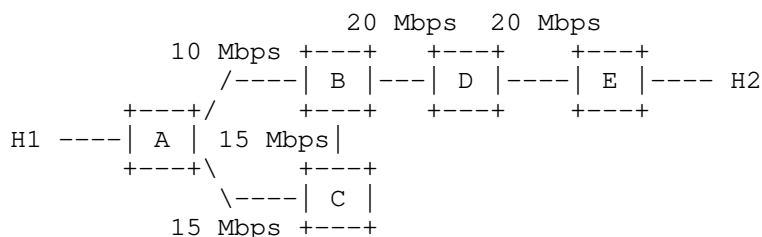
In this format, the entity ID combines:

- \* a defining information resource for the ANE on which a "persistent-entity-id" is queried, which is the Property Map resource defining the ANE as a persistent entity, together with the properties;
- \* the persistent name of the ANE in that Property Map.

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

#### 6.4.3. Examples

To illustrate the use of "max-reservable-bandwidth", consider the following network with 5 nodes. Assume the client wants to query the maximum reservable bandwidth from H1 to H2. An ALTO server may split the network into two ANEs: "ane1" that represents the subnetwork with routers A, B, and C, and "ane2" that represents the subnetwork with routers B, D and E. The maximum reservable bandwidth for "ane1" is 15 Mbps (using path A->C->B) and the maximum reservable bandwidth for "ane2" is 20 Mbps (using path B->D->E).



To illustrate the use of "persistent-entity-id", consider the scenario in Figure 6. As the life cycle of service edges are typically long, they may contain information that is not specific to the query. Such information can be stored in an individual unified property map and later be accessed by an ALTO client.

For example, "ane1" in Figure 7 represents the on-premise service edge closest to host a. Assume the properties of the service edges are provided in a unified property map called "se-props" and the ID of the on-premise service edge is "9a0b55f7-7442-4d56-8a2c-b4cc6a8e3aa1", the "persistent-entity-id" of "ane1" will be "se-props.ane:9a0b55f7-7442-4d56-8a2c-b4cc6a8e3aa1". With this persistent entity ID, an ALTO client may send queries to the "se-props" resource with the entity ID ".ane:9a0b55f7-7442-4d56-8a2c-b4cc6a8e3aa1".

#### 6.5. Path Vector Cost Type

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

##### 6.5.1. Cost Metric: ane-path

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

An ALTO client MUST interpret the Path Vector as if the traffic between a source and a destination logically traverses the ANEs in the same order as they appear in the Path Vector.

When the Path Vector procedures defined in this document are in use, an ALTO server using the "ane-path" cost metric and the "array" cost mode (see Section 6.5.2) MUST return as the cost value a JSON array of ANEName and the client MUST also check that each element contained in the array is an ANEName (Section 6.1). Otherwise, the client MUST discard the response and SHOULD follow the instructions in Section 8.3.4.3 of [RFC7285] to handle the error.



### 6.5.2. Cost Mode: array

The cost mode "array" indicates that every cost value in the response body of a (Filtered) Cost Map or an Endpoint Cost Service MUST be interpreted as a JSON array object. While this cost mode can be applied to all cost metrics, additional specifications will be needed to clarify the semantics of the array cost mode when combined with cost metrics other than 'ane-path'.

### 6.6. Part Resource ID and Part Content ID

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

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

A Part Content ID conforms to the format of msg-id as specified in [RFC2387] and [RFC5322]. Specifically, it has the following format:

"<" PART-RESOURCE-ID "@" DOMAIN-NAME ">"

**PART-RESOURCE-ID:** PART-RESOURCE-ID has the same format as the Part Resource ID. It is used to identify whether a part message is a Path Vector or a Property Map.

**DOMAIN-NAME:** DOMAIN-NAME has the same format as dot-atom-text specified in Section 3.2.3 of [RFC5322]. It must be the domain name of the ALTO server.

## 7. Specification: Service Extensions

### 7.1. Notations

This document uses the same syntax and notations as introduced in Section 8.2 of RFC 7285 [RFC7285] to specify the extensions to existing ALTO resources and services.

## 7.2. 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 with the Cost Map resource in the same response.

### 7.2.1. Media Type

The media type of the multipart Filtered Cost Map resource is "multipart/related" and the required "type" parameter MUST have a value of "application/alto-costmap+json".

### 7.2.2. HTTP Method

The multipart Filtered Cost Map is requested using the HTTP POST method.

### 7.2.3. 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, which is defined as a JSON object of type ReqFilteredCostMap in Section 4.1.2 of RFC 8189 [RFC8189], with a data format indicated by the media type "application/alto-costmapfilter+json", which is a JSON object of type PVReqFilteredCostMap:

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

with fields:

ane-property-names: A list of selected ANE properties to be included in the response. Each property in this list MUST match one of the supported ANE properties indicated in the resource's "ane-property-names" capability (Section 7.2.4). If the field is not present, it MUST be interpreted as an empty list.

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

```
POST /costmap/pv HTTP/1.1
Host: alto.example.com
Accept: multipart/related;type=application/alto-costmap+json,
       application/alto-error+json
Content-Length: 201
Content-Type: application/alto-costmapfilter+json

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

#### 7.2.4. Capabilities

The multipart Filtered Cost Map resource extends the capabilities defined in Section 4.1.1 of [RFC8189]. The capabilities are defined by a JSON object of type PVFilteredCostMapCapabilities:

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

with fields:

**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.

This extension also introduces additional restrictions for the following fields:

**cost-type-names:** The "cost-type-names" field MUST 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 (Section 4.1.1 of [RFC8189]): If the "cost-type-names" field includes the Path Vector cost type and the "testable-cost-type-names" field is present, the Path Vector cost type MUST NOT be included in the "testable-cost-type-names" field unless specifically instructed by a future document.

#### 7.2.5. Uses

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

#### 7.2.6. 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 is mandatory and MUST be "application/alto-costmap+json". Note that [RFC2387] permits both parameters with and without the double quotes.

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

boundary: The boundary parameter is as defined in Section 5.1.1 of [RFC2046] and is mandatory.

The body of the response MUST consist of two parts:

- \* The Path Vector part MUST include "Content-ID" and "Content-Type" in its header. The "Content-Type" MUST be "application/alto-costmap+json". The value of "Content-ID" MUST have the same format as the Part Content ID as specified in Section 6.6.

The body of the Path Vector part MUST be a JSON object with the same format as defined in Section 11.2.3.6 of [RFC7285] when the "cost-type" field is present in the input parameters and MUST be a JSON object with the same format as defined in Section 4.1.3 of [RFC8189] if the "multi-cost-types" field is present. The JSON object MUST include the "vtag" field in the "meta" field, which provides the version tag of the returned CostMapData. The resource ID of the version tag MUST follow the format of

resource-id '.' part-resource-id

where "resource-id" is the resource Id of the Path Vector resource, and "part-resource-id" has the same value as the PART-RESOURCE-ID in the "Content-ID" of the Path Vector part. The "meta" field MUST also include the "dependent-vtags" field, whose value is a single-element array to indicate the version tag of the network map used, where the network map is specified in the "uses" attribute of the multipart Filtered Cost Map resource in IRD.

- \* The Unified Property Map part MUST also include "Content-ID" and "Content-Type" in its header. The "Content-Type" MUST be "application/alto-propmap+json". The value of "Content-ID" MUST have the same format as the Part Content ID as specified in Section 6.6.

The body of the Unified Property Map part is a JSON object with the same format as defined in Section 7.6 of [I-D.ietf-alto-unified-props-new]. The JSON object MUST include the "dependent-vtags" field in the "meta" field. The value of the "dependent-vtags" field MUST be an array of VersionTag objects as defined by Section 10.3 of [RFC7285]. The "vtag" of the Path Vector part MUST be included in the "dependent-vtags". If "persistent-entity-id" is requested, the version tags of the dependent resources that may expose the entities in the response MUST also be included.

The PropertyMapData has one member for each ANENAME that appears in the Path Vector part, which is an entity identifier belonging to the self-defined entity domain as defined in Section 5.1.2.3 of [I-D.ietf-alto-unified-props-new]. The EntityProps for each ANE has one member for each property that is both 1) associated with the ANE, and 2) specified in the "ane-property-names" in the request. If the Path Vector cost type is not included in the "cost-type" field or the "multi-cost-type" field, the "property-map" field MUST be present and the value MUST be an empty object ({}).

A complete and valid response MUST include both the Path Vector part and the Property Map part in the multipart message. If any part is NOT present, the client MUST discard the received information and send another request if necessary.

According to [RFC2387], the Path Vector part, whose media type is the same as the "type" parameter of the multipart response message, is the root object. Thus, it is the element the application processes first. Even though the "start" parameter allows it to be placed anywhere in the part sequence, it is RECOMMENDED that the parts

arrive in the same order as they are processed, i.e., the Path Vector part is always put as the first part, followed by the Property Map part. When doing so, an ALTO server MAY choose not to set the "start" parameter, which implies the first part is the root object.

Example: Consider the network in Figure 1. The response of the example request in Section 7.2.3 is as follows, where "ANE1" represents the aggregation of all the switches in the network.

```
HTTP/1.1 200 OK
Content-Length: 859
Content-Type: multipart/related; boundary=example-1;
              type=application/alto-costmap+json

--example-1
Content-ID: <costmap@alto.example.com>
Content-Type: application/alto-costmap+json

{
  "meta": {
    "vtag": {
      "resource-id": "filtered-cost-map-pv.costmap",
      "tag": "fb20b76204814e9db37a51151faaaef2"
    },
    "dependent-vtags": [
      {
        "resource-id": "my-default-networkmap",
        "tag": "75ed013b3cb58f896e839582504f6228"
      }
    ],
    "cost-type": { "cost-mode": "array", "cost-metric": "ane-path" }
  },
  "cost-map": {
    "PID1": { "PID2": ["ANE1"] }
  }
}

--example-1
Content-ID: <propmap@alto.example.com>
Content-Type: application/alto-propmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "filtered-cost-map-pv.costmap",
        "tag": "fb20b76204814e9db37a51151faaaef2"
      }
    ]
  },
  "property-map": {
    ".ane:ANE1": { "max-reservable-bandwidth": 100000000 }
  }
}
```

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

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

#### 7.3.1. Media Type

The media type of the multipart Endpoint Cost Service resource is "multipart/related" and the required "type" parameter MUST have a value of "application/alto-endpointcost+json".

#### 7.3.2. HTTP Method

The multipart Endpoint Cost Service resource is requested using the HTTP POST method.

#### 7.3.3. Accept Input Parameters

The input parameters of the multipart Endpoint Cost Service resource are supplied in the body of an HTTP POST request. This document extends the input parameters to an Endpoint Cost Service, which is defined as a JSON object of type ReqEndpointCost in Section 4.2.2 of [RFC8189], with a data format indicated by the media type "application/alto-endpointcostparams+json", which is a JSON object of type PVReqEndpointCost:

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

with fields:

ane-property-names: This document defines the "ane-property-names" in PVReqEndpointcost as the same as in PVReqFilteredCostMap. See Section 7.2.3.

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



```
POST /ecs/pv HTTP/1.1
Host: alto.example.com
Accept: multipart/related;type=application/alto-endpointcost+json,
       application/alto-error+json
Content-Length: 227
Content-Type: application/alto-endpointcostparams+json
```

```
{
  "cost-type": {
    "cost-mode": "array",
    "cost-metric": "ane-path"
  },
  "endpoints": {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [ "ipv4:192.0.2.18" ]
  },
  "ane-property-names": [ "max-reservable-bandwidth" ]
}
```

#### 7.3.4. Capabilities

The capabilities of the multipart Endpoint Cost Service resource are defined by a JSON object of type `PVEndpointcostCapabilities`, which is defined as the same as `PVFilteredCostMapCapabilities`. See Section 7.2.4.

#### 7.3.5. Uses

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

#### 7.3.6. 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" and is mandatory.

**start:** The start parameter is as defined in Section 7.2.6.

**boundary:** The boundary parameter is as defined in Section 5.1.1 of [RFC2046] and is mandatory.

The body MUST consist of two parts:

- \* The Path Vector part MUST include "Content-ID" and "Content-Type" in its header. The "Content-Type" MUST be "application/alto-endpointcost+json". The value of "Content-ID" MUST have the same format as the Part Content ID as specified in Section 6.6.

The body of the Path Vector part MUST be a JSON object with the same format as defined in Section 11.5.1.6 of [RFC7285] when the "cost-type" field is present in the input parameters and MUST be a JSON object with the same format as defined in Section 4.2.3 of [RFC8189] if the "multi-cost-types" field is present. The JSON object MUST include the "vtag" field in the "meta" field, which provides the version tag of the returned EndpointCostMapData. The resource ID of the version tag MUST follow the format of

resource-id '.' part-resource-id

where "resource-id" is the resource Id of the Path Vector resource, and "part-resource-id" has the same value as the PART-RESOURCE-ID in the "Content-ID" of the Path Vector part.

- \* The Unified Property Map part MUST also include "Content-ID" and "Content-Type" in its header. The "Content-Type" MUST be "application/alto-propmap+json". The value of "Content-ID" MUST have the same format as the Part Content ID as specified in Section 6.6.

The body of the Unified Property Map part MUST be a JSON object with the same format as defined in Section 7.6 of [I-D.ietf-alto-unified-props-new]. The JSON object MUST include the "dependent-vtags" field in the "meta" field. The value of the "dependent-vtags" field MUST be an array of VersionTag objects as defined by Section 10.3 of [RFC7285]. The "vtag" of the Path Vector part MUST be included in the "dependent-vtags". If "persistent-entity-id" is requested, the version tags of the dependent resources that may expose the entities in the response MUST also be included.

The PropertyMapData has one member for each ANENAME that appears in the Path Vector part, which is an entity identifier belonging to the self-defined entity domain as defined in Section 5.1.2.3 of [I-D.ietf-alto-unified-props-new]. The EntityProps for each ANE has one member for each property that is both 1) associated with the ANE, and 2) specified in the "ane-property-names" in the request. If the Path Vector cost type is not included in the "cost-type" field or the "multi-cost-type" field, the "property-map" field MUST be present and the value MUST be an empty object ({}).

A complete and valid response MUST include both the Path Vector part and the Property Map part in the multipart message. If any part is NOT present, the client MUST discard the received information and send another request if necessary.

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

Example: Consider the network in Figure 1. The response of the example request in Section 7.3.3 is as follows.

```
HTTP/1.1 200 OK
Content-Length: 845
Content-Type: multipart/related; boundary=example-1;
             type=application/alto-endpointcost+json
```

```
--example-1
Content-ID: <ecs@alto.example.com>
Content-Type: application/alto-endpointcost+json
```

```
{
  "meta": {
    "vtag": {
      "resource-id": "ecs-pv.ecs",
      "tag": "ec137bb78118468c853d5b622ac003f1"
    },
    "dependent-vtags": [
      {
        "resource-id": "my-default-networkmap",
        "tag": "677fe5f4066848d282ece213a84f9429"
      }
    ],
    "cost-type": { "cost-mode": "array", "cost-metric": "ane-path" },
    "cost-map": {
      "ipv4:192.0.2.2": { "ipv4:192.0.2.18": ["ANE1"] }
    }
  }
}
```

```
--example-1
Content-ID: <propmap@alto.example.com>
Content-Type: application/alto-propmap+json
```

```
{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "ecs-pv.ecs",
        "tag": "ec137bb78118468c853d5b622ac003f1"
      }
    ]
  },
  "property-map": {
    ".ane:ANE1": { "max-reservable-bandwidth": 100000000 }
  }
}
```

## 8. Examples

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

### 8.1. Sample Setup

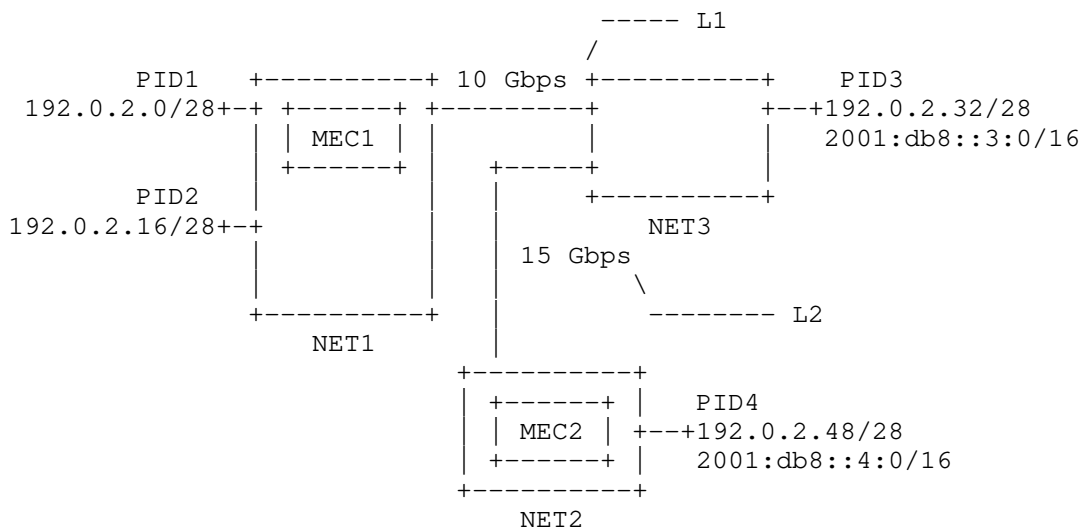


Figure 10: Examples of ANE Properties

In this document, Figure 10 is used to illustrate the message contents. There are 3 sub-networks (NET1, NET2 and NET3) and two interconnection links (L1 and L2). It is assumed that each sub-network has sufficiently large bandwidth to be reserved.

### 8.2. Information Resource Directory

To give a comprehensive example of the extension defined in this document, we consider the network in Figure 10. Assume that the ALTO server provides the following information resources:

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

- \* "ane-props": A filtered Unified Property resource that exposes the information for persistent ANEs in the network.
- \* "endpoint-cost-pv": A Multipart Endpoint Cost Service for Path Vector, which exposes the "max-reservable-bandwidth" and the "persistent-entity-id" properties.
- \* "update-pv": An Update Stream service, which provides the incremental update service for the "endpoint-cost-pv" service.
- \* "multicost-pv": A Multipart Endpoint Cost Service with both Multi-Cost and Path Vector.

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

```
{
  "meta": {
    "cost-types": {
      "path-vector": {
        "cost-mode": "array",
        "cost-metric": "ane-path"
      },
      "num-rc": {
        "cost-mode": "numerical",
        "cost-metric": "routingcost"
      }
    }
  },
  "resources": {
    "my-default-networkmap": {
      "uri": "https://alto.example.com/networkmap",
      "media-type": "application/alto-networkmap+json"
    },
    "filtered-cost-map-pv": {
      "uri": "https://alto.example.com/costmap/pv",
      "media-type": "multipart/related;
        type=application/alto-costmap+json",
      "accepts": "application/alto-costmapfilter+json",
      "capabilities": {
        "cost-type-names": [ "path-vector" ],
        "ane-property-names": [ "max-reservable-bandwidth" ]
      },
      "uses": [ "my-default-networkmap" ]
    }
  },
}
```

```
"ane-props": {
  "uri": "https://alto.example.com/ane-props",
  "media-type": "application/alto-propmap+json",
  "accepts": "application/alto-propmapparams+json",
  "capabilities": {
    "mappings": {
      ".ane": [ "cpu" ]
    }
  }
},
"endpoint-cost-pv": {
  "uri": "https://alto.exmaple.com/endpointcost/pv",
  "media-type": "multipart/related;
    type=application/alto-endpointcost+json",
  "accepts": "application/alto-endpointcostparams+json",
  "capabilities": {
    "cost-type-names": [ "path-vector" ],
    "ane-property-names": [
      "max-reservable-bandwidth", "persistent-entity-id"
    ]
  },
  "uses": [ "ane-props" ]
},
"update-pv": {
  "uri": "https://alto.example.com/updates/pv",
  "media-type": "text/event-stream",
  "uses": [ "endpoint-cost-pv" ],
  "accepts": "application/alto-updatestreamparams+json",
  "capabilities": {
    "support-stream-control": true
  }
},
"multicost-pv": {
  "uri": "https://alto.exmaple.com/endpointcost/mcpv",
  "media-type": "multipart/related;
    type=application/alto-endpointcost+json",
  "accepts": "application/alto-endpointcostparams+json",
  "capabilities": {
    "cost-type-names": [ "path-vector", "num-rc" ],
    "max-cost-types": 2,
    "testable-cost-type-names": [ "num-rc" ],
    "ane-property-names": [
      "max-reservable-bandwidth", "persistent-entity-id"
    ]
  },
  "uses": [ "ane-props" ]
}
}
```

```
}
```

### 8.3. Multipart Filtered Cost Map

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

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

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

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

```
POST /costmap/pv HTTP/1.1
Host: alto.example.com
Accept: multipart/related;type=application/alto-costmap+json,
       application/alto-error+json
Content-Length: 153
Content-Type: application/alto-costmapfilter+json
```

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

```
HTTP/1.1 200 OK
Content-Length: 855
Content-Type: multipart/related; boundary=example-1;
             type=application/alto-costmap+json
```

```
--example-1
Content-ID: <costmap@alto.example.com>
Content-Type: application/alto-costmap+json
```



```

{
  "meta": {
    "vtag": {
      "resource-id": "filtered-cost-map-pv.costmap",
      "tag": "d827f484cb66ce6df6b5077cb8562b0a"
    },
    "dependent-vtags": [
      {
        "resource-id": "my-default-networkmap",
        "tag": "c04bc5da49534274a6daeee8ea1dec62"
      }
    ],
    "cost-type": {
      "cost-mode": "array",
      "cost-metric": "ane-path"
    }
  },
  "cost-map": {
    "PID1": {
      "PID3": [ "L1" ],
      "PID4": [ "L1", "L2" ]
    }
  }
}
--example-1
Content-ID: <propmap@alto.example.com>
Content-Type: application/alto-propmap+json

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

```

#### 8.4. Multipart Endpoint Cost Service 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 for two IPv4 source and destination pairs (192.0.2.34 -> 192.0.2.2 and 192.0.2.34 -> 192.0.2.50) and one IPv6 source and destination pair (2001:db8::3:1 -> 2001:db8::4:1).

The response consists of two parts. The first part returns the array of ANENAME for each valid source and destination pair. As one can see in Figure 10, flow 192.0.2.34 -> 192.0.2.2 traverses NET2, L1 and NET1, and flows 192.0.2.34 -> 192.0.2.50 and 2001:db8::3:1 -> 2001:db8::4:1 traverse NET2, L2 and NET3.

The second part returns the requested properties of ANEs. Assume NET1, NET2 and NET3 has sufficient bandwidth and their "max-reservable-bandwidth" values are set to a sufficiently large number (50 Gbps in this case). On the other hand, assume there are no prior reservation on L1 and L2, and their "max-reservable-bandwidth" values are the corresponding link capacity (10 Gbps for L1 and 15 Gbps for L2).

Both NET1 and NET2 have a mobile edge deployed, i.e., MEC1 in NET1 and MEC2 in NET2. Assume the ANENAME for MEC1 and MEC2 are "MEC1" and "MEC2" and their properties can be retrieved from the Property Map "ane-props". Thus, the "persistent-entity-id" property of NET1 and NET3 are "ane-props.ane:MEC1" and "ane-props.ane:MEC2" respectively.

```
POST /endpointcost/pv HTTP/1.1
Host: alto.example.com
Accept: multipart/related;
       type=application/alto-endpointcost+json,
       application/alto-error+json
Content-Length: 362
Content-Type: application/alto-endpointcostparams+json
```

```
{
  "cost-type": {
    "cost-mode": "array",
    "cost-metric": "ane-path"
  },
  "endpoints": {
    "srcs": [
      "ipv4:192.0.2.34",
      "ipv6:2001:db8::3:1"
    ],
    "dsts": [
      "ipv4:192.0.2.2",
      "ipv4:192.0.2.50",
      "ipv6:2001:db8::4:1"
    ]
  },
  "ane-property-names": [
    "max-reservable-bandwidth",
    "persistent-entity-id"
  ]
}
```

```
HTTP/1.1 200 OK
Content-Length: 1432
Content-Type: multipart/related; boundary=example-2;
             type=application/alto-endpointcost+json
```

```
--example-2
Content-ID: <ecs@alto.example.com>
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": {
    "ipv4:192.0.2.34": {
      "ipv4:192.0.2.2": [ "NET3", "L1", "NET1" ],
      "ipv4:192.0.2.50": [ "NET3", "L2", "NET2" ]
    },
    "ipv6:2001:db8::3:1": {
      "ipv6:2001:db8::4:1": [ "NET3", "L2", "NET2" ]
    }
  }
}
--example-2
Content-ID: <propmap@alto.example.com>
Content-Type: application/alto-propmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "endpoint-cost-pv.ecs",
        "tag": "bb6bb72eafe8f9bdc4f335c7ed3b10822a391cef"
      },
      {
        "resource-id": "ane-props",
        "tag": "bf3c8c1819d2421c9a95a9d02af557a3"
      }
    ]
  },
  "property-map": {
    ".ane:NET1": {
      "max-reservable-bandwidth": 50000000000,
      "persistent-entity-id": "ane-props.ane:MEC1"
    },
    ".ane:NET2": {
      "max-reservable-bandwidth": 50000000000,
      "persistent-entity-id": "ane-props.ane:MEC2"
    },
    ".ane:NET3": {
      "max-reservable-bandwidth": 50000000000
    },
    ".ane:L1": {
      "max-reservable-bandwidth": 10000000000
    },
    ".ane:L2": {
      "max-reservable-bandwidth": 15000000000
    }
  }
}
```

```
}
```

Under certain scenarios where the traversal order is not crucial, an ALTO server implementation may choose to not follow strictly the physical traversal order and may even obfuscate the order intentionally to preserve its own privacy or conform to its own policies. For example, an ALTO server may choose to aggregate NET1 and L1 as a new ANE with ANE name "AGGR1", and aggregate NET2 and L2 as a new ANE with ANE name "AGGR2". The "max-reservable-bandwidth" of "AGGR1" takes the value of L1, which is smaller than that of NET1, and the "persistent-entity-id" of "AGGR1" takes the value of NET1. The properties of "AGGR2" are computed in a similar way and the obfuscated response is as shown below. Note that the obfuscation of Path Vector responses is implementation-specific and is out of the scope of this document, and developers may refer to Section 11 for further references.

```
HTTP/1.1 200 OK
Content-Length: 1263
Content-Type: multipart/related; boundary=example-2;
              type=application/alto-endpointcost+json
```

```
--example-2
Content-ID: <ecs@alto.example.com>
Content-Type: application/alto-endpointcost+json
```

```
{
  "meta": {
    "vtags": {
      "resource-id": "endpoint-cost-pv.ecs",
      "tag": "bb975862fbe3422abf4dae386b132c1d"
    },
    "cost-type": {
      "cost-mode": "array",
      "cost-metric": "ane-path"
    }
  },
  "endpoint-cost-map": {
    "ipv4:192.0.2.34": {
      "ipv4:192.0.2.2": [ "NET3", "AGGR1" ],
      "ipv4:192.0.2.50": [ "NET3", "AGGR2" ]
    },
    "ipv6:2001:db8::3:1": {
      "ipv6:2001:db8::4:1": [ "NET3", "AGGR2" ]
    }
  }
}
```

```
--example-2
```

Content-ID: <propmap@alto.example.com>  
Content-Type: application/alto-propmap+json

```
{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "endpoint-cost-pv.ecs",
        "tag": "bb975862fbe3422abf4dae386b132c1d"
      },
      {
        "resource-id": "ane-props",
        "tag": "bf3c8c1819d2421c9a95a9d02af557a3"
      }
    ]
  },
  "property-map": {
    ".ane:AGGR1": {
      "max-reservable-bandwidth": 10000000000,
      "persistent-entity-id": "ane-props.ane:MEC1"
    },
    ".ane:AGGR2": {
      "max-reservable-bandwidth": 15000000000,
      "persistent-entity-id": "ane-props.ane:MEC2"
    },
    ".ane:NET3": {
      "max-reservable-bandwidth": 50000000000
    }
  }
}
```

### 8.5. Incremental Updates

In this example, an ALTO client subscribes to the incremental update for the multipart Endpoint Cost Service 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: 112
```

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

Based on the server-side process defined in [RFC8895], the ALTO server will send the "control-uri" first using Server-Sent Event (SSE), followed by the full response of the multipart message.

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

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

```
event: multipart/related;boundary=example-3;
      type=application/alto-endpointcost+json,ecspvsub1
data: --example-3
data: Content-ID: <ecsmap@alto.example.com>
data: Content-Type: application/alto-endpointcost+json
data:
data: <endpoint-cost-map-entry>
data: --example-3
data: Content-ID: <propmap@alto.example.com>
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, based on Section 6.7.3 of [RFC8895].

```
event: application/merge-patch+json, ecspvsubl.ecsmap@alto.example.com  
data: <Merge patch for endpoint-cost-map-update>
```

```
event: application/merge-patch+json, ecspvsubl.propmap@alto.example.com  
data: <Merge patch for property-map-update>
```

## 8.6. Multi-cost

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

The request asks for two cost types: the first is the Path Vector cost type, and the second is a numerical routing cost. It also queries the Maximum Reservable Bandwidth ANE property and the Persistent Entity property for two IPv4 source and destination pairs (192.0.2.34 -> 192.0.2.2 and 192.0.2.34 -> 192.0.2.50) and one IPv6 source and destination pair (2001:db8::3:1 -> 2001:db8::4:1).

The response consists of two parts. The first part returns a JSONArray that contains two JSONValue for each requested source and destination pair: the first JSONValue is a JSONArray of ANENames, which is the value of the Path Vector cost type, and the second JSONValue is a JSONNumber which is the value of the routing cost. The second part contains a Property Map that maps the ANEs to their requested properties.



```
POST /endpointcost/mcpv HTTP/1.1
Host: alto.example.com
Accept: multipart/related;
       type=application/alto-endpointcost+json,
       application/alto-error+json
Content-Length: 433
Content-Type: application/alto-endpointcostparams+json

{
  "multi-cost-types": [
    { "cost-mode": "array", "cost-metric": "ane-path" },
    { "cost-mode": "numerical", "cost-metric": "routingcost" }
  ],
  "endpoints": {
    "srcs": [
      "ipv4:192.0.2.34",
      "ipv6:2001:db8::3:1"
    ],
    "dsts": [
      "ipv4:192.0.2.2",
      "ipv4:192.0.2.50",
      "ipv6:2001:db8::4:1"
    ]
  },
  "ane-property-names": [
    "max-reservable-bandwidth",
    "persistent-entity-id"
  ]
}

HTTP/1.1 200 OK
Content-Length: 1350
Content-Type: multipart/related; boundary=example-4;
             type=application/alto-endpointcost+json

--example-4
Content-ID: <ecs@alto.example.com>
Content-Type: application/alto-endpointcost+json

{
  "meta": {
    "vtags": {
      "resource-id": "endpoint-cost-pv.ecs",
      "tag": "84a4f9c14f9341f0983e3e5f43a371c8"
    },
    "multi-cost-types": [
      { "cost-mode": "array", "cost-metric": "ane-path" },
      { "cost-mode": "numerical", "cost-metric": "routingcost" }
    ]
  }
}
```

```

    ]
  },
  "endpoint-cost-map": {
    "ipv4:192.0.2.34": {
      "ipv4:192.0.2.2": [[ "NET3", "AGGR1" ], 3],
      "ipv4:192.0.2.50": [[ "NET3", "AGGR2" ], 2]
    },
    "ipv6:2001:db8::3:1": {
      "ipv6:2001:db8::4:1": [[ "NET3", "AGGR2" ], 2]
    }
  }
}
--example-4
Content-ID: <propmap@alto.example.com>
Content-Type: application/alto-propmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "endpoint-cost-pv.ecs",
        "tag": "84a4f9c14f9341f0983e3e5f43a371c8"
      },
      {
        "resource-id": "ane-props",
        "tag": "be157afa031443a187b60bb80a86b233"
      }
    ]
  },
  "property-map": {
    ".ane:AGGR1": {
      "max-reservable-bandwidth": 10000000000,
      "persistent-entity-id": "ane-props.ane:MEC1"
    },
    ".ane:AGGR2": {
      "max-reservable-bandwidth": 15000000000,
      "persistent-entity-id": "ane-props.ane:MEC2"
    },
    ".ane:NET3": {
      "max-reservable-bandwidth": 50000000000
    }
  }
}

```

## 9. Compatibility with Other ALTO Extensions

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

The multipart Filtered Cost Map resource and the multipart Endpoint Cost Service 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 incurring any incompatibility problem.

### 9.2. Compatibility with Multi-Cost Extension

The extension defined in this document is compatible with the multi-cost extension [RFC8189]. Such a resource has a media type of either "multipart/related; type=application/alto-costmap+json" or "multipart/related; type=application/alto-endpointcost+json". Its "cost-constraints" field must either be "false" or not present and the Path Vector cost type must be present in the "cost-type-names" capability field but must not be present in the "testable-cost-type-names" field, as specified in Section 7.2.4 and Section 7.3.4.

### 9.3. Compatibility with Incremental Update

This extension is compatible with the incremental update extension [RFC8895]. ALTO clients and servers MUST follow the specifications given in Sections 5.2 and 6.7.3 of [RFC8895] to support incremental updates for a Path Vector resource.

### 9.4. Compatibility with Cost Calendar

The extension specified in this document is compatible with the Cost Calendar extension [RFC8896]. 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, a property of a single ANE may also have different values in different time intervals. In this case, if such an ANE has different property values in two time intervals, it MUST be treated as two different ANEs, i.e., with different entity identifiers. However, if it has the same property values in two time intervals, it MAY use the same identifier.

This rule allows the Path Vector extension to represent both changes of ANEs and changes of the ANEs' properties in a uniform way. The Path Vector part is calendared in a compatible way, and the Property Map part is not affected by the calendar extension.

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

## 10. General Discussions

### 10.1. Constraint Tests for General Cost Types

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

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

In practice, developing a bespoke language for general-purpose boolean tests can be a complex undertaking, and it is conceivable that there are some existing implementations already (the authors have not done an exhaustive search to determine whether there are such implementations). One avenue to develop such a language may be to explore extending current query languages like XQuery [XQuery] or JSONiq [JSONiq] and integrating these with ALTO.

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

### 10.2. General Multi-Resource Query

Querying multiple ALTO information resources continuously is a general requirement. Enabling such a capability, however, must address general issues like efficiency and consistency. The incremental update extension [RFC8895] supports submitting multiple queries in a single request, and allows flexible control over the queries. However, it does not cover the case introduced in this document where multiple resources are needed for a single request.

This extension gives an example of using a multipart message to encode the responses from two specific ALTO information resources: a Filtered Cost Map or an Endpoint Cost Service, and a Property Map. By packing multiple resources in a single response, the implication is that servers may proactively push related information resources to clients.

Thus, it is worth looking into the direction of extending the SSE mechanism as used in the incremental update extension [RFC8895], or upgrading to HTTP/2 [I-D.ietf-httpbis-http2bis] and HTTP/3 [I-D.ietf-quic-http], which provides the ability to multiplex queries and to allow servers proactively send related information resources.

Defining a general multi-resource query mechanism is out of the scope of this document.

## 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 scrutiny on three security considerations discussed in the base protocol: confidentiality of ALTO information (Section 15.3 of [RFC7285]), potential undesirable guidance from authenticated ALTO information (Section 15.2 of [RFC7285]), and availability of ALTO service (Section 15.5 of [RFC7285]).

For confidentiality of ALTO information, a network operator should be aware that this extension may introduce a new risk: the Path Vector information, when used together with sensitive ANE properties such as capacities of bottleneck links, 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 attacker may identify the bottleneck link and start a distributed denial-of-service (DDoS) attack involving minimal flows to conduct the in-network congestion. Given the potential risk of leaking sensitive information, the Path Vector extension is mainly applicable in scenarios where 1) the ANE structures and ANE properties do not impose security risks to the ALTO service provider, e.g., not carrying sensitive information, or 2) the ALTO server and client have established a reliable trust relationship, for example, operated in the same administrative domain, or managed by business partners with legal contracts.

Three risk types are identified in Section 15.3.1 of [RFC7285]: (1) Excess disclosure of the ALTO service provider's data to an unauthorized ALTO client; (2) Disclosure of the ALTO service provider's data (e.g., network topology information or endpoint addresses) to an unauthorized third party; and (3) Excess retrieval of the ALTO service provider's data by collaborating ALTO clients. To mitigate these risks, an ALTO server MUST follow the guidelines in Section 15.3.2 of [RFC7285]. Furthermore, an ALTO server MUST follow the following additional protections strategies for risk types (1) and (3).

For risk type (1), an ALTO server MUST use the authentication methods specified in Section 15.3.2 of [RFC7285] to authenticate the identify of an ALTO client, and apply access control techniques to restrict unprivileged ALTO clients from retrieving sensitive Path Vector information. For settings where the ALTO server and client are not in the same trust domain, the ALTO server should reach agreements with the ALTO client on protecting the confidentiality before granting the access to Path Vector service with sensitive information. Such agreements may include legal contracts or Digital Right Management (DRM) techniques. Otherwise, the ALTO server MUST NOT offer the Path Vector service carrying sensitive information to the clients unless the potential risks are fully assessed and mitigated.

For risk type (3), an ALTO service provider must be aware that persistent ANEs may be used as "landmarks" in collaborative inferences. Thus, they should only be used when exposing public service access points (e.g., API gateways, CDNi) and/or when the granularity is coarse-grained (e.g., when an ANE represents an AS, a data center or a WAN). Otherwise, an ALTO server MUST use dynamic mappings from ephemeral ANE names to underlying physical entities. Specifically, for the same physical entity, an ALTO server SHOULD assign a different ephemeral ANE name when the entity appears in the responses to different clients or even for different request from the same client. A RECOMMENDED assignment strategy is to generate ANE names from random numbers.

Further, to protect the network topology from graph reconstruction (e.g., through isomorphic graph identification [BONDY]), the ALTO server SHOULD consider protection mechanisms to reduce information exposure or obfuscate the real information. When doing so, the ALTO server must be aware that information reduction/obfuscation may lead to potential Undesirable Guidance from Authenticated ALTO Information risk (Section 15.2 of [RFC7285]).

Thus, implementations of ALTO servers involving reduction or obfuscation of the Path Vector information SHOULD consider reduction/obfuscation mechanisms that can preserve the integrity of ALTO information, for example, by using minimal feasible region compression algorithms [NOVA] or obfuscation protocols [RESA][MERCATOR]. However, these obfuscation methods are experimental and their practical applicability of these methods to the generic capability provided by this extension is not fully assessed. The ALTO server MUST carefully verify that the deployment scenario satisfies the security assumptions of these methods before applying them to protect Path Vector services with sensitive network information.

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 consume intolerable amounts of the server-side computation and storage, which can break the ALTO server. For example, if an ALTO server implementation dynamically computes the Path Vectors for each request, the service providing Path Vectors may become an entry point for denial-of-service attacks on the availability of an ALTO server.

To mitigate this risk, an ALTO server may consider using optimizations such as precomputation-and-projection mechanisms [MERCATOR] to reduce the overhead for processing each query. Also, an ALTO server may also protect itself from malicious clients by monitoring the behaviors of clients and stopping serving clients with suspicious behaviors (e.g., sending requests at a high frequency).

The ALTO service providers must be aware that providing incremental updates of the "max-reservable-bandwidth" may provide information about other consumers of the network. For example, a change of the value may indicate one or more reservations has been made or changed. To mitigate this risk, an ALTO server can batch the updates and/or add a random delay before publishing the updates.

## 12. IANA Considerations

### 12.1. ALTO Cost Metric Registry

This document registers a new entry to the ALTO Cost Metric Registry, as instructed by Section 14.2 of [RFC7285]. The new entry is as shown below in Table 1.

Identifier	Intended Semantics	Security Considerations
ane-path	See Section 6.5.1	See Section 11

Table 1: ALTO Cost Metric Registry

## 12.2. ALTO Cost Mode Registry

This document registers a new entry to the ALTO Cost Mode Registry, as instructed by Section 4 of [I-D.bw-alto-cost-mode]. The new entry is as shown below in Table 2.

Identifier	Intended Semantics
array	See Section 6.5.2

Table 2: ALTO Cost Mode Registry

## 12.3. ALTO Entity Domain Type Registry

This document registers a new entry to the ALTO Domain Entity Type Registry, as instructed by Section 12.2 of [I-D.ietf-alto-unified-props-new]. The new entry is as shown below in Table 3.

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

Table 3: ALTO Entity Domain Type Registry

Identifier: See Section 6.2.1.

Entity Identifier Encoding: See Section 6.2.2.

Hierarchy: None

Inheritance: None



Media Type of Defining Resource: See Section 6.2.4.

Mapping to ALTO Address Type: This entity type does not map to ALTO address type.

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

#### 12.4. ALTO Entity Property Type Registry

Two initial entries "max-reservable-bandwidth" and "persistent-entity-id" are registered to the ALTO Domain "ane" in the "ALTO Entity Property Type Registry", as instructed by Section 12.3 of [I-D.ietf-alto-unified-props-new]. The two new entries are shown below in Table 4 and their details can be found in Section 12.4.1 and Section 12.4.2.

Identifier	Intended Semantics	Media Type of Defining Resource
max-reservable-bandwidth	See Section 6.4.1	application/alto-propmap+json
persistent-entity-id	See Section 6.4.2	application/alto-propmap+json

Table 4: Initial Entries for ane Domain in the ALTO Entity Property Types Registry

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

Identifier: "max-reservable-bandwidth"

Intended Semantics: See Section 6.4.1.

Media Type of Defining Resource: application/alto-propmap+json

Security Considerations: This property is essential for applications such as large-scale data transfers or overlay network interconnection to make better choice of bandwidth reservation. It may reveal the bandwidth usage of the underlying network and can potentially be leveraged to reduce the cost of conducting

denial-of-service attacks. Thus, the ALTO server MUST consider protection mechanisms including only providing the information to authorized clients, and information reduction and obfuscation as introduced in Section 11.

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

Identifier: "persistent-entity-id"

Intended Semantics: See Section 6.4.2.

Media Type of Defining Resource: application/alto-propmap+json

Security Considerations: This property is useful when an ALTO server wants to selectively expose certain service points whose detailed properties can be further queried by applications. The entity IDs may consider sensitive information about the underlying network, and an ALTO server should follow the security considerations in Section 11 of [I-D.ietf-alto-unified-props-new].

### 13. References

#### 13.1. Normative References

[I-D.bw-alto-cost-mode]

Boucadair, M. and Q. Wu, "A Cost Mode Registry for the Application-Layer Traffic Optimization (ALTO) Protocol", Work in Progress, Internet-Draft, draft-bw-alto-cost-mode-01, 4 March 2022, <<https://datatracker.ietf.org/doc/html/draft-bw-alto-cost-mode-01>>.

[I-D.ietf-alto-unified-props-new]

Roome, W., Randriamasy, S., Yang, Y. R., Zhang, J. J., and K. Gao, "An ALTO Extension: Entity Property Maps", Work in Progress, Internet-Draft, draft-ietf-alto-unified-props-new-24, 28 February 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-alto-unified-props-new-24>>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <<https://www.rfc-editor.org/rfc/rfc2046>>.

[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/rfc/rfc2119>>.

- [RFC2387] Levinson, E., "The MIME Multipart/Related Content-type", RFC 2387, DOI 10.17487/RFC2387, August 1998, <<https://www.rfc-editor.org/rfc/rfc2387>>.
- [RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <<https://www.rfc-editor.org/rfc/rfc5322>>.
- [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/rfc/rfc7285>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.
- [RFC8189] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost Application-Layer Traffic Optimization (ALTO)", RFC 8189, DOI 10.17487/RFC8189, October 2017, <<https://www.rfc-editor.org/rfc/rfc8189>>.
- [RFC8895] Roome, W. and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Incremental Updates Using Server-Sent Events (SSE)", RFC 8895, DOI 10.17487/RFC8895, November 2020, <<https://www.rfc-editor.org/rfc/rfc8895>>.
- [RFC8896] Randriamasy, S., Yang, R., Wu, Q., Deng, L., and N. Schwan, "Application-Layer Traffic Optimization (ALTO) Cost Calendar", RFC 8896, DOI 10.17487/RFC8896, November 2020, <<https://www.rfc-editor.org/rfc/rfc8896>>.

### 13.2. Informative References

- [BONDY] Bondy, J.A. and R.L. Hemminger, "Graph reconstruction survey", Journal of Graph Theory, Volume 1, Issue 3, pp 227-268 , 1977, <<https://doi.org/10.1002/jgt.3190010306>>.
- [BOXOPT] 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, <<https://doi.org/10.1609/aaai.v33i01.33011674>>.

- [CLARINET] Viswanathan, R., Ananthanarayanan, G., and A. Akella, "CLARINET: WAN-Aware Optimization for Analytics Queries", In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), USENIX Association, Savannah, GA, 435-450 , 2016, <<https://dl.acm.org/doi/abs/10.5555/3026877.3026911>>.
- [G2] Ros-Giralt, J., Bohara, A., Yellamraju, S., Langston, M.H., Lethin, R., Jiang, Y., Tassiulas, L., Li, J., Tan, Y., and M. Veeraraghavan, "On the Bottleneck Structure of Congestion-Controlled Networks", Proceedings of the ACM on Measurement and Analysis of Computing Systems, Volume 3, Issue 3, pp 1-31 , 2019, <<https://dl.acm.org/doi/10.1145/3366707>>.
- [HUG] Chowdhury, M., Liu, Z., Ghodsi, A., and I. Stoica, "HUG: Multi-Resource Fairness for Correlated and Elastic Demands", 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16) (Santa Clara, CA, 2016), 407-424. , 2016, <<https://dl.acm.org/doi/10.5555/2930611.2930638>>.
- [I-D.ietf-alto-performance-metrics]  
Wu, Q., Yang, Y. R., Lee, Y., Dhody, D., Randriamasy, S., and L. M. C. Murillo, "ALTO Performance Cost Metrics", Work in Progress, Internet-Draft, draft-ietf-alto-performance-metrics-26, 2 March 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-alto-performance-metrics-26>>.
- [I-D.ietf-httpbis-http2bis]  
Thomson, M. and C. Benfield, "HTTP/2", Work in Progress, Internet-Draft, draft-ietf-httpbis-http2bis-07, 24 January 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-http2bis-07>>.
- [I-D.ietf-quic-http]  
Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-http-34, 2 February 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-34>>.
- [JSONiq] "The JSON Query language", 2020, <<https://www.jsoniq.org/>>.

- [MERCATOR] Xiang, Q., Zhang, J., Wang, X., Liu, Y., Guok, C., Le, F., MacAuley, J., Newman, H., and Y.R. Yang, "Toward Fine-Grained, Privacy-Preserving, Efficient Multi-Domain Network Resource Discovery", IEEE/ACM IEEE Journal on Selected Areas of Communication 37(8): 1924-1940, 2019, <<https://doi.org/10.1109/JSAC.2019.2927073>>.
- [MOWIE] Zhang, Y., Li, G., Xiong, C., Lei, Y., Huang, W., Han, Y., Walid, A., Yang, Y.R., and Z. Zhang, "MoWIE: Toward Systematic, Adaptive Network Information Exposure as an Enabling Technique for Cloud-Based Applications over 5G and Beyond", In Proceedings of the Workshop on Network Application Integration/CoDesign, ACM, Virtual Event USA, 20-27. , 2020, <<https://doi.org/10.1145/3405672.3409489>>.
- [NOVA] 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, <<https://doi.org/10.1109/IWQoS.2017.7969117>>.
- [RESA] Xiang, Q., Zhang, J., Wang, X., Liu, Y., Guok, C., Le, F., MacAuley, J., Newman, H., and Y.R. Yang, "Fine-grained, multi-domain network resource abstraction as a fundamental primitive to enable high-performance, collaborative data sciences", Proceedings of the Super Computing 2018, 5:1-5:13 , 2019, <<https://doi.org/10.1109/SC.2018.000008>>.
- [RFC2216] Shenker, S. and J. Wroclawski, "Network Element Service Specification Template", RFC 2216, DOI 10.17487/RFC2216, September 1997, <<https://www.rfc-editor.org/rfc/rfc2216>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/rfc/rfc4271>>.
- [SENSE] "Software Defined Networking (SDN) for End-to-End Networked Science at the Exascale", 2019, <<https://www.es.net/network-r-and-d/sense/>>.
- [SEREDGE] Contreras, L., Baliosian, J., Martnez-Julia, P., and J. Serrat, "Computing at the Edge: But, what Edge?", In proceedings of the NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium. pp. 1-9. , 2020, <<https://doi.org/10.1109/NOMS47738.2020.9110342>>.

- [SWAN] Hong, C., Kandula, S., Mahajan, R., Zhang, M., Gill, V., Nanduri, M., and R. Wattenhofer, "Achieving High Utilization with Software-driven WAN", In Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13), ACM, New York, NY, USA, 15-26. , 2013, <<http://doi.acm.org/10.1145/2486001.2486012>>.
- [UNICORN] Xiang, Q., Chen, S., Gao, K., Newman, H., Taylor, I., Zhang, J., and Y.R. Yang, "Unicorn: Unified Resource Orchestration for Multi-Domain, Geo-Distributed Data Analytics", 2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI) (Aug. 2017), 1-6. , 2017, <<https://doi.org/10.1016/j.future.2018.09.048>>.
- [XQuery] "XQuery 3.1: An XML Query Language", 2017, <<https://www.w3.org/TR/xquery-31/>>.

## Appendix A. 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, Dawn Chen, Wendy Roome, and Michael Scharf for their contributions to earlier drafts.

The authors would also like to thank Tim Chown, Luis Contreras, Roman Danyliw, Benjamin Kaduk, Erik Kline, Suresh Krishnan, Murray Kucherawy, Warren Kumari, Danny Lachos, Francesca Palombini, Eric Vyncke, Samuel Weiler, and Qiao Xiang whose feedback and suggestions are invaluable to improve the practicability and conciseness of this document, and Mohamed Boucadair, Martin Duke, Vijay Gurbani, Jan Seedorf, and Qin Wu who provide great support and guidance.

## Appendix B. Revision Logs (To be removed before publication)

### B.1. Changes since -20

Reivision -21

- \* changes the normative requirement on protecting confidentiality of PV information with softer language

## B.2. Changes since -19

Revision -20

- \* changes the IANA registry information
- \* adopts the comments from IESG reviews

## B.3. Changes since -18

Revision -19

- \* adds detailed examples for use cases
- \* clarify terms with ambiguous meanings

## B.4. Changes since -17

Revision -18

- \* changes the specification for content-id to conform to [RFC2387] and [RFC5322]
- \* adds IPv6 examples

## B.5. Changes since -16

Revision -17

- \* adds items for media type of defining resources in IANA considerations

## B.6. Changes since -15

Revision -16

- \* resolves the compatibility with the Multi-Cost extension (RFC 8189)
- \* adds media types of defining resources for ANE property types (for IANA registration)

## B.7. Changes since -14

Revision -15

- \* fixes the IDNits warnings,

- \* fixes grammar issues,
- \* addresses the comments in the AD review.

#### B.8. Changes since -13

##### Revision -14

- \* addresses the comments in the chair review,
- \* fixes most issues raised by IDNits.

#### B.9. Changes since -12

##### Revision -13

- \* changes the abstract based on the chairs' reviews
- \* integrates Richard's responds to WGLC reviews

#### B.10. Changes since -11

##### Revision -12

- \* clarifies the definition of ANEs in a similar way as how Network Elements is defined in [RFC2216]
- \* restructures several paragraphs that are not clear (Sec 3, Path Vector bullet, Sec 4.2, Sec 5.1.3, Sec 6.2.4, Sec 6.4.2, Sec 9.3)
- \* uses "ALTO Entity Domain Type Registry"

#### B.11. Changes since -10

##### Revision -11

- \* replaces "part" with "components" in the abstract;
- \* identifies additional requirements (AR) derived from the flow scheduling example, and introduces how the extension addresses the additional requirements
- \* fixes the inconsistent use of "start" parameter in multipart responses;
- \* specifies explicitly how to handle "cost-constraints";



- \* uses the latest IANA registration mechanism defined in [I-D.ietf-alto-unified-props-new];
- \* renames "persistent-entities" to "persistent-entity-id";
- \* makes "application/alto-propmap+json" as the media type of defining resources for the "ane" domain;
- \* updates the examples;
- \* adds the discussion on ephemeral and persistent ANEs.

#### B.12. Changes since -09

Revision -10

- \* revises the introduction which
  - extends the scope where the PV extension can be applied beyond the "path correlation" information
- \* brings back the capacity region use case to better illustrate the problem
- \* revises the overview to explain and defend the concepts and decision choices
- \* fixes inconsistent terms, typos

#### B.13. 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

#### B.14. 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  
Sichuan University  
No.24 South Section 1, Yihuan Road  
Chengdu  
610000  
China  
Email: kaigao@scu.edu.cn

Young Lee  
Samsung  
South Korea  
Email: younglee.tx@gmail.com

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  
Tongji University  
4800 Caoan Road  
Shanghai  
201804  
China  
Email: [jingxuan.n.zhang@gmail.com](mailto:jingxuan.n.zhang@gmail.com)

ALTO  
Internet-Draft  
Intended status: Standards Track  
Expires: February 21, 2020

S. Kiesel  
University of Stuttgart  
M. Stiemerling  
H-DA  
August 20, 2019

Application Layer Traffic Optimization (ALTO) Cross-Domain Server  
Discovery  
draft-ietf-alto-xdom-disc-06

Abstract

The goal of Application-Layer Traffic Optimization (ALTO) is to provide guidance to applications that have to select one or several hosts from a set of candidates capable of providing a desired resource. ALTO is realized by a client-server protocol. Before an ALTO client can ask for guidance it needs to discover one or more ALTO servers that can provide suitable guidance.

In some deployment scenarios, in particular if the information about the network topology is partitioned and distributed over several ALTO servers, it may be needed to discover an ALTO server outside of the own network domain, in order to get appropriate guidance. This document details applicable scenarios, itemizes requirements, and specifies a procedure for ALTO cross-domain server discovery.

Technically, the procedure specified in this document takes one IP address or prefix and a U-NAPTR Service Parameter (typically, "ALTO:https") as parameters. It performs DNS lookups (for NAPTR resource records in the in-addr.arpa. or ip6.arpa. tree) and returns one or more URI(s) of information resources related to that IP address or prefix.

## Terminology and Requirements Language

This document makes use of the ALTO terminology defined in RFC 5693 [RFC5693].

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

## Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 21, 2020.

## Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	4
2. ALTO Cross-Domain Server Discovery Procedure: Overview . . . . .	5
3. ALTO Cross-Domain Server Discovery Procedure: Specification . . . . .	6
3.1. Interface . . . . .	6
3.2. Step 1: Prepare Domain Name for Reverse DNS Lookup . . . . .	7
3.3. Step 2: Prepare Shortened Domain Names . . . . .	8
3.4. Step 3: Perform DNS U-NAPTR lookups . . . . .	9
3.5. Error Handling . . . . .	10
4. Using the ALTO Protocol with Cross-Domain Server Discovery . . . . .	11
4.1. Network and Cost Map Service . . . . .	11
4.2. Map-Filtering Service . . . . .	12
4.3. Endpoint Property Service . . . . .	12
4.4. Endpoint Cost Service . . . . .	12
4.5. Summary and Further Extensions . . . . .	14
5. Implementation, Deployment, and Operational Considerations . . . . .	15
5.1. Considerations for ALTO Clients . . . . .	15
5.2. Considerations for Network Operators . . . . .	17
6. Security Considerations . . . . .	19
6.1. Integrity of the ALTO Server's URI . . . . .	19
6.2. Availability of the ALTO Server Discovery Procedure . . . . .	20
6.3. Confidentiality of the ALTO Server's URI . . . . .	21
6.4. Privacy for ALTO Clients . . . . .	21
7. IANA Considerations . . . . .	23
8. References . . . . .	24
8.1. Normative References . . . . .	24
8.2. Informative References . . . . .	24
Appendix A. Solution Approaches for Partitioned ALTO Knowledge . . . . .	27
A.1. Classification of Solution Approaches . . . . .	27
A.2. Discussion of Solution Approaches . . . . .	28
A.3. The Need for Cross-Domain ALTO Server Discovery . . . . .	28
A.4. Our Solution Approach . . . . .	29
A.5. Relation to the ALTO Requirements . . . . .	29
Appendix B. Requirements for Cross-Domain Server Discovery . . . . .	30
B.1. Discovery Client Application Programming Interface . . . . .	30
B.2. Data Storage and Authority Requirements . . . . .	30
B.3. Cross-Domain Operations Requirements . . . . .	30
B.4. Protocol Requirements . . . . .	31
B.5. Further Requirements . . . . .	31
Appendix C. ALTO and Tracker-based Peer-to-Peer Applications . . . . .	32
C.1. A generic Tracker-based Peer-to-Peer Application . . . . .	32
C.2. Architectural Options for Placing the ALTO Client . . . . .	33
C.3. Evaluation . . . . .	36
C.4. Example . . . . .	38
Appendix D. Contributors List and Acknowledgments . . . . .	43
Authors' Addresses . . . . .	44

## 1. Introduction

The goal of Application-Layer Traffic Optimization (ALTO) is to provide guidance to applications that have to select one or several hosts from a set of candidates capable of providing a desired resource [RFC5693]. ALTO is realized by an HTTP-based client-server protocol [RFC7285], which can be used in various scenarios [RFC7971].

The ALTO base protocol document [RFC7285] specifies the communication between an ALTO client and one ALTO server. In principle, the client may send any ALTO query. For example, it might ask for the routing cost between any two IP addresses, or it might request network and cost maps for the whole network, which might be the worldwide Internet. It is assumed that the server can answer any query, possibly with some kind of default value if no exact data is known.

No special provisions were made for deployment scenarios with multiple ALTO servers, with some servers having more accurate information about some parts of the network topology while others having better information about other parts of the network ("partitioned knowledge"). Various ALTO use cases have been studied in the context of such scenarios. In some cases, one cannot assume that a topologically nearby ALTO server (e.g., a server discovered with the procedure specified in [RFC7286]) will always provide useful information to the client. One such scenario is detailed in Appendix C. Several solution approaches, such as redirecting a client to a server that has more accurate information or forwarding the request to it on behalf of the client, have been proposed and analyzed (see Appendix A), but none has been specified so far.

Section 3 of this document specifies the "ALTO Cross-Domain Server Discovery Procedure" for client-side usage in these scenarios. An ALTO client that wants to send an ALTO query related to a specific IP address or prefix X, may call this procedure with X as a parameter. It will use Domain Name System (DNS) lookups to find of one or more ALTO servers that can provide a competent answer. The above wording "related to" was intentionally kept somewhat unspecific, as the exact semantics depends on the ALTO service to be used; see Section 4.

Those who are in control of the "reverse DNS" for a given IP address or prefix (i.e., the corresponding subdomain of in-addr.arpa. or ip6.arpa.) – typically an Internet Service Provider (ISP), a corporate IT department, or a university's computing center – may add resource records to the DNS that point to one or more relevant ALTO server(s). In many cases, it may be an ALTO server run by that ISP or IT department, as they naturally have good insight into routing costs from and to their networks. However, they may also refer to an ALTO server provided by someone else, e.g., their upstream ISP.

## 2. ALTO Cross-Domain Server Discovery Procedure: Overview

This section gives a non-normative overview on the ALTO Cross-Domain Server Discovery Procedure. The detailed specification will follow in the next section.

This procedure was inspired by the "Location Information Server (LIS) Discovery Using IP Addresses and Reverse DNS" [RFC7216] and re-uses parts of the basic ALTO Server Discovery Procedure [RFC7286].

The basic idea is to use the Domain Name System (DNS), more specifically the "in-addr.arpa." or "ip6.arpa." trees, which are mostly used for "reverse mapping" of IP addresses to host names by means of PTR resource records. There, URI-enabled Naming Authority Pointer (U-NAPTR) resource records [RFC4848], which allow the mapping of domain names to Uniform Resource Identifiers (URIs), are installed as needed. Thereby, it is possible to store a mapping from an IP address or prefix to one or more ALTO server URIs in the DNS.

The ALTO Cross-Domain Server Discovery Procedure is called with one IP address or prefix and a U-NAPTR Service Parameter [RFC4848] as parameters.

The service parameter is usually set to "ALTO:https". However, other parameter values may be used in some scenarios, e.g., "ALTO:http" to search for a server that supports unencrypted transmission for debugging purposes, or other application protocol or service tags if applicable.

The procedure performs DNS lookups and returns one or more URI(s) of information resources related to said IP address or prefix, usually the URI(s) of one or more ALTO Information Resource Directory (IRD, see Section 9 of [RFC7285]). The U-NAPTR records also provide preference values, which should be considered if more than one URI is returned.

The discovery procedure sequentially tries two different lookup strategies: First, an ALTO-specific U-NAPTR record is searched in the "reverse tree", i.e., in subdomains of in-addr.arpa. or ip6.arpa. corresponding to the given IP address or prefix. If this lookup does not yield a usable result, the procedure tries further lookups with truncated domain names, which correspond to shorter prefix lengths. The goal is to allow deployment scenarios that require fine-grained discovery on a per-IP basis, as well as large-scale scenarios where discovery is to be enabled for a large number of IP addresses with a small number of additional DNS resource records.



### 3. ALTO Cross-Domain Server Discovery Procedure: Specification

#### 3.1. Interface

The procedure specified in this document takes two parameters, X and SP, where X is an IP address or prefix and SP is a U-NAPTR Service Parameter.

The parameter X may be an IPv4 or an IPv6 address or prefix in CIDR notation (see [RFC4632] for the IPv4 CIDR notation and [RFC4291] for IPv6). Consequently, the address type AT is either "IPv4" or "IPv6". In both cases, X consists of an IP address A and a prefix length L. From the definition of IPv4 and IPv6 it follows that syntactically valid values for L are  $0 \leq L \leq 32$  when AT=IPv4 and  $0 \leq L \leq 128$  when AT=IPv6. However, not all syntactically valid values of L are actually supported by this procedure – Step 1 (see below) will check for unsupported values and report an error if necessary.

For example, for X=198.51.100.0/24, we get AT=IPv4, A=198.51.100.0 and L=24. Similarly, for X=2001:0DB8::20/128, we get AT=IPv6, A=2001:0DB8::20 and L=128.

In the intended usage scenario, the procedure is normally always called with the parameter SP set to "ALTO:https". However, for general applicability and in order to support future extensions, the procedure MUST support being called with any valid U-NAPTR Service Parameter (see Section 4.5. of [RFC4848] for the syntax of U-NAPTR Service Parameters and Section 5. of the same document for information about the IANA registries).

The procedure performs DNS lookups and returns one or more URI(s) of information resources related to that IP address or prefix, usually the URI(s) of one or more ALTO Information Resource Directory (IRD, see Section 9 of [RFC7285]). For each URI, it also returns order and preference values (see Section 4.1 of [RFC3403]), which should be considered if more than one URI is returned.

During execution of this procedure, various error conditions may occur and have to be reported to the caller; see Section 3.5.

For the remainder of the document, we use the following notation for calling the ALTO Cross-Domain Server Discovery Procedure:

```
IRD_URIS_X = XDMDISC(X, "ALTO:https")
```





### 3.4. Step 3: Perform DNS U-NAPTR lookups

The address type and the prefix length of X are matched against the first and the second column of the following table, respectively:

1: Address Type AT	2: Prefix Length L	3: MUST do 1st lookup	4: SHOULD do further lookups in that order
IPv4	32	R32	R24, R16, R8
IPv4	24 .. 31	R24	R16, R8
IPv4	16 .. 23	R16	R8
IPv4	8 .. 15	R8	(none)
IPv4	0 .. 7	(none, abort: unsupported prefix length)	
IPv6	128	R128	R64, R56, R48, R40, R32
IPv6	64 (..127)	R64	R56, R48, R40, R32
IPv6	56 .. 63	R56	R48, R40, R32
IPv6	48 .. 55	R48	R40, R32
IPv6	40 .. 47	R40	R32
IPv6	32 .. 39	R32	(none)
IPv6	0 .. 31	(none, abort: unsupported prefix length)	

Then, the domain name given in the 3rd column and the U-NAPTR Service Parameter SP the procedure was called with (usually "ALTO:https") MUST be used for an U-NAPTR [RFC4848] lookup, in order to obtain one or more URIs (indicating protocol, host, and possibly path elements) for the ALTO server's Information Resource Directory (IRD). If such URI(s) can be found, the ALTO Cross-Domain Server Discovery Procedure returns that information to the caller and terminates successfully.

For example, the following two U-NAPTR resource records can be used for mapping "100.51.198.IN-ADDR.ARPA." (i.e., R24 from the example in the previous step) to the HTTPS URIs "https://alto1.example.net/ird" and "https://alto2.example.net/ird", with the former being preferred.

```
100.51.198.IN-ADDR.ARPA. IN NAPTR 100 10 "u" "ALTO:https"
"!.*!https://alto1.example.net/ird!" ""
```

```
100.51.198.IN-ADDR.ARPA. IN NAPTR 100 20 "u" "ALTO:https"
"!.*!https://alto2.example.net/ird!" ""
```

If no matching U-NAPTR records can be found, the procedure SHOULD try further lookups, using the domain names from the fourth column in the indicated order, until one lookup succeeds. If no IRD URI could be found after looking up all domain names from the 3rd and 4th column, the procedure terminates unsuccessfully, returning an empty URI list.

### 3.5. Error Handling

The ALTO Cross-Domain Server Discovery Procedure may fail for several reasons.

If the procedure is called with syntactically invalid parameters or unsupported parameter values (in particular the prefix length *L*, see Section 3.2), the procedure aborts, no URI list will be returned and the error has to be reported to the caller.

The procedure performs one or more DNS lookups in a well-defined order (corresponding to descending prefix lengths, see Section 3.4), until one produces a usable result. Each of these DNS lookups might not produce a usable result, either due to a normal condition (e.g., domain name exists, but no ALTO-specific NAPTR resource records are associated with it), a permanent error (e.g., non-existent domain name), or due to a temporary error (e.g., timeout). In all three cases, and as long as there are further domain names that can be looked up, the procedure SHOULD immediately try to lookup the next domain name (from column 4 in the table given in Section 3.4). Only after all domain names have been tried at least once, the procedure MAY retry those domain names that had caused temporary lookup errors.

Generally speaking, ALTO provides advisory information for the optimization of applications (e.g., peer-to-peer applications, overlay networks, etc.), but applications should not rely on the availability of such information for their basic functionality (see Section 8.3.4.3 of RFC 7285 [RFC7285]). Consequently, the speedy detection of an ALTO server, even though it may give less accurate answers than other servers, or the quick realization that there is no suitable ALTO server, is in general more preferable than causing long delays by retrying failed queries. Nevertheless, the ALTO Cross-Domain Server Discovery Procedure SHOULD inform its caller, if DNS queries have failed due to temporary errors and that retrying the discovery at a later point in time might give more accurate results.

#### 4. Using the ALTO Protocol with Cross-Domain Server Discovery

Based on a modular design principle, ALTO provides several ALTO services, each consisting of a set of information resources that can be accessed using the ALTO protocol. The information resources that are available at a specific ALTO server are listed in its Information Resource Directory (IRD, see Section 9 of [RFC7285]). The ALTO protocol specification defines the following ALTO services and their corresponding information resources:

- o Network and Cost Map Service, see Section 11.2 of [RFC7285]
- o Map-Filtering Service, see Section 11.3 of [RFC7285]
- o Endpoint Property Service, see Section 11.4 of [RFC7285]
- o Endpoint Cost Service, see Section 11.5 of [RFC7285]

The ALTO Cross-Domain Server Discovery Procedure is most useful in conjunction with the Endpoint Property Service and the Endpoint Cost Service. However, for the sake of completeness, possible interaction with all four services is discussed below. Extension documents may specify further information resources; however, these are out of scope of this document.

##### 4.1. Network and Cost Map Service

An ALTO client may invoke the ALTO Cross-Domain Server Discovery Procedure (as specified in Section 3) for an IP address or prefix "X" and get a list of one or more IRD URI(s), including order and preference values: `IRD_URI_X = XDOMDISC(X, "ALTO:https")`. The IRD(s) referenced by these URI(s) will always contain a network and a cost map, as these are mandatory information resources (see Section 11.2 of [RFC7285]). However, the cost matrix may be very sparse. If, according to the network map, `PID_X` is the PID that contains the IP address or prefix X, and `PID_1`, `PID_2`, `PID_3`, ... are other PIDs, the cost map may look like this:

From \ To	PID_1	PID_2	PID_X	PID_3
PID_1			92	
PID_2			6	
PID_X	46	3	1	19
PID_3			38	

In this example, all cells outside column "X" and row "X" are unspecified. A cost map with this structure contains the same information as what could be retrieved using the Endpoint Cost

Service, cases 1 and 2 in Section 4.4. Accessing cells that are neither in column "X" nor row "X" may not yield useful results.

Trying to assemble a more densely populated cost map from several cost maps with this very sparse structure may be a non-trivial task, as different ALTO servers may use different PID definitions (i.e., network maps) and incompatible scales for the costs, in particular for the "routingcost" metric.

#### 4.2. Map-Filtering Service

An ALTO client may invoke the ALTO Cross-Domain Server Discovery Procedure (as specified in Section 3) for an IP address or prefix "X" and get a list of one or more IRD URI(s), including order and preference values: `IRD_URI_X = XDOMDISC(X,"ALTO:https")`. These IRD(s) may provide the optional Map-Filtering Service (see Section 11.3 of [RFC7285]). This service returns a subset of the full map, as specified by the client. As discussed in Section 4.1, a cost map may be very sparse in the envisioned deployment scenario. Therefore, depending on the filtering criteria provided by the client, this service may return results similar to the Endpoint Cost Service, or it may not return any useful result.

#### 4.3. Endpoint Property Service

If an ALTO client wants to query an Endpoint Property Service (see Section 11.4 of RFC 7285 [RFC7285]) about an endpoint with IP address "X" or a group of endpoints within IP prefix "X", respectively, it has to invoke the ALTO Cross-Domain Server Discovery Procedure (as specified in Section 3): `IRD_URI_X = XDOMDISC(X,"ALTO:https")`. The result `IRD_URI_X` is a list of one or more URIs of Information Resource Directories (IRD, see Section 9 of [RFC7285]). Considering the order and preference values, the client has to check these IRDs for a suitable Endpoint Property Service and query it.

If the ALTO client wants to do a similar Endpoint Property query for a different IP address or prefix "Y", the whole procedure has to be repeated, as `IRD_URI_Y = XDOMDISC(Y,"ALTO:https")` may yield a different list of IRD URIs. Of course, the results of individual DNS queries may be cached as indicated by their respective time-to-live (TTL) values.

#### 4.4. Endpoint Cost Service

The optional ALTO Endpoint Cost Service (ECS, see Section 11.5 of RFC 7285 [RFC7285]) provides information about costs between individual endpoints and it also supports ranking. The ECS allows that endpoints may be denoted by IP addresses or prefixes. The ECS is

called with a list of one or more source IP addresses or prefixes, which we will call (S1, S2, S3, ...), and a list of one or more destination IP addresses or prefixes, called (D1, D2, D3, ...).

This specification distinguishes several cases, regarding the number of elements in the list of source and destination addresses, respectively:

1. Exactly one source address S1 and more than one destination addresses D1, D2, D3, ... In this case, the ALTO client has to invoke the ALTO Cross-Domain Server Discovery Procedure (as specified in Section 3) with that single source address as a parameter: `IRD_URIS_S1 = XDOMDISC(S1,"ALTO:https")`. The result `IRD_URIS_S1` is a list of one or more URIs of Information Resource Directories (IRD, see Section 9 of [RFC7285]). Considering the order and preference values, the client has to check these IRDs for a suitable Endpoint Cost Service and query it. The ECS is an optional service (see Section 11.5.1 of RFC 7285 [RFC7285]) and therefore, it may well be that an IRD does not refer to an ECS.

Calling the Cross-Domain Server Discovery Procedure only once with the single source address as a parameter - as opposed to multiple calls, e.g., one for each destination address - is not only a matter of efficiency. In the given scenario, it is advisable to send all ECS queries to the same ALTO server. This ensures that the results can be compared (e.g., for sorting candidate resource providers), even with cost metrics without a well-defined base unit, e.g., the "routingcost" metric.

2. More than one source addresses S1, S2, S3, ... and exactly one destination address D1. In this case, the ALTO client has to invoke the ALTO Cross-Domain Server Discovery Procedure with that single destination address as a parameter: `IRD_URIS_D1 = XDOMDISC(D1,"ALTO:https")`. The result `IRD_URIS_D1` is a list of one or more URIs of IRDs. Considering the order and preference values, the client has to check these IRDs for a suitable ECS and query it.
3. Exactly one source address S1 and exactly one destination address D1. The ALTO client may perform the same steps as in case 1, as specified above. As an alternative, it may also perform the same steps as in case 2, as specified above.
4. More than one source addresses S1, S2, S3, ... and more than one destination addresses D1, D2, D3, ... In this case, the ALTO client should split the list of desired queries based on source addresses and perform separately for each source address the same steps as in case 1, as specified above. As an alternative, the



ALTO client may also group the list based on destination addresses and perform separately for each destination address the same steps as in case 2, as specified above. However, comparing results between these sub-queries may be difficult, in particular if the cost metric is a relative preference without a well-defined base unit (e.g., the "routingcost" metric).

See Appendix C for a detailed example showing the interaction of a tracker-based peer-to-peer application, the ALTO Endpoint Cost Service, and the ALTO Cross-Domain Server Discovery Procedure.

#### 4.5. Summary and Further Extensions

Considering the four services defined in the ALTO base protocol specification [RFC7285], the ALTO Cross-Domain Server Discovery Procedure works best with the Endpoint Property Service (EPS) and the Endpoint Cost Service (ECS). Both the EPS and the ECS take one or more IP addresses as a parameter. The previous sections specify how the parameter for calling the ALTO Cross-Domain Server Discovery Procedure has to be derived from these IP addresses.

In contrast, the ALTO Cross-Domain Server Discovery Procedure seems less useful if the goal is to retrieve network and cost maps that cover the whole network topology. However, the procedure may be useful if a map centered at a specific IP address is desired (i.e., a map detailing the vicinity of said IP address or a map giving costs from said IP address to all potential destinations).

The interaction between further ALTO services (and their corresponding information resources) needs to be investigated and defined once such further ALTO services are specified in an extension document.

## 5. Implementation, Deployment, and Operational Considerations

### 5.1. Considerations for ALTO Clients

#### 5.1.1. Resource Consumer Initiated Discovery

Resource consumer initiated ALTO server discovery (c.f. ALTO requirement AR-32 [RFC6708]) can be seen as a special case of cross-domain ALTO server discovery. To that end, an ALTO client embedded in a resource consumer would have to perform the ALTO Cross-Domain Server Discovery Procedure with its own IP address as a parameter. However, due to the widespread deployment of Network Address Translators (NAT), additional protocols and mechanisms such as STUN [RFC5389] are usually needed to detect the client's "public" IP address, before it can be used as a parameter to the discovery procedure. Note that a different approach for resource consumer initiated ALTO server discovery, which is based on DHCP, is specified in [RFC7286].

#### 5.1.2. IPv4/v6 Dual Stack, Multihoming and Host Mobility

The procedure specified in this document can discover ALTO server URIs for a given IP address or prefix. The intention is, that a third party (e.g., a resource directory) that receives query messages from a resource consumer can use the source address in these messages to discover suitable ALTO servers for this specific resource consumer.

However, resource consumers (as defined in Section 2 of [RFC5693]) may reside on hosts with more than one IP address, e.g., due to IPv4/v6 dual stack operation and/or multihoming. IP packets sent with different source addresses may be subject to different routing policies and path costs. In some deployment scenarios, it may even be required to ask different sets of ALTO servers for guidance. Furthermore, source addresses in IP packets may be modified en-route by Network Address Translators (NAT).

If a resource consumer queries a resource directory for candidate resource providers, the locally selected (and possibly en-route translated) source address of the query message - as observed by the resource directory - will become the basis for the ALTO server discovery and the subsequent optimization of the resource directory's reply. If, however, the resource consumer then selects different source addresses to contact returned resource providers, the desired better-than-random "ALTO effect" may not occur.

One solution approach for this problem is, that a dual stack or multihomed resource consumer could always use the same address for

contacting the resource directory and all resource providers, thus overriding the operating system's automatic source IP address selection. For example, when using the BSD socket API, one could always `bind()` the socket to one of the local IP addresses before trying to `connect()` to the resource directory or the resource providers, respectively. Another solution approach is to perform ALTO-influenced resource provider selection (and source address selection) locally in the resource consumer, in addition to or instead of performing it in the resource directory. See Section 5.1.1 for a discussion how to discover ALTO servers for local usage in the resource consumer.

Similarly, resource consumers on mobile hosts SHOULD query the resource directory again after a change of IP address, in order to get a list of candidate resource providers that is optimized for the new IP address.

#### 5.1.3. Interaction with Network Address Translation

The ALTO Cross-Domain Server Discovery Procedure has been designed to enable the ALTO-based optimization of applications such as large-scale overlay networks, that span - on the IP layer - multiple administrative domains, possibly the whole Internet. Due to the widespread usage of Network Address Translators (NAT) it may well be that nodes of the overlay network (i.e., resource consumers or resource providers) are located behind a NAT, maybe even behind several cascaded NATs.

If a resource directory is located in the public Internet (i.e., not behind a NAT) and if it receives a message from a resource consumer behind one or more NATs, the message's source address will be the public IP address of the outermost NAT in front of the resource consumer. The same applies if the resource directory is behind a different NAT than the resource consumer. The resource directory may call the ALTO Cross-Domain Server Discovery Procedure with the message's source address as a parameter. In effect, not the resource consumer's (private) IP address, but the public IP address of the outermost NAT in front of it will be used as a basis for ALTO-optimization. This will work fine as long as the network behind the NAT is not too big (e.g., if the NAT is in a residential gateway).

If a resource directory receives a message from a resource consumer and the message's source address is a "private" IP address [RFC1918], this may be a sign that both of them are behind the same NAT. An invocation of the ALTO Cross-Domain Server Discovery Procedure with this private address may be problematic, as this will only yield usable results if a DNS "split horizon" and DNSSEC trust anchors are configured correctly. In this situation it may be more advisable to

query an ALTO server that has been discovered using [RFC7286] or any other local configuration. The interaction between intra-domain ALTO for large private domains (e.g., behind a "carrier-grade NAT") and cross-domain, Internet-wide optimization, is beyond the scope of this document.

## 5.2. Considerations for Network Operators

### 5.2.1. Flexibility vs. Load on the DNS

The ALTO Cross-Domain Server Discovery Procedure, as specified in Section 3, first produces a list of domain names (steps 1 and 2) and then looks for relevant NAPTR records associated with these names, until a useful result can be found (step 3). The number of candidate domain names on this list is a compromise between flexibility when installing NAPTR records and avoiding excess load on the DNS.

A single invocation of the ALTO Cross-Domain Server Discovery Procedure, with an IPv6 address as a parameter, may cause up to, but no more than, six DNS lookups for NAPTR records. For IPv4, the maximum is four lookups. Should the load on the DNS infrastructure caused by these lookups become a problem, one solution approach is to actually populate the DNS with ALTO-specific NAPTR records. If such records can be found for individual IP addresses (possibly installed using a wildcarding mechanism in the name server) or for long prefixes, the procedure will terminate successfully and not perform lookups for shorter prefix lengths, thus reducing the total number of DNS queries. Another approach for reducing the load on the DNS infrastructure is to increase the TTL for caching negative answers.

On the other hand, the ALTO Cross-Domain Server Discovery Procedure trying to lookup truncated domain names allows for efficient configuration of large-scale scenarios, where discovery is to be enabled for a large number of IP addresses with a small number of additional DNS resource records. Note that expressly, it has not been a design goal of this procedure to give clients a means to understand the IP prefix delegation structure. Furthermore, this specification does not assume or recommend that prefix delegations should preferably occur at those prefix lengths that are used in Step 2 of this procedure (see Section 3.3). A network operator that uses, for example, an IPv4 /18 prefix and wants to install the NAPTR records efficiently, could either install 64 NAPTR records (one for each of the /24 prefixes contained within the /18 prefix), or they could try to team up with the owners of the other fragments of the enclosing /16 prefix, in order to run a common ALTO server to which only one NAPTR would point.

### 5.2.2. BCP20 and missing delegations of the reverse DNS

RFC2317 [RFC2317], also known as BCP20, describes a way to delegate the "reverse DNS" (i.e., subdomains of in-addr.arpa.) for IPv4 address ranges with fewer than 256 addresses (i.e., less than a whole /24 prefix). The ALTO Cross-Domain Server Discovery procedure is compatible with this method.

In some deployment scenarios, e.g., residential Internet access, where customers often dynamically receive a single IPv4 address (and/or a small IPv6 address block) from a pool of addresses, ISPs typically will not delegate the "reverse DNS" to their customers. This practice makes it impossible for these customers to populate the DNS with NAPTR resource records that point to an ALTO server of their choice. Yet, the ISP may publish NAPTR resource records in the "reverse DNS" for individual addresses or larger address pools (i.e., shorter prefix lengths).

While ALTO is by no means technologically tied to the Border Gateway Protocol (BGP), it is anticipated that BGP will be an important source of information for ALTO and that the operator of the outermost BGP-enabled router will have a strong incentive to publish a digest of their routing policies and costs through ALTO. In contrast, an individual user or an organization that has been assigned only a small address range (i.e., an IPv4 prefix with a prefix length longer than /24) will typically connect to the Internet using only a single ISP, and they might not be interested in publishing their own ALTO information. Consequently, they might wish to leave the operation of an ALTO server up to their ISP. This ISP may install NAPTR resource records, which are needed for the ALTO Cross-Domain Server Discovery procedure, in the subdomain of in-addr.arpa. that corresponds to the whole /24 prefix (c.f., R24 in Section 3.3 of this document), even if BCP20-style delegations or no delegations at all are in use.

## 6. Security Considerations

A high-level discussion of security issues related to ALTO is part of the ALTO problem statement [RFC5693]. A classification of unwanted information disclosure risks, as well as specific security-related requirements can be found in the ALTO requirements document [RFC6708].

The remainder of this section focuses on security threats and protection mechanisms for the cross-domain ALTO server discovery procedure as such. Once the ALTO server's URI has been discovered and the communication between the ALTO client and the ALTO server starts, the security threats and protection mechanisms discussed in the ALTO protocol specification [RFC7285] apply.

### 6.1. Integrity of the ALTO Server's URI

#### Scenario Description

An attacker could compromise the ALTO server discovery procedure or the underlying infrastructure in a way that ALTO clients would discover a "wrong" ALTO server URI.

#### Threat Discussion

The cross-domain ALTO server discovery procedure relies on a series of DNS lookups, in order to produce one or more URI(s). If an attacker was able to modify or spoof any of the DNS records, the resulting URI(s) could be replaced by forged URI(s). This is probably the most serious security concern related to ALTO server discovery. The discovered "wrong" ALTO server might not be able to give guidance to a given ALTO client at all, or it might give suboptimal or forged information. In the latter case, an attacker could try to use ALTO to affect the traffic distribution in the network or the performance of applications (see also Section 15.1. of [RFC7285]). Furthermore, a hostile ALTO server could threaten user privacy (see also Section 5.2.1, case (5a) in [RFC6708]).

#### Protection Strategies and Mechanisms

The application of DNS security (DNSSEC) [RFC4033] provides a means to detect and avert attacks that rely on modification of the DNS records while in transit. All implementations of the cross-domain ALTO server discovery procedure MUST support DNSSEC or be able to use such functionality provided by the underlying operating system. Network operators that publish U-NAPTR resource records to be used for the cross-domain ALTO server discovery procedure SHOULD use DNSSEC to protect their subdomains of in-addr.arpa. and/or ip6.arpa., respectively. Additional operational precautions for safely operating the DNS infrastructure are required in order to ensure that name servers do not sign forged

(or otherwise "wrong") resource records. Security considerations specific to U-NAPTR are described in more detail in [RFC4848].

In addition to active protection mechanisms, users and network operators can monitor application performance and network traffic patterns for poor performance or abnormalities. If it turns out that relying on the guidance of a specific ALTO server does not result in better-than-random results, the usage of the ALTO server may be discontinued (see also Section 15.2 of [RFC7285]).

#### Note

The cross-domain ALTO server discovery procedure finishes successfully when it has discovered one or more URI(s). Once an ALTO server's URI has been discovered and the communication between the ALTO client and the ALTO server starts, the security threats and protection mechanisms discussed in the ALTO protocol specification [RFC7285] apply.

A threat related to the one considered above is the impersonation of an ALTO server after its correct URI has been discovered. This threat and protection strategies are discussed in Section 15.1 of [RFC7285]. The ALTO protocol's primary mechanism for protecting authenticity and integrity (as well as confidentiality) is the use of HTTPS-based transport, i.e., HTTP over TLS [RFC2818]. Typically, when the URI's host component is a host name, a further DNS lookup is needed to map it to an IP address, before the communication with the server can begin. This last DNS lookup (for A or AAAA resource records) does not necessarily have to be protected by DNSSEC, as the server identity checks specified in [RFC2818] are able to detect DNS spoofing or similar attacks, after the connection to the (possibly wrong) host has been established. However, this validation, which is based on the server certificate, can only protect the steps that occur after the server URI has been discovered. It cannot detect attacks against the authenticity of the U-NAPTR lookups needed for the cross-domain ALTO server discovery procedure, and therefore, these resource records have to be secured using DNSSEC.

## 6.2. Availability of the ALTO Server Discovery Procedure

### Scenario Description

An attacker could compromise the cross-domain ALTO server discovery procedure or the underlying infrastructure in a way that ALTO clients would not be able to discover any ALTO server.

#### Threat Discussion

If no ALTO server can be discovered (although a suitable one exists) applications have to make their decisions without ALTO guidance. As ALTO could be temporarily unavailable for many reasons, applications must be prepared to do so. However, The resulting application performance and traffic distribution will correspond to a deployment scenario without ALTO.

#### Protection Strategies and Mechanisms

Operators should follow best current practices to secure their DNS and ALTO (see Section 15.5 of [RFC7285]) servers against Denial-of-Service (DoS) attacks.

### 6.3. Confidentiality of the ALTO Server's URI

#### Scenario Description

An unauthorized party could invoke the cross-domain ALTO server discovery procedure, or intercept discovery messages between an authorized ALTO client and the DNS servers, in order to acquire knowledge of the ALTO server URI for a specific IP address.

#### Threat Discussion

In the ALTO use cases that have been described in the ALTO problem statement [RFC5693] and/or discussed in the ALTO working group, the ALTO server's URI as such has always been considered as public information that does not need protection of confidentiality.

#### Protection Strategies and Mechanisms

No protection mechanisms for this scenario have been provided, as it has not been identified as a relevant threat. However, if a new use case is identified that requires this kind of protection, the suitability of this ALTO server discovery procedure as well as possible security extensions have to be re-evaluated thoroughly.

### 6.4. Privacy for ALTO Clients

#### Scenario Description

An unauthorized party could eavesdrop on the messages between an ALTO client and the DNS servers, and thereby find out the fact that said ALTO client uses (or at least tries to use) the ALTO service in order to optimize traffic from/to a specific IP address.

#### Threat Discussion

In the ALTO use cases that have been described in the ALTO problem statement [RFC5693] and/or discussed in the ALTO working group, this scenario has not been identified as a relevant threat. However, Pervasive Surveillance [RFC7624] and DNS Privacy



Considerations [RFC7626] have seen significant attention in the Internet community in recent years.

#### Protection Strategies and Mechanisms

DNS over TLS [RFC7858] and DNS over HTTPS [RFC8484] provide means for protecting confidentiality (and integrity) of DNS traffic between a client (stub) and its recursive name servers, including DNS queries and replies caused by the ALTO Cross-Domain Server Discovery Procedure.

## 7. IANA Considerations

This document does not require any IANA action.

## 8. References

### 8.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [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>>.
- [RFC3403] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database", RFC 3403, DOI 10.17487/RFC3403, October 2002, <<https://www.rfc-editor.org/info/rfc3403>>.
- [RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", STD 88, RFC 3596, DOI 10.17487/RFC3596, October 2003, <<https://www.rfc-editor.org/info/rfc3596>>.
- [RFC4848] Daigle, L., "Domain-Based Application Service Location Using URIs and the Dynamic Delegation Discovery Service (DDDS)", RFC 4848, DOI 10.17487/RFC4848, April 2007, <<https://www.rfc-editor.org/info/rfc4848>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

### 8.2. Informative References

- [I-D.kiesel-alto-alto4alto] Kiesel, S., "Using ALTO for ALTO server selection", draft-kiesel-alto-alto4alto-00 (work in progress), July 2010.
- [I-D.kiesel-alto-ip-based-srv-disc] Kiesel, S. and R. Penno, "Application-Layer Traffic Optimization (ALTO) Anycast Address", draft-kiesel-alto-ip-based-srv-disc-03 (work in progress), July 2014.
- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996,

<<https://www.rfc-editor.org/info/rfc1918>>.

- [RFC2317] Eidnes, H., de Groot, G., and P. Vixie, "Classless IN-ADDR.ARPA delegation", BCP 20, RFC 2317, DOI 10.17487/RFC2317, March 1998, <<https://www.rfc-editor.org/info/rfc2317>>.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<https://www.rfc-editor.org/info/rfc2818>>.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <<https://www.rfc-editor.org/info/rfc4632>>.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <<https://www.rfc-editor.org/info/rfc5389>>.
- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, DOI 10.17487/RFC5693, October 2009, <<https://www.rfc-editor.org/info/rfc5693>>.
- [RFC6708] Kiesel, S., Ed., Previdi, S., Stiemerling, M., Woundy, R., and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Requirements", RFC 6708, DOI 10.17487/RFC6708, September 2012, <<https://www.rfc-editor.org/info/rfc6708>>.
- [RFC7216] Thomson, M. and R. Bellis, "Location Information Server (LIS) Discovery Using IP Addresses and Reverse DNS", RFC 7216, DOI 10.17487/RFC7216, April 2014, <<https://www.rfc-editor.org/info/rfc7216>>.
- [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>>.
- [RFC7286] Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and H. Song, "Application-Layer Traffic Optimization (ALTO) Server Discovery", RFC 7286, DOI 10.17487/RFC7286, November 2014, <<https://www.rfc-editor.org/info/rfc7286>>.
- [RFC7624] Barnes, R., Schneier, B., Jennings, C., Hardie, T., Trammell, B., Huitema, C., and D. Borkmann, "Confidentiality in the Face of Pervasive Surveillance: A Threat Model and Problem Statement", RFC 7624, DOI 10.17487/RFC7624, August 2015, <<https://www.rfc-editor.org/info/rfc7624>>.
- [RFC7626] Bortzmeyer, S., "DNS Privacy Considerations", RFC 7626, DOI 10.17487/RFC7626, August 2015, <<https://www.rfc-editor.org/info/rfc7626>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC7971] Stiemerling, M., Kiesel, S., Scharf, M., Seidel, H., and S. Previdi, "Application-Layer Traffic Optimization (ALTO) Deployment Considerations", RFC 7971, DOI 10.17487/RFC7971, October 2016, <<https://www.rfc-editor.org/info/rfc7971>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.

## Appendix A. Solution Approaches for Partitioned ALTO Knowledge

The ALTO base protocol document [RFC7285] specifies the communication between an ALTO client and a single ALTO server. It is implicitly assumed that this server can answer any query, possibly with some kind of default value if no exact data is known. No special provisions were made for the case that the ALTO information originates from multiple sources, which are possibly under the control of different administrative entities (e.g., different ISPs) or that the overall ALTO information is partitioned and stored on several ALTO servers.

### A.1. Classification of Solution Approaches

Various protocol extensions and other solutions have been proposed to deal with multiple information sources and partitioned knowledge. They can be classified as follows:

- 1    Ensure that all ALTO servers have the same knowledge
  - 1.1   Ensure data replication and synchronization within the provisioning protocol (cf. RFC 5693, Fig 1 [RFC5693]).
  - 1.2   Use an Inter-ALTO-server data replication protocol. Possibly, the ALTO protocol itself – maybe with some extensions – could be used for that purpose; however, this has not been studied in detail so far.
- 2    Accept that different ALTO servers (possibly operated by different organizations, e.g., ISPs) do not have the same knowledge
  - 2.1   Allow ALTO clients to send arbitrary queries to any ALTO server (e.g. the one discovered using [RFC7286]). If this server cannot answer the query itself, it will fetch the data on behalf of the client, using the ALTO protocol or a to-be-defined inter-ALTO-server request forwarding protocol.
  - 2.2   Allow ALTO clients to send arbitrary queries to any ALTO server (e.g. the one discovered using [RFC7286]). If this server cannot answer the query itself, it will redirect the client to the "right" ALTO server that has the desired information, using a small to-be-defined extension of the ALTO protocol.
  - 2.3   ALTO clients need to use some kind of "search engine" that indexes ALTO servers and redirects and/or gives cached results.

- 2.4 ALTO clients need to use a new discovery mechanism to discover the ALTO server that has the desired information and contact it directly.

#### A.2. Discussion of Solution Approaches

The provisioning or initialization protocol for ALTO servers (cf. RFC 5693, Fig 1 [RFC5693]) is currently not standardized. It was a conscious decision not to include this in the scope of the IETF ALTO working group. The reason is that there are many different kinds of information sources. This implementation specific protocol will adapt them to the ALTO server, which offers a standardized protocol to the ALTO clients. However, adding the task of synchronization between ALTO servers to this protocol (i.e., approach 1.1) would overload this protocol with a second functionality that requires standardization for seamless multi-domain operation.

For the 1.? solution approaches, in addition to general technical feasibility and issues like overhead and caching efficiency, another aspect to consider is legal liability. Operator "A" might prefer not to publish information about nodes in or paths between the networks of operators "B" and "C" through A's ALTO server, even if A knew that information. This is not only a question of map size and processing load on A's ALTO server. Operator A could also face legal liability issues if that information had a bad impact on the traffic engineering between B's and C's networks, or on their business models.

No specific actions to build a "search engine" based solution (approach 2.3) are currently known and it is unclear what could be the incentives to operate such an engine. Therefore, this approach is not considered in the remainder of this document.

#### A.3. The Need for Cross-Domain ALTO Server Discovery

Approaches 1.1, 1.2, 2.1, and 2.2 do not only require the specification of an ALTO protocol extension or a new protocol that runs between ALTO servers. A large-scale, maybe Internet-wide, multi-domain deployment would also need mechanisms by which an ALTO server could discover other ALTO servers, learn which information is available where, and ideally also who is authorized to publish information related to a given part of the network. Approach 2.4 needs the same mechanisms, except that they are used on the client-side instead of the server-side.

It is sometimes questioned whether there is a need for a solution that allows clients to ask arbitrary queries, even if the ALTO information is partitioned and stored on many ALTO servers. The main

argument is, that clients are supposed to optimize the traffic from and to themselves, and that the information needed for that is most likely stored on a "nearby" ALTO server, i.e., the one that can be discovered using [RFC7286]. However, there are scenarios where the ALTO client is not co-located with an endpoint of the to-be-optimized data transmission. Instead, the ALTO client is located at a third party, which takes part in the application signaling, e.g., a so-called "tracker" in a peer-to-peer application. One such scenario, where it is advantageous to place the ALTO client not at an endpoint of the user data transmission, is analyzed in Appendix C.

#### A.4. Our Solution Approach

Several solution approaches for cross-domain ALTO server discovery have been evaluated, using the criteria documented in Appendix B. One of them was to use the ALTO protocol itself for the exchange of information availability [I-D.kiesel-alto-alto4alto]. However, the drawback of that approach is that a new registration administration authority would have to be established.

This document specifies a DNS-based procedure for cross-domain ALTO server discovery, which was inspired by "Location Information Server (LIS) Discovery Using IP Addresses and Reverse DNS" [RFC7216]. The primary goal is that this procedure can be used on the client-side (i.e., approach 2.4), but together with new protocols or protocol extensions it could also be used to implement the other solution approaches itemized above.

#### A.5. Relation to the ALTO Requirements

During the design phase of the overall ALTO solution, two different server discovery scenarios have been identified and documented in the ALTO requirements document [RFC6708]. The first scenario, documented in Req. AR-32, can be supported using the discovery mechanisms specified in [RFC7286]. An alternative approach, based on IP anycast [I-D.kiesel-alto-ip-based-srv-disc], has also been studied. This document, in contrast, tries to address Req. AR-33.



## Appendix B. Requirements for Cross-Domain Server Discovery

This appendix itemizes requirements that have been collected before the design phase and that are reflected by the design of the ALTO Cross-Domain Server Discovery Procedure.

### B.1. Discovery Client Application Programming Interface

The discovery client will be called through some kind of application programming interface (API) and the parameters will be an IP address and, for purposes of extensibility, a service identifier such as "ALTO". It will return one or more URI(s) that offers the requested service ("ALTO") for the given IP address.

In other words, the client would be used to retrieve a mapping:

(IP address, "ALTO") -> IRD-URI(s)

where IRD-URI(s) is one or more URI(s) of Information Resource Directories (IRD, see Section 9 of [RFC7285]) of ALTO server(s) that can give reasonable guidance to a resource consumer with the indicated IP address.

### B.2. Data Storage and Authority Requirements

The information for mapping IP addresses and service parameters to URIs should be stored in a - preferably distributed - database. It must be possible to delegate administration of parts of this database. Usually, the mapping from a specific IP address to an URI is defined by the authority that has administrative control over this IP address, e.g., the ISP in residential access networks or the IT department in enterprise, university, or similar networks.

### B.3. Cross-Domain Operations Requirements

The cross-domain server discovery mechanism should be designed in such a way that it works across the public Internet and also in other IP-based networks. This in turn means that such mechanisms cannot rely on protocols that are not widely deployed across the Internet or protocols that require special handling within participating networks. An example is multicast, which is not generally available across the Internet.

The ALTO Cross-Domain Server Discovery protocol must support gradual deployment without a network-wide flag day. If the mechanism needs some kind of well-known "rendezvous point", re-using an existing infrastructure (such as the DNS root servers or the WHOIS database) should be preferred over establishing a new one.

#### B.4. Protocol Requirements

The protocol must be able to operate across middleboxes, especially across NATs and firewalls.

The protocol shall not require any pre-knowledge from the client other than any information that is known to a regular IP host on the Internet.

#### B.5. Further Requirements

The ALTO cross domain server discovery cannot assume that the server discovery client and the server discovery responding entity are under the same administrative control.

## Appendix C. ALTO and Tracker-based Peer-to-Peer Applications

This appendix provides a complete example of using ALTO and the ALTO Cross-Domain Server Discovery Procedure in one specific application scenario, namely a tracker-based peer-to-peer application. First, in subsection C.1, we introduce a generic model of such an application and show why ALTO optimization is desirable. Then, in C.2, we introduce two architectural options for integrating ALTO into the tracker-based peer-to-peer application - one option is based on the "regular" ALTO server discovery procedure [RFC7286], one relies on the ALTO Cross-Domain Server Discovery Procedure. In C.3, a simple mathematical model is used to show that the latter approach is expected to yield significantly better optimization results. The appendix concludes with subsection C.4, which details an exemplary complete walk-through of the ALTO Cross-Domain Server Discovery Procedure.

### C.1. A generic Tracker-based Peer-to-Peer Application

The optimization of peer-to-peer (P2P) applications such as BitTorrent was one of the first use cases that lead to the inception of the IETF ALTO working group. Further use cases have been identified as well, yet we will use this scenario to illustrate the operation and usefulness of the ALTO Cross-Domain Server Discovery Procedure.

For the remainder of this chapter we consider a generic, tracker-based peer-to-peer file sharing application. The goal is the dissemination of a large file, without using one large server with a correspondingly high upload bandwidth. The file is split into chunks. So-called "peers" assume the role of both a client and a server. That is, they may request chunks from other peers and they may serve the chunks they already possess to other peers at the same time, thereby contributing their upload bandwidth. Peers that want to share the same file participate in a "swarm". They use the peer-to-peer protocol to inform each other about the availability of chunks and to request and transfer chunks from one peer to another. A swarm may consist of a very large number of peers. Consequently, peers usually maintain logical connections only to a subset of all peers in the swarm. If a new peer wants to join a swarm, it first contacts a well-known server, the "tracker", which provides a list of IP addresses of peers in the swarm.

A swarm is an overlay network on top of the IP network. Algorithms that determine the overlay topology and the traffic distribution in the overlay may consider information about the underlying IP network, such as topological distance, link bandwidth, (monetary) costs for sending traffic from one host to another, etc. ALTO is a protocol

for retrieving such information. The goal of such "topology aware" decisions is to improve performance or Quality of Experience in the application while reducing the utilization of the underlying network infrastructure.

## C.2. Architectural Options for Placing the ALTO Client

The ALTO protocol specification [RFC7285] details how an ALTO client can query an ALTO server for guiding information and receive the corresponding replies. However, in the considered scenario of a tracker-based P2P application, there are two fundamentally different possibilities where to place the ALTO client:

1. ALTO client in the resource consumer ("peer")
2. ALTO client in the resource directory ("tracker")

In the following, both scenarios are compared in order to explain the need for ALTO queries on behalf of remote resource consumers.

In the first scenario (see Figure 2), the resource consumer queries the resource directory for the desired resource (F1). The resource directory returns a list of potential resource providers without considering ALTO (F2). It is then the duty of the resource consumer to invoke ALTO (F3/F4), in order to solicit guidance regarding this list.

In the second scenario (see Figure 4), the resource directory has an embedded ALTO client. After receiving a query for a given resource (F1) the resource directory invokes this ALTO client to evaluate all resource providers it knows (F2/F3). Then it returns a, possibly shortened, list containing the "best" resource providers to the resource consumer (F4).

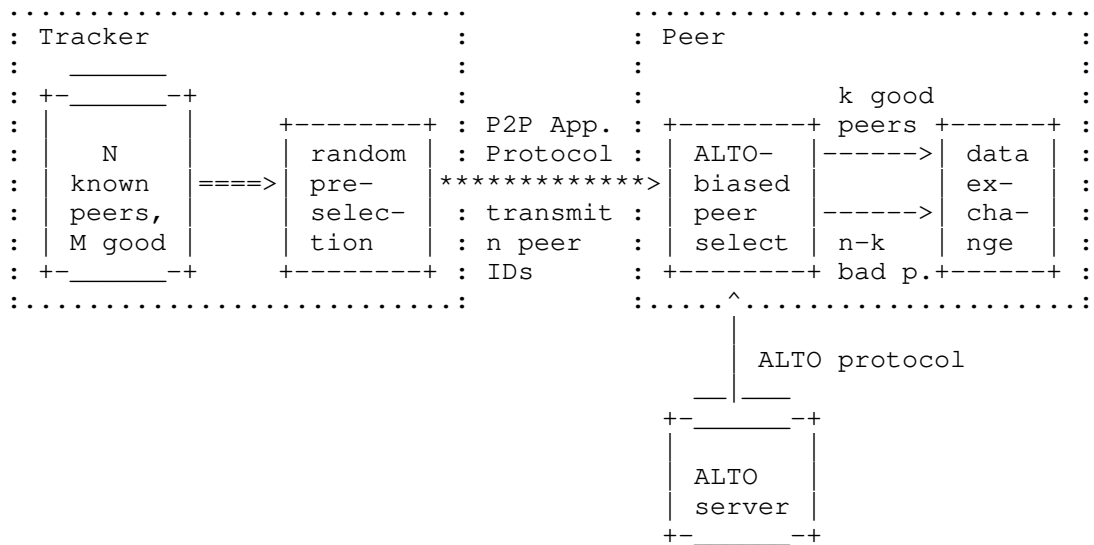


Figure 1: Tracker-based P2P Application with random peer preselection

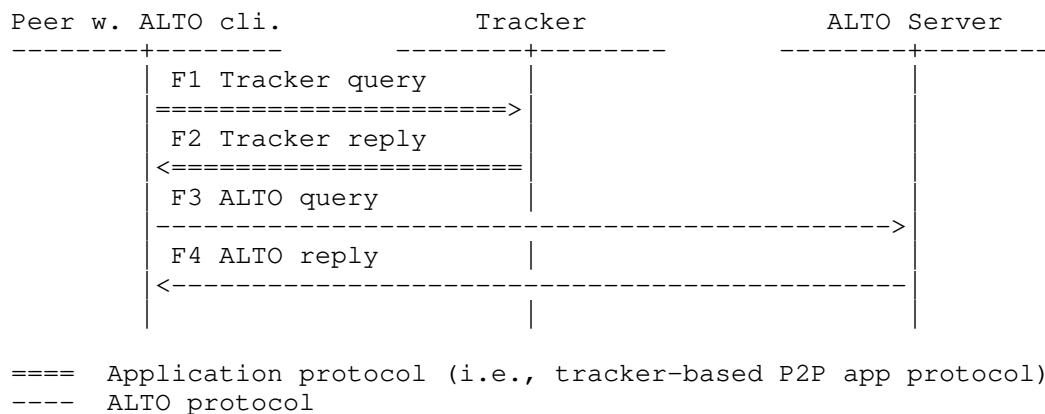


Figure 2: Basic message sequence chart for resource consumer-initiated ALTO query

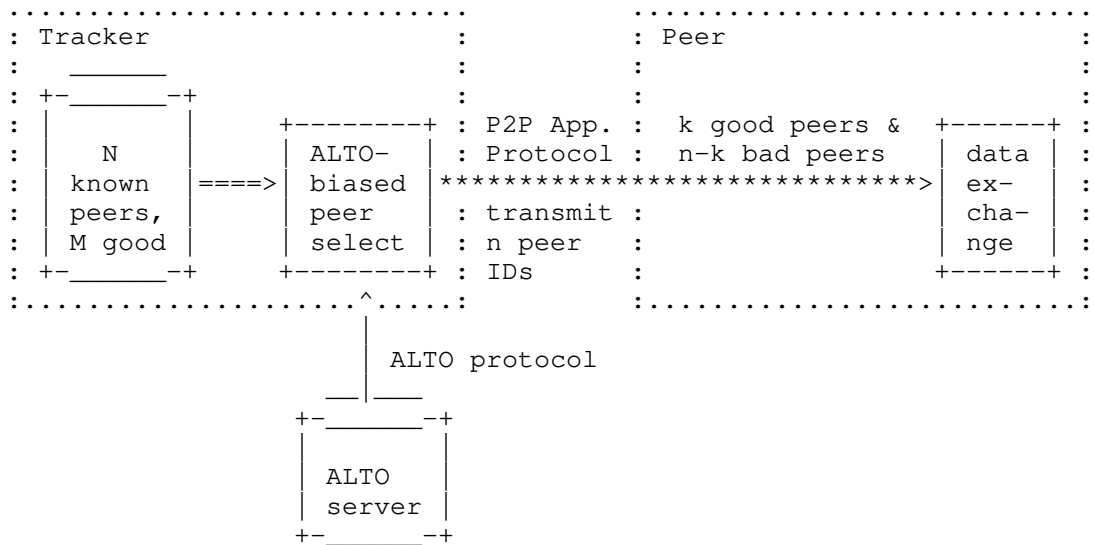


Figure 3: Tracker-based P2P Application with ALTO client in tracker

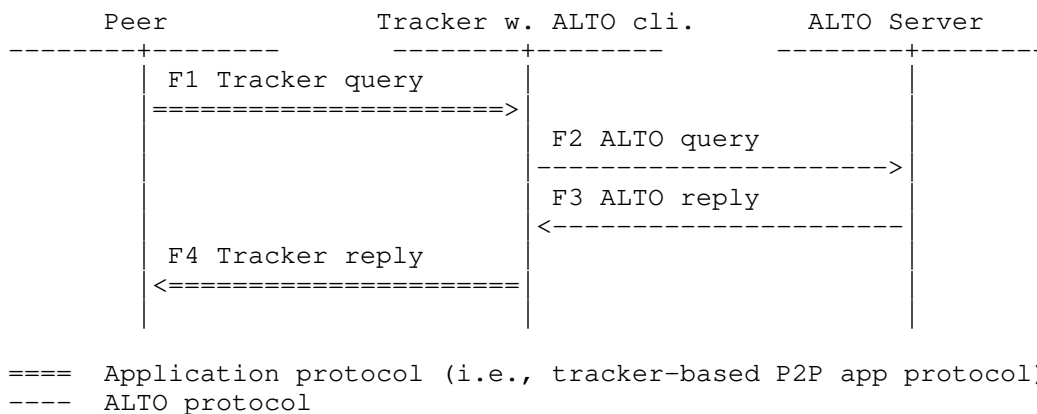


Figure 4: Basic message sequence chart for ALTO query on behalf of remote resource consumer

Note: the message sequences depicted in Figure 2 and Figure 4 may occur both in the target-aware and the target-independent query mode (c.f. [RFC6708]). In the target-independent query mode no message exchange with the ALTO server might be needed after the tracker query, because the candidate resource providers could be evaluated using a locally cached "map", which has been retrieved from the ALTO server some time ago.

## C.3. Evaluation

The problem with the first approach is, that while the resource directory might know thousands of peers taking part in a swarm, the list returned to the resource consumer is usually shortened for efficiency reasons. Therefore, the "best" (in the sense of ALTO) potential resource providers might not be contained in that list anymore, even before ALTO can consider them.

For illustration, consider a simple model of a swarm, in which all peers fall into one of only two categories: assume that there are "good" ("good" in the sense of ALTO's better-than-random peer selection, based on an arbitrary desired rating criterion) and "bad" peers only. Having more different categories makes the maths more complex but does not change anything to the basic outcome of this analysis. Assume that the swarm has a total number of  $N$  peers, out of which are  $M$  "good" and  $N-M$  "bad" peers, which are all known to the tracker. A new peer wants to join the swarm and therefore asks the tracker for a list of peers.

If, according to the first approach, the tracker randomly picks  $n$  peers from the  $N$  known peers, the result can be described with the hypergeometric distribution. The probability that the tracker reply contains exactly  $k$  "good" peers (and  $n-k$  "bad" peers) is:

$$P(X=k) = \frac{\frac{M!}{k!(M-k)!} \cdot \frac{(N-M)!}{(n-k)!(N-M-n+k)!}}{\frac{N!}{n!(N-n)!}}$$

$$\text{with } \frac{n!}{k!(n-k)!} = \frac{n!}{k! (n-k)!} \quad \text{and} \quad n! = n * (n-1) * (n-2) * \dots * 1$$

The probability that the reply contains at most  $k$  "good" peers is:  
 $P(X \leq k) = P(X=0) + P(X=1) + \dots + P(X=k)$ .

For example, consider a swarm with  $N=10,000$  peers known to the tracker, out of which  $M=100$  are "good" peers. If the tracker randomly selects  $n=100$  peers, the formula yields for the reply:  $P(X=0)=36\%$ ,  $P(X \leq 4)=99\%$ . That is, with a probability of approx. 36% this list does not contain a single "good" peer, and with 99% probability there are only four or less of the "good" peers on the

list. Processing this list with the guiding ALTO information will ensure that the few favorable peers are ranked to the top of the list; however, the benefit is rather limited as the number of favorable peers in the list is just too small.

Much better traffic optimization could be achieved if the tracker would evaluate all known peers using ALTO, and return a list of 100 peers afterwards. This list would then include a significantly higher fraction of "good" peers. (Note, that if the tracker returned "good" peers only, there might be a risk that the swarm might disconnect and split into several disjunct partitions. However, finding the right mix of ALTO-biased and random peer selection is out of the scope of this document.)

Therefore, from an overall optimization perspective, the second scenario with the ALTO client embedded in the resource directory is advantageous, because it is ensured that the addresses of the "best" resource providers are actually delivered to the resource consumer. An architectural implication of this insight is that the ALTO server discovery procedures must support ALTO queries on behalf of remote resource consumers. That is, as the tracker issues ALTO queries on behalf of the peer which contacted the tracker, the tracker must be able to discover an ALTO server that can give guidance suitable for that respective peer. This task can be solved using the ALTO Cross-Domain Server Discovery Procedure.





In theory, there are many options how the ALTO Cross-Domain Server Discovery Procedure could be used. For example, the tracker could do the following steps:

```
IRD_URIS_A = XDOMDISC(A, "ALTO:https")
COST_X_A   = query the ECS(X,A, routingcost) found in IRD_URIS_A

IRD_URIS_B = XDOMDISC(B, "ALTO:https")
COST_X_B   = query the ECS(X,B, routingcost) found in IRD_URIS_B

IRD_URIS_C = XDOMDISC(C, "ALTO:https")
COST_X_C   = query the ECS(X,C, routingcost) found in IRD_URIS_C
```

Maybe, the ALTO Cross-Domain Server Discovery Procedure queries would yield in this scenario: `IRD_URIS_A = ALTO_SRV_A`, `IRD_URIS_B = ALTO_SRV_B`, and `IRD_URIS_C = ALTO_SRV_C`. That is, each ECS query would be sent to a different ALTO server. The problem with this approach is that we are not necessarily able to compare `COST_X_A`, `COST_X_B`, and `COST_X_C` with each other. The specification of the routingcost metric mandates that "A lower value indicates a higher preference", but "an ISP may internally compute routing cost using any method that it chooses" (see section 6.1.1.1. of [RFC7285]). Thus, `COST_X_A` could be 10 (milliseconds round-trip time), while `COST_X_B` could be 200 (kilometers great circle distance between the approximate geographic locations of the hosts) and `COST_X_C` could be 3 (router hops, corresponding to a decrease of the TTL field in the IP header). Each of these metrics fulfills the "lower value is more preferable" requirement on its own, but obviously, they cannot be compared with each other. Even if there was a reasonable formula to compare, for example, kilometers with milliseconds, we could not use it, as the units of measurement (or any other information about the computation method for the routingcost) are not sent along with the value in the ECS reply.

To avoid this problem, the tracker tries to send all ECS queries to the same ALTO server. As specified in Section 4.4 of this document, case 2, it uses the IP address of Resource Consumer x as parameter to the discovery procedure:

```
IRD_URIS_X = XDOMDISC(X, "ALTO:https")
COST_X_A   = query the ECS(X,A, routingcost) found in IRD_URIS_X
COST_X_B   = query the ECS(X,B, routingcost) found in IRD_URIS_X
COST_X_C   = query the ECS(X,C, routingcost) found in IRD_URIS_X
```

This strategy ensures that `COST_X_A`, `COST_X_B`, and `COST_X_C` can be compared with each other.

As discussed above, the tracker calls the ALTO Cross-Domain Server Discovery Procedure with IP address X as a parameter. For the remainder of this example, we assume that X = 2001:DB8:1:2:227:eff:fe6a:de42. Thus, the procedure call is

```
IRD_URIS_X = XDOMDISC(2001:DB8:1:2:227:eff:fe6a:de42,"ALTO:https").
```

The first parameter 2001:DB8:1:2:227:eff:fe6a:de42 is a single IPv6 address. Thus, we get AT = IPv6, A = 2001:DB8:1:2:227:eff:fe6a:de42, L = 128, and SP = "ALTO:https".

The procedure constructs (see Step 1 in Section 3.2)  
R128 = "2.4.E.D.A.6.E.F.F.F.E.0.7.2.2.0.2.0.0.0.1.0.0.0.  
8.B.D.0.1.0.0.2.IP6.ARPA.", as well as (see Step 2 in Section 3.3)  
R64 = "2.0.0.0.1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.",  
R56 = "0.0.1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.",  
R48 = "1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.",  
R40 = "0.0.8.B.D.0.1.0.0.2.IP6.ARPA.", and  
R32 = "8.B.D.0.1.0.0.2.IP6.ARPA.".

In order to illustrate the third step of the ALTO Cross-Domain Server Discovery Procedure, we use the "dig" (domain information groper) DNS lookup utility that is available for many operating systems (e.g., Linux). A real implementation of the ALTO Cross-Domain Server Discovery Procedure would not be based on the "dig" utility, but use appropriate libraries and/or operating system APIs. Please note that the following steps have been performed in a controlled lab environment with a appropriately configured name server. A suitable DNS configuration will be needed to reproduce these results. Please also note that the rather verbose output of the "dig" tool has been shortened to the relevant lines.

Since AT = IPv6 and L = 128, in the table given in Section 3.4, the sixth row (not counting the column headers) applies.

As mandated by the third column, we start with a lookup of R128, looking for NAPTR resource records:

```
user@labpc:~$ dig -tNAPTR 2.4.E.D.A.6.E.F.F.F.E.0.7.2.2.0.\
2.0.0.0.1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 26553
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADD'L: 0
```

The domain name R128 does not exist (status: NXDOMAIN), so we cannot get a useful result. Therefore, we continue with the fourth column of the table and do a lookup of R64:

```
user@labpc:~$ dig -tNAPTR 2.0.0.0.1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33193
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADD'L: 0
```

The domain name R64 could be looked up (status: NOERROR), but there are no NAPTR resource records associated with it (ANSWER: 0). Maybe, there are some other resource records such as PTR, NS, or SOA, but we are not interested in them. Thus, we do not get a useful result and we continue with looking up R56:

```
user@labpc:~$ dig -tNAPTR 0.0.1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35966
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADD'L: 2

;; ANSWER SECTION:
0.0.1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA. 604800 IN NAPTR 100 10 "u"
    "LIS:HELD" "!.*!https://lis1.example.org:4802/?c=ex!" .
0.0.1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA. 604800 IN NAPTR 100 20 "u"
    "LIS:HELD" "!.*!https://lis2.example.org:4802/?c=ex!" .
```

The domain name R56 could be looked up and there are NAPTR resource records associated with it. However, each of these records has a service parameter that does not match our SP = "ALTO:https" (see [RFC7216] for "LIS:HELD"), and therefore, we have to ignore them. Consequently, we still do not have a useful result and continue with a lookup of R48:

```
user@labpc:~$ dig -tNAPTR 1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50459
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADD'L: 2

;; ANSWER SECTION:
1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA. 604800 IN NAPTR 100 10 "u"
    "ALTO:https" "!.*!https://alto1.example.net/ird!" .
1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA. 604800 IN NAPTR 100 10 "u"
    "LIS:HELD" "!.*!https://lis.example.net:4802/?c=ex!" .
```

This lookup yields two NAPTR resource records. We have to ignore the second one as its service parameter does not match our SP, but the first NAPTR resource record has a matching service parameter. Therefore, the procedure terminates successfully and the final outcome is: IRD\_URIS\_X = "https://alto1.example.net/ird".

The ALTO client that is embedded in the tracker will access the ALTO Information Resource Directory (IRD, see Section 9 of [RFC7285]) at this URI, look for the Endpoint Cost Service (ECS, see Section 11.5 of [RFC7285]), and query the ECS for the costs between A and X, B and X, as well as C and X, before returning an ALTO-optimized list of candidate resource providers to resource consumer X.

## Appendix D. Contributors List and Acknowledgments

The initial version of this document was co-authored by Marco Tomsu (Alcatel-Lucent).

This document borrows some text from [RFC7286], as historically, it has been part of the draft that eventually became said RFC. Special thanks to Michael Scharf and Nico Schwan.

## Authors' Addresses

Sebastian Kiesel  
University of Stuttgart Information Center  
Allmandring 30  
Stuttgart 70550  
Germany

Email: [ietf-alto@skiesel.de](mailto:ietf-alto@skiesel.de)  
URI: <http://www.izus.uni-stuttgart.de>

Martin Stiernerling  
University of Applied Sciences Darmstadt, Computer Science Dept.  
Haardtring 100  
Darmstadt 64295  
Germany

Phone: +49 6151 16 37938  
Email: [mls.ietf@gmail.com](mailto:mls.ietf@gmail.com)  
URI: <http://ietf.stiernerling.org>





ALTO  
Internet-Draft  
Intended status: Experimental  
Expires: January 1, 2018

S. Randriamasy  
Nokia-Bell-labs  
June 30, 2017

ALTO cellular addresses  
draft-randriamasy-alto-cellular-adresses-00

Abstract

This draft proposes to use the cellular address format composed of elements as specified by 3GPP and called ECGI. ECGI stands for E-UTRAN Cell Global Identifier and is used in Public Land Mobile Networks based on E-UTRAN.

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 1, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Relevant ALTO services and documents . . . . .	3
3. Proposed format for ALTO cell identifiers . . . . .	3
3.1. ALTO address type for cellular networks . . . . .	3
3.2. Endpoint address canonical string format . . . . .	3
3.3. ALTO Cell Id formats . . . . .	4
3.4. Examples . . . . .	4
4. IANA Considerations . . . . .	5
5. Security Considerations . . . . .	5
6. Acknowledgements . . . . .	5
7. References . . . . .	5
7.1. Normative References . . . . .	5
7.2. Informative References . . . . .	5
Appendix A. An Appendix . . . . .	5
Author's Address . . . . .	5

## 1. Introduction

Cellular networks are present in a number of use cases investigated in the ALTO WG and it is useful to specify a format for Cellular addresses. In these cases, Endpoints, PIDs and entities may be cells. In order to specify services such as Network Maps, Cost Maps, Endpoint property or Property Maps, it is necessary to order to specify an ALTO format for Cell addresses.

For the sake of efficiency, a preferred option is to use the cell identifier format as specified by 3GPP [TS 36.300] and called ECGI, as already proposed in [draft-rauschenbach-alto-wireless-access-00] and in other discussions. ECGI stands for E-UTRAN Cell Global Identifier and is used in Public Land Mobile Networks based on E-UTRAN, see [TS 36.331].

The purpose of this document is to be completed by the ALTO WG and in particular:

- Amend and finalize the specification for the ALTO Cell identifier format proposed in the present version,
- define a placeholder for this specification and identify impacted documents.

## 2. Relevant ALTO services and documents

Particular services and drafts where an ALTO address type for cellular networks is needed include:

- Endpoint property service: extended to allow endpoints to be cells on which properties can be requested,
- (Filtered) Cost Map Service: where PIDs can be cells within and among which cost values can be requested, see also[draft-randriamasy-alto-cost-context-01],
- "Mobility Network Models in ALTO" defined in [draft-bertz-alto-mobilitynets] propose to identify network points of attachment (PoA) such as cells to PIDs.
- "ALTO Performance Cost Metrics": being defined in [draft-ietf-alto-performance-metrics-01], they will be extended to performance costs in cellular networks,
- "Extensible Property Maps for the ALTO Protocol", being defined in [draft-roome-alto-unified-props-new-00] will cover entities that may be cells which are identified by their addresses. In this document a domain identifier for cells will need to be accordingly defined, and the entity domain identifier "ecgi" is proposed.

## 3. Proposed format for ALTO cell identifiers

The prenent draft proposed the following specification

### 3.1. ALTO address type for cellular networks

ECGI -- E-UTRAN Cell Global Identifier

### 3.2. Endpoint address canonical string format

'ecgi:' MCC '.' MNC ':' ECI

Where:

- o MCC: Mobile Country Code, as assigned by ITU. A 3 digits decimal number without leading zeros.
- o MNC: Mobile Network Code, as assigned by National Authority. A 2-3 digits decimal number without leading zeros.
- o ECI: E-UTRAN Cell Identifier. A 7 digits lower-case hex number.

Examples:

- o ecgi:311.481:1234abc
- o MNC value 20 stands for Network N1 in France and other networks in other countries
- o MNC value 020 stands for Network N2 in Argentina and other networks in other countries

### 3.3. ALTO Cell Id formats

Three formats are proposed:

- o 'ecgi:' MCC
- o 'ecgi': MCC '.' MNC
- o 'ecgi:' MCC '.' MNC ':' ECI-MASK '/' MASK-LEN

where:

MASK-LEN is a decimal number.

ECI-MASK is a string of lower-case hex digits, of which all but the first MASK-LEN bits are zero.

Prefix ecgi:P-MCC.P-MNC:P-ECI/N matches ecgi:MCC.MNC:ECI iff

MCC == P-MCC, and

MNC == P-MNC, and

ECI has the same number of hex digits as P-ECI, and the first N bits of ECI match those of P-ECI.

### 3.4. Examples

- o ecgi:311
  - \* Matches every cell address with MCC 311.
- o ecgi:311.481
  - \* Matches every cell address with MCC 311 and MNC 481.
- o ecgi:311.481:1234800/18
  - \* Matches every cell address with MCC 311 , MNC 481, and a 7-digit ECI that starts with the 18 bits 0x12348. Thus it matches ecgi:311.481:1234abc and ecgi:311.481:123480, but does not match ecgi:311.481:12348d.

#### 4. IANA Considerations

This document currently makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

#### 5. Security Considerations

TBC

#### 6. Acknowledgements

Great thanks to Wendy Roome who initiated this document.

#### 7. References

##### 7.1. 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, <<http://www.rfc-editor.org/info/rfc2119>>.

##### 7.2. Informative References

[draft-roome-alto-unified-props-new-00]  
Roome, W. and Y. Yang, "Extensible Property Maps for the ALTO Protocol", March 2017.

#### Appendix A. An Appendix

##### Author's Address

Sabine Randriamasy  
Nokia-Bell-labs  
Route de Villejust  
Nozay 91460  
FRANCE

Email: [Sabine.Randriamasy@nokia-bell-labs.com](mailto:Sabine.Randriamasy@nokia-bell-labs.com)

ALTO  
Internet-Draft  
Intended status: Standards Track  
Expires: January 4, 2018

S. Randriamasy  
Nokia Bell Labs  
July 3, 2017

ALTO Contextual Cost Values  
draft-randriamasy-alto-cost-context-02

Abstract

The Application-Layer Traffic Optimization (ALTO) Service has defined network and cost maps to provide basic network information, where the cost maps allow only one JSON value for a requested metric.

This document introduces several protocol extensions to allow ALTO clients to support use cases such as context based connection selection in cellular networks and calendaring for unattended data. This document refers to other extension proposals posted in the ALTO WG that can support the present use cases as well. Likewise, some of the proposed extensions may serve other ALTO use cases.

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 4, 2018.

## Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Use cases . . . . .	4
2.1. Use Case 1: conditional RF costs in cellular networks . .	4
2.2. Use case 2: access-aware endpoint selection . . . . .	5
3. Required ALTO extensions . . . . .	6
4. Design options and examples . . . . .	7
4.1. Overview of context features . . . . .	7
4.1.1. Applicable ALTO services . . . . .	8
4.2. Example IRD . . . . .	8
4.3. Use case 1: Example scenario for the FCM Service . . . .	10
4.4. Design option: Network Map with cells as PIDs . . . . .	11
4.4.1. Example: FCM Request for contextual 'ARFcost' . . . .	11
4.4.2. Example: FCM Response for contextual ARFcost . . . .	12
4.5. Use case 2: example ALTO transactions for the ECS . . . .	13
4.5.1. Use case 2: example with logical context parameter combinations . . . . .	13
5. Deployment case: local ALTO Server cascaded with global ALTO Server . . . . .	16
5.1. Cascaded ALTO Servers with one network map each . . . . .	16
6. IANA Considerations . . . . .	17
7. Security Considerations . . . . .	17
8. Acknowledgements . . . . .	17
9. References . . . . .	17
9.1. Normative References . . . . .	17
9.2. Informative References . . . . .	17
Appendix A. An Appendix . . . . .	18
Author's Address . . . . .	18

## 1. Introduction

The IETF ALTO protocol specified in [RFC7285] provides guidance to over the top applications which have to select one or several hosts or endpoints from a set of candidates that are able to provide a desired data resource, or which need some provider-centric insight on the cost of application paths to these endhosts. The ALTO Service has defined network and cost maps to provide basic network information, where the cost maps allow only one JSON value for a requested metric.

This draft brings a use case where providing different values for the same cost metric can help in optimizing the application path selection. Typically, when an end host can connect to the network via multiple technologies or access points, the path performance for a metric may be accordingly impacted.

The present draft proposes to extend the cost information specified in [RFC7285] by providing, for the same cost metric, several possible cost values. Which value to provide depends on qualitative criteria as opposed to quantitative criteria such as time. The purpose is to allow a finer grained decision on which application endpoint or sub network to access.

Previous ALTO WG discussions have suggested introducing "the ability to "name" cost maps so that a single Information Resource Directory can link multiple cost maps with the same cost type to a single network map." The goal was to provide, for a given cost metric, multiple cost values depending on qualitative conditions named "circumstance", where a circumstance reflects a given policy.

For applications such as video download or streaming, a user equipment (UE) may use [RFC7285] to choose the best possible application resource location.

Currently, the insight of ALTO information on the path between a UE and a connection node (or say Endpoint) does not provide details below IP hops. However the major QoE challenges of wireless network users arise in the access network, that is, in the first hop between the UE and its one or more serving packet data network gateway (PGW). The path of a UE to its serving PGW(s) impacts the path to the content and thus the related QoE. Therefore, it is necessary to inform the UE, which could take the appropriate decisions w.r.t. the utilized access path. The access technology in current ALTO documents is accounted at the content location (last hop) side by distinguishing whether the requested content is located in a fixed or a wireless access network, as described in [draft-ietf-alto-deployment]



This document introduces several protocol extensions to allow ALTO clients to support use cases such as context-based connection selection in cellular networks and calendaring for unattended data. This document refers to other extension proposals posted in the ALTO WG that can support the present use cases as well. Likewise, some of the proposed extensions may serve other ALTO use cases.

## 2. Use cases

This section presents motivating use cases for contextual ALTO Costs with a focus on conditional RF costs in cellular networks. In these 2 use cases, a terminal UE is located in a LTE network and associated to a "local" ALTO Server(LAOS) that covers this access network, say up to the Packet Data Network (PDN) Gateway PGW and can itself connect to another ALTO Server having a more global view covering up to the whole ISP network. Such a deployment is proposed in section Section 5 of this draft.

### 2.1. Use Case 1: conditional RF costs in cellular networks

Let's assume a terminal UE located in a cellular network. An ALTO Client (LAOC) associated to the UE queries the local ALTO Server in order to know via which cell it should connect to the network. So in a first place, LAOC will query the connection cost associated to cells C1,... CK.

The present example assumes that the connection cost conveyed by the ALTO Server is a unitless value abstracting a non-real-time metric reflecting traffic conditions, aggregated over time and space. This metric aims at providing guidance to applications. It may integrate abstractions by the network provider, of actual costs impacted by other values such as congestion or available bandwidth are assumed to be not easily available to UEs or applications otherwise. The ALTO Server does not aim at reporting accurate radio conditions that indeed vary in time and space.

Let us call this metric: "ARF cost", standing for Abstracted RF cost.

Our example however includes 2 additional considerations:

- the ARF cost to a cell may be impacted by its load,
- a UE usually transmits a fair amount of "unattended data" (UD).

UD is considered in one of the key features for LTE enhancements in Release 13 and defined in 3GPP TS22.101 as follows: "Unattended Data Traffic : Data traffic of which the user is unaware he/she initiated, e.g. based on the screen/keypad lock being activated, length of time

since the UE last received any input from the user, known type of app (e.g. an application monitoring a user's health "mHealth" may need its data never treated as Unattended Data Traffic.)". UD traffic is often delay tolerant and it would be beneficial for the network if the UE can schedule its transmission. To this end, the UE can use an instant UD Indicator (UDI) sent by the LTE network. The UDI, accepted for LTE Release 13 is a single bit sent to the UE indicating whether UD in a cell is allowed (UDA) or not (UDNA). The status change of a UDI from UDA to UDNA is presumably triggered when the cell load exceeds a given threshold  $T(udna)$ . The value of  $T(udna)$  may change across cells and in time but is not provided to UEs. If the UE had an ALTO calendar for either  $T(udna)$  or for the abstracted cell load values, it could appropriately schedule the transmission of its UD, that will have to occur anyway. The UE could combine this calendar with the UDI it receives from the cellular network. The UDI state may change within sub-seconds and impact the data exchange. What is missing in the provided LTE information is:

- knowing whether the UDI threshold relates to downlink or uplink congestion.
- knowing the level of congestion that triggers a change in UDI and how it may evolve in time.

The UE thus can advantageously combine the non-real time ALTO information with the real-time UDI provided by the LTE network. The present draft illustrates how ALTO can fill these gaps with the support of:

- ALTO Cost Calendars,
- the proposed protocol extension providing context-dependent ALTO Cost values.

In this use case: ALTO calendars need to be requested via for the ALTO Filtered Cost Map (FCM) Service, the context parameters impacting the cost values are: "uda" (Unattended Data Allowed), "udna" (Unattended Data Not Allowed), "uplink", "downlink".

## 2.2. Use case 2: access-aware endpoint selection

In a second use case, an end-system called UEP is located in a LTE network and may connect via several access technologies, e.g. Cellular or WiFi. UEP may also benefit from a given Service Level Agreement SLA-m. Other parameters may characterize the UEP generated traffic.

Currently the insight of ALTO information in the path between a UE and a connection node (or say Endpoint) does not provide details below IP hops. However the major QoE challenges of wireless network users arise in the access network, that is, in the first hop between the UE and its one or more associated packet data network gateway (PGW). The path of a UE to its associated PGW(s) impacts the path to the content and thus the related QoE. Therefore, it is necessary to inform the UE, which could take the appropriate decisions w.r.t. the utilized access path. The access technology in current ALTO proposals is accounted at the content location (last hop) side by distinguishing whether the requested content is located in a fixed or a wireless access network, as described in [draft-ietf-alto-deployments] that states: "For ISPs with mobile network and fixed network, the traffic optimizing problems they focus will be optimizing the mobile traffic, except problems on last hop section."

For Mobile Network Operators (MNO) and their users, being connected via e.g. cellular or trusted Wifi can hugely impact the QoE and routing cost. Sometimes a 4G connection is preferable for users than a poor WiFi connection although potentially more expensive. Sometimes, MNOs have spare data resources or offer them for given SLAs. For both parties, access-aware Endpoint selection for Users is thus beneficial. One way to achieve this is that ALTO provides cost values depending on qualitative contextual parameters such as access technology and the access technology and SLA.

### 3. Required ALTO extensions

The aforementioned use cases can be supported with a few simple extensions to the ALTO protocol. A number of them have already been discussed in other WG drafts and use cases. The proposed extensions include:

- Cost value context parameters: a capability to allow exposing several possible context-dependent values for one metric, as proposed in the present document,
- Entities with associated domain and properties for cellular and wireless networks, that could be added to [draft-roome-alto-unified-props],
- Cost metrics for cellular and wireless networks: these features would extend current proposals in the WG, that could be added to [draft-ietf-alto-performance-metrics],
- Extended input for the Filtered Cost Map Service: to allow the input to comprise several (source-array, destination-array) pairs, as it has been proposed in [draft-yang-alto-path-vector].

#### 4. Design options and examples

Similarly to Multi-Cost and Cost Calendar ([draft-ietf-alto-cost-calendar]), this proposal does not introduce new cost modes or new media-types. It ensures backwards compatibility with legacy ALTO Clients, that is: "A legacy ALTO Client must be able to send legacy requests to a Cost Context aware ALTO Server and get legacy responses as specified in RFC7285".

"A Cost Context aware ALTO Server must be able to receive and process requests sent by legacy ALTO Clients, as specified in RFC 7285".

Besides, the proposed extension is designed to be compatible with Multi-Cost ALTO and ALTO Cost Calendars ([draft-ietf-alto-cost-calendar]).

In the present draft version, the IRD indicates the supported context attributes as values encoded in JSON strings. This design simplifies the transactions, as it allows a limited number of context attributes or their combinations, say 1 to 5. Context attributes taking numerous or unpredictable values should be handled as values properties or metrics expressed in constraints.

- A cost context aware ALTO Server MUST provide metric values, as specified in RFC 7285, without any context consideration for all the Cost Types indicated in its "meta".

##### 4.1. Overview of context features

Cost context attributes are strings with values such as "wifi", "cellular", "uda".

Cost context attributes are indicated in the IRD as capabilities of an information resource. They are associated to cost type names.

- A cost context aware ALTO may indicate in its IRD capabilities, whether and how context attributes may be combined in ALTO requests.

- A cost context aware ALTO Server MUST return metric values, without any context consideration, as specified in RFC 7285, if the value for a context attribute or a combination of attributes requested by the client is not available.

- A cost context aware ALTO may indicate a maximum number of context attributes or their combinations authorised in context-aware Client requests.

#### 4.1.1. Applicable ALTO services

Draft [draft-bertz-alto-mobilitynets] proposes to identify network points of attachment (PoA) such as cells to PIDs, as PoAs are endpoint types not currently supported in ALTO. The current proposal is to represent cellular PIDs in an ALTO Network Map with no routes. PID properties as specified in [draft-roome-alto-unified-props] could be used to indicate the type of the PoA, together with other properties. ALTO properties are well suited for almost static attributes such as access type.

To abstract and convey connection properties with frequently changing values such as ARF Cost, load or congestion, the ALTO Filtered Cost Map service can be used. Connection properties may also be conveyed with the Endpoint property service or its extensions defined in [draft-roome-alto-unified-props].

Costs and properties with the extensions proposed in this document may be conveyed with different values depending on the context parameter. The present version of this draft focuses on context parameters associated to costs.

#### 4.2. Example IRD

The purpose of ALTO is to guide the behavior of the end systems or applications without the need for networks to explicitly expose their performance values. In this example, the IRD does not expose the real load percentage of a cell to UE. Instead, it abstracts the cell congestion by a metric called 'ARFcost' represented by a number between 0 and 100, where the optimal value is 0. The values of 'ARFcost' are provided as an ALTO Calendar as specified in [draft-ietf-alto-cost-calendar-00] in shorter time intervals. In addition they differ, depending on the context values "uda" and "udna".

Besides, the IRD provides metric 'routingcost' as a MUST specified in [RFC7285], that may represent a more administrative or monetary access cost.

The IRD could publish the capability of a resource to provide context dependent 'routingcost' values as expressed for resource "filtered-cost-calendar-map".

HTTP/1.1 200 OK  
 Content-Length: [TODO]  
 Content-Type: application/alto-directory+json

```
{
  "meta" : {
    "cost-types": {
      "num-routingcost": {
        "cost-mode" : "numerical",
        "cost-metric" : "routingcost"
      },
      "num-ARFcost": {
        "cost-mode" : "numerical",
        "cost-metric": "ARFcost",
      }
    }
    ... other meta ...
  },
  "resources" : {
    "filtered-cost-calendar-map" : {
      "uri" : "http://alto.local.example.com/costmap/filtered/calendar/cont
ext",
      "media-types" : [ "application/alto-endpointcost+json" ],
      "accepts" : [ "application/alto-endpointcostparams+json" ],
      "capabilities" : {
        "cost-constraints" : true,
        "cost-type-names" : [ "num-routingcost",
                              "num-ARFcost"], // ++NEW
        "calendar-attributes" : [
          { "cost-type-names" : "num-routingcost",
            "time-interval-size" : "1 hour",
            "number-of-intervals" : 24}, // MAY ALSO BE SINGLE VALUE
          { "cost-type-names" : "num-ARFcost", // ++NEW
            "time-interval-size" : "5 minute",
            "number-of-intervals" : 12}
        ],
        "cost-context" : [ // ++NEW
          { "cost-type-names" : "num-ARFcost",
            "context-params" : [ ["uda", "udna"],
                                ["uplink", "downlink"]]
          }
        ], // ++NEW
        "max-context-attributes" : 10,
        "uses": [ "my-default-network-map" ]
      } // end FCM capab
    }
    ... other resources ...
  } // end resources
} // end IRD
```

#### 4.3. Use case 1: Example scenario for the FCM Service

We assume an example scenario where a UE has the choice to connect to 2 cells C1 and C2.

As suggested in [draft-bertz-alto-mobilitynets], we may represent the cellular topology with an ALTO Network Map comprising PIDs representing the cells and named "Cell1", "Cell2", ... "Celln". A format for a cell identifier has been proposed in [draft-rauschenbach-alto-wireless-access] and is not being discussed here.

As a Network Map may cover a large number of cells, the Filtered Cost Map Service can be used to reduce data exchange and get information on a restricted number of cells.

We assume that the ALTO Client in the UE wants to get calendared values for ALTO metric "ARFcost" in order to appropriately schedule its unattended data transmission. The ALTO information resource 'ALTO Calendar' provides an array of time-dependent cost values and is being specified in [draft-ietf-alto-cost-calendar]. In addition, the ALTO Client wants these values for both the "uda" and "udna" context. Last, we assume that the UE needs the Cost values for both the uplink (UE to Cell-k) and downlink (Cell-k to UE) directions. We assume that the UE is located in the PID called "Cell1".

In this scenario, Cell1 is limited by its uplink capacity and Cell2 is limited by its downlink capacity. ALTO can be used to convey the following information:

At time interval T1 of the next Calendar:

- if Cell1 indicates "unattended data allowed" the downlink ARF cost is 20, and the uplink ARF cost is 70
- if Cell1 indicates "unattended data NOT allowed", the downlink ARF cost is 20, and the uplink ARF cost is 90
- if Cell2 indicates "unattended data allowed" the downlink ARF cost is 70, and the uplink ARF cost is 20
- if Cell2 indicates "unattended data NOT allowed", the downlink ARF cost is 90, in the uplink ARF cost is 20.

The ALTO Calendar provides values for the other 11 time intervals Ti.

#### 4.4. Design option: Network Map with cells as PIDs

In this design, the cellular topology is represented with an ALTO Network Map comprising PIDs named "Cell1", "Cell2", ... "Celln". The UE is located in one of these PIDs. A Cost Map is associated to this Network Map and conveys metrics indicated in the IRD. The Cost Map can be to convey connection costs between firstly the UE to its serving cell (that is the PID to itself) and secondly the UE and neighboring cells (that is the PID to another one) and last, for both uplink and downlink directions.

The ALTO Server can regularly update the Cost Map and send filtered information to the ALTO Client. The proposed IRD design announces additional context attributes "uplink", "downlink". In this case and other potential cases, the context parameters need to be arranged w.r.t. their possible combinations (to be further specified in the IRD). For example, the IRD may announce that costs are provided for contexts "uda" and "udna" and this in both contexts "uplink" and "downlink". Or that costs are provided for contexts "uplink" and "downlink" and this in both contexts "udna" and "uda". In such a case, the IRD capability member may list the possible combinations in the capabilities as follows:

```
"cost-context" : [ // ++NEW
  { "cost-type-names" : "num-ARFcost",
    "context-params" : [{"uda", "uplink"},
                        [{"uda", "downlink"},
                         {"udna", "uplink"},
                         {"udna", "downlink"}]} // ++NEW
  }
]
```

This arrangement indicates that for the metric named "num-ARFcost", the ALTO Server can provide 4 different values: v1 for ["uda" AND "uplink"], ... v4 for ["udna" AND "downlink"].

Further versions of this draft will specify more elaborated logical combinations of context attributes, to moderate the length of the ALTO request and support use cases as described in section 4.5.1.

##### 4.4.1. Example: FCM Request for contextual 'ARFcost'

The ALTO Client can specify the desired cost value context parameters in the request input. In particular, it can select one or more combinations indicated in the IRD. Its input parameter "context-params" is an array of all the desired combinations. In this



example, the ALTO Client wants to know the ALTO connection costs within Cell1 and Cell2. For each cell, the Client wants the 4 values, corresponding to all the combinations indicated below.

```
POST /costmap/filtered/calendar/context HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,application/alto-error+json
Content-Type: application/alto-costmapfilter+json
Content-Length: ###

{
  "cost-type" : { "cost-mode": "numerical", "cost-metric": "ARFcost"},
  "calendared" : true,
  "context-params" : [{"uda", "uplink"}, // ++NEW
                      ["uda", "downlink"],
                      ["udna", "uplink"],
                      ["udna", "downlink"]],
  "pids" : [
    { "srcs" : [ "Cell1"], "dsts" : [ "Cell1"] },
    { "srcs" : [ "Cell2"], "dsts" : [ "Cell2"] }
  ]
}
```

#### 4.4.2. Example: FCM Response for contextual ARFcost

The ALTO response provides, for each requested ("src", "dest") pair, a calendar of 12 JSON values, where each is an array of cost values arranged as specified in the "meta" of the ALTO response.

```

HTTP/1.1 200 OK
Content-Type: application/alto-costmap+json
Content-Length: ###

{
  "meta" : {
    "dependent-vtags" : [
      { "resource-id": "my-default-network-map",
        "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
      }
    ],
    "cost-type" : { "cost-mode": "numerical", "cost-metric": "ARFcost" },
    "calendar-response-attributes" :
      { "calendar-start-time" : Tue, 1 Sept 2016 13:00:00 GMT,
        "time-interval-size" : "5 minute",
        "numb-intervals" : 12 },
    "context-params" : [ [ "uda", "uplink" ], // ++NEW
                        [ "uda", "downlink" ],
                        [ "udna", "uplink" ],
                        [ "udna", "downlink" ] ]
  } // end meta
  "cost-map" : {
    "Cell1": { "Cell1": [[70, 20, 90, 20], ... , [50, 20, 70, 20]],
    "Cell2": { "Cell2": [[20, 70, 20, 90], ... , [20, 50, 20, 70]]
  }
}

```

#### 4.5. Use case 2: example ALTO transactions for the ECS

In this use case, the UE requests the ECS to a local ALTO server for the routingcost to the PGW and wants the metric values varying w.r.t. the "access-type" and "SLA-id". Note that the "context" related design feature can be easily transposed for the Cost Map Service.

##### 4.5.1. Use case 2: example with logical context parameter combinations

This section proposes a design, allowing a Client to arrange input context parameters in logical combinations. The purpose is to show how such combinations of context parameters avoids specifying as many metrics and moderates the amount of exchanged data.

In this example the ALTO Server indicates in its IRD that it can provide endpoint cost maps for the example metrics "routingcost" and "bandwidthscore". Values for metric "routingcost" are provided w.r.t. 2 types of context parameters. The ALTO Client may query values for metric "routingcost" for either of these types of parameters or both or none.

For each type, the parameters are listed in an array. We have 2 arrays:

- ["cell", "wifi", "lan"]
- ["SLA-1", "SLA-2", "SLA-3"]

This indicates that in each array, the client can pick one or more parameters and combine them with one or more parameters in the second array. The ALTO Server will provide costs w.r.t. the AND combination across and within arrays.

In the present example, if the Client requests cost values for the combination:

```
[["cell", "wifi"], ["SLA-3"]]
```

The server will provide 2 values: one for ("cell" AND "SLA-3") and the second one for ("wifi" and "SLA-3").

#### 4.5.1.1. Example IRD with logical context parameter combinations

The IRD below specifies the possibility to combine parameters from the two arrays of the example above.

```
"resources" : {
  "filtered-cost-calendar-map" : {
    "uri" : "http://alto.local.example.com/endpointcostmap/lookup/context
",
    "media-types" : [ "application/alto-endpointcost+json" ],
    "accepts" : [ "application/alto-endpointcostparams+json" ],
    "capabilities" : {
      "cost-constraints" : true,
      "cost-type-names" : [ "num-routingcost",
                           "num-bandwidthscore" ],
      "cost-context" : [ // ++NEW
        { "cost-type-names" : "num-routingcost",
          "context-params" : [ ["cell", "wifi", "lan"],
                              ["SLA-1", "SLA-2", "SLA-3"] ]
        }
      ]
    } // end ECM capab
    ... other resources ...
  } // end resources
```

#### 4.5.1.2. Use case 2: example ECS request with logical context parameter combinations

The ALTO Client queries the ECS between 2 endpoints for the following combinations: ("cell" AND "SLA-3") and ("wifi" and "SLA-3") and thus arranges its input context parameters as follows:

```
POST /endpointcost/lookup/context HTTP/1.1
Host: alto.local.example.com
Content-Length: [TODO]
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

{
  "cost-type" : {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
  "context-params" : [["cell", "wifi"], ["SLA-3"]],
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv6:2000::1:2345:6789:abcd"
    ]
  }
}
```

#### 4.5.1.3. Use case 2: example ECS response with logical context parameter combinations

Following the ALTO Client request of the above example, The ALTO Server provides a response as follows:

```
HTTP/1.1 200 OK
Content-Length: [TODO]
Content-Type: application/alto-endpointcost+json

{
  "meta" : {
    "cost-type" : {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
    "context-params" : [{"cell", "wifi"}, ["SLA-3"]]
  } // end meta

  "endpoint-cost-map" : {
    "ipv4:192.0.2.2": {
      "ipv4:192.0.2.89" : [10, 4],
      "ipv6:2000::1:2345:6789:abcd" : [4, 6]
    }
  }
}
```

## 5. Deployment case: local ALTO Server cascaded with global ALTO Server

To maintain scalability, the ALTO coverage network zone can be decomposed in one "local" ALTO Server part covering a restricted local network zone, for instance within the first IP hop range and another "global" part covering the rest of the ISP network, similarly to what is proposed in [draft-ietf-alto-deployments]. The local ALTO server may include the guidance given by the ISP ALTO server and compose it with the "global" guidance in its replies to its ALTO clients. Recent ALTO WG discussions open the possibility for one IRD to indicate multiple network maps having different levels of detail.

### 5.1. Cascaded ALTO Servers with one network map each

In the "cascaded" use case, the ALTO Service is preferably distributed among two ALTO Servers as follows:

The ALTO Client serving the UE is referred to as the LAOC and can be located either in the UE or in the network.

#### 1. A Local ALTO Server (LAOS)

- \* Hosts the information on the local EPS network, covering the paths between e.g. the UEs and the cells or the PGWs,
- \* Hosts an ALTO Client that sends an ALTO request to a "global" ALTO Server, covering the zone beyond the PGW. It can possibly get the global Server updates using the extensions specified in [draft-ietf-alto-incr-update-sse].

- \* receives the ALTO request issued by the ALTO Client associated to the UE.

2. a "core" ALTO Server covers the whole ISP network view, as it would if the "local ALTO Service" is not available or deactivated.

## 6. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

## 7. Security Considerations

## 8. Acknowledgements

Many thanks to Dawn Chan, Li Geng, Xin Wan, Yichen Qian for their feedback on this draft.

## 9. References

### 9.1. 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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC7285] Alimi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.

### 9.2. Informative References

- [draft-bertz-alto-mobilitynets] Bertz, L., "Mobility Network Models in ALTO", October 2015.
- [draft-ietf-alto-cost-calendar] Randriamasy, S., Yang, Y., Wu, Q., Deng, L., and N. Schwan, "ALTO Cost Calendar", February 2017.
- [draft-ietf-alto-deployment] Stiernerling, M., Kiesel, S., Scharf, M., Seidel, H., and S. Previdi, "draft-ietf-alto-deployments-16", July 2016.

[draft-ietf-alto-incr-update-sse]

Roome, W. and Y. Yang, "ALTO Incremental Updates Using Server-Sent Events (SSE)", Septembre 2016.

[draft-ietf-alto-performance-metrics]

Wu, Q., Yang, Y., Lee, Y., Dhody, D., and S. Randriamasy, "ALTO Performance Cost Metrics", March 2017.

[draft-rauschenbach-alto-wireless-access]

Rauschenbach, U., "ALTO in wireless access networks", October 2014.

[draft-roome-alto-unified-props]

Roome, W., "Extensible Property Maps for the ALTO Protocol", July 2016.

[draft-yang-alto-path-vector]

Bernstein, G., Gao, K., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Extension: Path Vector Cost Mode", July 2016.

#### Appendix A. An Appendix

##### Author's Address

Sabine Randriamasy  
Nokia Bell Labs  
Route de Villejust  
Nozay 91460  
FRANCE

Email: [sabine.randriamasy@nokia-bell-labs.com](mailto:sabine.randriamasy@nokia-bell-labs.com)

ALTO WG  
Internet-Draft  
Intended status: Standards Track  
Expires: January 4, 2018

W. Roome  
Nokia Bell Labs  
R. Yang  
Yale University  
July 3, 2017

Extensible Property Maps for the ALTO Protocol  
draft-roome-alto-unified-props-new-01

Abstract

This document extends the Application-Layer Traffic Optimization (ALTO) Protocol [RFC7285] by generalizing the concept of "endpoint properties" to other entity domains, and by presenting those properties as maps, similar to the network and cost maps in ALTO.

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 4, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of



publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Definitions and Concepts . . . . .	4
2.1. Entity . . . . .	4
2.2. Domain . . . . .	4
2.3. Entity Address . . . . .	4
2.4. Domain Name . . . . .	5
2.5. Property Name . . . . .	5
2.6. Property Value . . . . .	6
2.7. Hierarchy and Inheritance . . . . .	6
2.8. Relationship to Network Maps . . . . .	6
3. Entity Domains . . . . .	7
3.1. Internet Address Domains . . . . .	7
3.1.1. IPv4 Domain . . . . .	7
3.1.2. IPv6 Domain . . . . .	8
3.1.3. Hierarchy and Inheritance of ipv4/ipv6 Domains . . . . .	8
3.1.4. Relationship to Network Maps . . . . .	9
3.2. PID Domain . . . . .	10
3.2.1. Domain Name . . . . .	10
3.2.2. Domain-Specific Entity Addresses . . . . .	10
3.2.3. Hierarchy and Inheritance . . . . .	10
3.2.4. Relationship To Internet Addresses Domains . . . . .	10
3.3. Internet Address Properties vs. PID Properties . . . . .	10
3.4. ANE Domain . . . . .	11
3.4.1. Domain Name . . . . .	11
3.4.2. Domain-Specific Entity Addresses . . . . .	11
3.4.3. Hierarchy and Inheritance . . . . .	11
3.4.4. Relationship to Cost Map . . . . .	11
4. Property Map Resource . . . . .	11
4.1. Media Type . . . . .	11
4.2. HTTP Method . . . . .	12
4.3. Accept Input Parameters . . . . .	12
4.4. Capabilities . . . . .	12
4.5. Uses . . . . .	12
4.6. Response . . . . .	12
5. Filtered Property Map Resource . . . . .	13
5.1. Media Type . . . . .	14
5.2. HTTP Method . . . . .	14
5.3. Accept Input Parameters . . . . .	14
5.4. Capabilities . . . . .	14

5.5. Uses . . . . .	15
5.6. Response . . . . .	15
6. Impact on Legacy ALTO Servers and ALTO Clients . . . . .	15
6.1. Impact on Endpoint Property Service . . . . .	15
6.2. Impact on Resource-Specific Properties . . . . .	15
6.3. Impact on the "pid" Property . . . . .	16
6.4. Impact on Other Properties . . . . .	16
7. Examples . . . . .	16
7.1. Network Map . . . . .	16
7.2. Property Definitions . . . . .	17
7.3. Information Resource Directory (IRD) . . . . .	17
7.4. Property Map Example . . . . .	19
7.5. Filtered Property Map Example #1 . . . . .	19
7.6. Filtered Property Map Example #2 . . . . .	20
7.7. Filtered Property Map Example #3 . . . . .	21
7.8. Filtered Property Map Example #4 . . . . .	22
8. Security Considerations . . . . .	23
9. IANA Considerations . . . . .	24
9.1. application/alto-* Media Types . . . . .	24
9.2. ALTO Entity Domain Registry . . . . .	25
9.3. ALTO Endpoint Property Type Registry . . . . .	26
10. References . . . . .	26
Authors' Addresses . . . . .	27

## 1. Introduction

The ALTO protocol [RFC7285] introduced the concept of "properties" attached to "endpoint addresses," and defined the Endpoint Property Service (EPS) to allow clients to retrieve those properties. While useful, the EPS, as defined in RFC7285, has at least two limitations.

First, it only allows properties to be associated with a particular domain of entities, namely individual IP addresses. It is reasonable to think that collections of endpoints, as defined by CIDRs ([RFC4632]) or PIDs, may also have properties. Furthermore, recent proposal ([I-D.ietf-alto-path-vector]) have suggested new classes of entities (ANE) with properties. The EPS cannot be extended to new entity domains. Instead, new services, with new request and response messages, would have to be defined for each new entity domain.

Second, the EPS is only defined as a POST-mode service. Clients must request the properties for an explicit set of addresses. By contrast, [RFC7285] defines a GET-mode Cost Map resource which returns all available costs, so a client can get the full set of costs once, and then lookup costs without querying the ALTO server. RFC7285 does not define an equivalent service for endpoint properties. And it is unlikely a property will be defined for every possible address. It is very likely that properties will only be

defined for a subset of addresses, and that subset would be small enough to enumerate. This is particularly true if blocks of addresses with a common prefix (e.g., a CIDR) have the same value for a property. Furthermore, entities in other domains may very well be enumerable.

This document proposes a new approach to retrieve ALTO properties. Specifically, it defines two new resource types, namely Property Maps (see Section 4) and Filtered Property Maps (see Section 5). The former are GET-mode resources which return the property values for all entities in a domain, and are analogous to the ALTO's Network Maps and Cost Maps. The latter are POST-mode resources which return the values for a set of properties and entities requested by the client, and are analogous to ALTO's Filtered Network Maps and Filtered Cost Maps.

Entity domains and property names are extensible, so that new domains can be defined without revising the messages defined in this document, in the same way that new cost metrics and new endpoint properties can be defined without revising the messages defined by the ALTO protocol.

This proposal would subsume the Endpoint Property Service defined in RFC7285, although that service may be retained for legacy clients (see Section 6).

## 2. Definitions and Concepts

### 2.1. Entity

An entity is an object with a (possibly empty) set of properties. Every entity is in a domain, such as the IPv4 and IPv6 domains, and has a unique address.

### 2.2. Domain

A domain is a family of entities. Two examples are the Internet address and PID domain (see Section 3.1 and Section 3.2) that this document will define. An additional example is the proposed domain of Abstract Network Elements associated with topology and routing, as suggested by [I-D.ietf-alto-path-vector].

### 2.3. Entity Address

Each entity has a unique address of the format:

domain-name : domain-specific-entity-address

Examples from the IP domain include individual addresses such as "ipv4:192.0.2.14" and "ipv6:2001:db8::12", as well as address blocks such as "ipv4:192.0.2.0/26" and "ipv6:2001:db8::1/48".

The type `EntityAddr` denotes a JSON string with an entity address in this format.

The format of the second part of an entity address depends on the domain, and **MUST** be specified when registering a new domain. Addresses **MAY** be hierarchical, and properties **MAY** be inherited based on that hierarchy. Again, the rules defining any hierarchy or inheritance **MUST** be defined when the domain is registered.

Note that entity addresses **MAY** have different textual representations, for a given domain. For example, the strings "ipv6:2001:db8::1" and "ipv6:2001:db8:0:0:0:0:1" refer to the same entity.

#### 2.4. Domain Name

Each domain has a unique name. A domain name **MUST** be no more than 32 characters, and **MUST NOT** contain characters other than US-ASCII alphanumeric characters (U+0030-U+0039, U+0041-U+005A, and U+0061-U+007A), hyphen ('-', U+002D), and low line ('\_', U+005F). For example, the names "ipv4" and "ipv6" identify objects in the Internet address domain (Section 3.1).

The type `DomainName` denotes a JSON string with a domain name in this format.

Domain names **MUST** be registered with the IANA, and the format of the entity addresses in that domain, as well as any hierarchical or inheritance rules for those entities, **MUST** be specified at the same time.

#### 2.5. Property Name

The space of property names associated with entities defined by this document is the same as, and is shared with, the endpoint property names defined by [RFC7285]. Thus entity property names are as defined in Section 10.8.2 of that document, and **MUST** be registered with the "ALTO Endpoint Property Type Registry" defined in Section 14.3 of that document.

The type `PropertyName` denotes a JSON string with a property name in this format.

A main design decision is that property names are defined in a single namespace, not specific to a domain, although some properties MAY only be applicable for particular domains. This design decision is to enforce a design that similar properties are named similarly.

The interpretation of the value of a property, however, MAY depend on the domain. For example, suppose the "geo-location" property is defined as the coordinates of a point, encoded as "latitude longitude [altitude]." When applied to an entity that represents a specific host computer, such as an Internet address, the property defines the host's location. When applied to an entity that represents a set of computers, such as a CIDR, the property would be the location of the center of that set. If it is necessary to represent the bounding box of a set of hosts, another property, such as "geo-region", SHOULD be defined.

## 2.6. Property Value

The property value MAY BE defined or undefined. If it is defined, it SHOULD BE a JSONString or a JSON "null" value. Otherwise, a protocol implementation SHOULD fail to parse the property value, unless the implementation is using an extension to this document that indicates when and how property values of other data types are signaled.

## 2.7. Hierarchy and Inheritance

Entities in a given domain MAY form hierarchy based on entity address. Each domain MAY define its own hierarchy and inheritance semantics. Given a property, the semantics MUST NOT allow an entity with a defined property value to inherit the property value of another entity. Given a property, the semantics MAY allow an entity with an undefined property value to inherit the property value of another entity. An entity may not be able to inherit a property value if no other entities satisfy the inheritance conditions defined by the semantics. The hierarchy and inheritance semantics SHOULD be defined carefully to avoid multiple inheritance.

## 2.8. Relationship to Network Maps

[RFC7285] recognizes that some properties MAY be specific to an ALTO resource, such as a network map. Accordingly [RFC7285] defines the concept of "resource-specific endpoint properties" (Section 10.8.1), and indicates that dependency by prefixing the property name with the ID of the resource on which it depends. That document defines one resource-specific property, namely the "pid" property, whose value is the name of the PID containing that endpoint in the associated network map.

This document takes a different approach. Instead of defining the dependency by qualifying the property name, this document attaches the dependency to the property map as a whole. Thus all properties in a given property map depend on the same resource. Furthermore, entity addresses MAY depend on a network map (for example, the Abstract Network Elements suggested by [I-D.ietf-alto-path-vector]). Associating the dependency with the property map handles any entity address dependencies as well.

The "uses" field in an IRD entry defines the dependencies of a property map resource, and the "dependent-vtags" field in a property map response defines the dependencies of that map. These fields are defined in Sections 9.1.5 and 11.1 of [RFC7285], respectively.

This is similar to how RFC7285 handles dependencies between cost maps and network maps. Recall that cost maps present the costs between PIDs, and PID names depend on a network map. If an ALTO server provides the "routingcost" metric for the network maps "net1" and "net2", then the server defines two separate cost maps, one for "net1" and the other for "net2".

According to [RFC7285], a legacy ALTO server with two network maps, with resource IDs "net1" and "net2", could offer a single Endpoint Property Service for the two properties "net1.pid" and "net2.pid". An ALTO server which supports the extensions defined in this document, would, instead, offer two different Property Maps for the "pid" property, one depending on "net1", the other on "net2".

### 3. Entity Domains

This document defines the following entity domains. For the definition of each domain, it includes the following template: domain name, domain-specific addresses, and hierarchy and inheritance semantics.

#### 3.1. Internet Address Domains

The document defines two domains (IPv4 and IPv6) for Internet addresses. Both domains include individual addresses and blocks of addresses.

##### 3.1.1. IPv4 Domain

###### 3.1.1.1. Domain Name

ipv4

### 3.1.1.2. Domain-Specific Entity Addresses

Individual addresses are strings as specified by the IPv4Addresses rule of Section 3.2.2 of [RFC3986]. Blocks of addresses are prefix-match strings as specified in Section 3.1 of [RFC4632]. For the purpose of defining properties, an individual Internet address and the corresponding full-length prefix are considered aliases for the same entity. Thus "ipv4:192.0.2.0" and "ipv4:192.0.2.0/32" are equivalent.

### 3.1.2. IPv6 Domain

#### 3.1.2.1. Domain Name

ipv6

### 3.1.2.2. Domain-Specific Entity Addresses

Individual addresses are strings as specified by Section 4 of [RFC5952]. Blocks of addresses are prefix-match strings as specified in Section 7 of [RFC5952]. For the purpose of defining properties, an individual Internet address and the corresponding 128-bit prefix are considered aliases for the same entity. That is, "ipv6:2001:db8::1" and "ipv6:2001:db8::1/128" are equivalent, and have the same set of properties.

### 3.1.3. Hierarchy and Inheritance of ipv4/ipv6 Domains

Both domains allow property values to be inherited. Specifically, if a property P is not defined for a specific Internet address IP, but P is defined for some block C which prefix-matches IP, then the address IP inherits the value of P defined for block C. If more than one such block defines a value for P, IP inherits the value of P in the block with the longest prefix. It is important to notice that this longest prefix rule will ensure no multiple inheritance, and hence no ambiguity.

Address blocks can also inherit properties: if property P is not defined for a block C, but is defined for some block C' prefix-matches C, and C' has a shorter mask than C, then block C inherits the property from C'. If there are several such blocks C', C inherits from the block with the longest prefix.

```
ipv4:192.0.2.0/26: P=v1
ipv4:192.0.2.0/28: P=v2
ipv4:192.0.2.0/30: P=v3
ipv4:192.0.2.0:    P=v4
```

Figure 1: Defined Property Values.

Then the following entities have the indicated values:

```
ipv4:192.0.2.0:    P=v4
ipv4:192.0.2.1:    P=v3
ipv4:192.0.2.16:   P=v1
ipv4:192.0.2.32:   P=v1
ipv4:192.0.2.64:   (not defined)
ipv4:192.0.2.0/32: P=v4
ipv4:192.0.2.0/31: P=v3
ipv4:192.0.2.0/29: P=v2
ipv4:192.0.2.0/27: P=v1
ipv4:192.0.2.0/25: (not defined)
```

Figure 2: Inherited Property Values.

An ALTO Server MAY explicitly define a property as not having a value for a particular entity. That is, a server MAY say that a property is "defined to have no value", as opposed to the property being "undefined". If that entity would inherit a value for that property, then the ALTO server MUST return a "null" value for that property, and an ALTO client MUST recognize a "null" value means "do not apply the inheritance rules for this property." If the entity would not inherit a value, the ALTO server MAY return "null" or MAY just omit the property. TODO: Discuss more.

If the ALTO Server does not define any properties for an entity, then the server MAY omit that entity from the response.

#### 3.1.4. Relationship to Network Maps

TODO: Need discussion. An Internet address domain MAY be associated with an ALTO network map resource. Logically, there is a map of Internet address entities to property values for each network map defined by the ALTO server, plus an additional property map for Internet address entities which are not associated with a network map. These maps are separate from each other. The prefixes in the property map do not have to correspond to the prefixes defining the network map's PIDs. For example, the property map for a network map MAY assign properties to "ipv4:192.0.2.0/24" even if that prefix is not associated with any PID in the network map.



### 3.2. PID Domain

The PID domain associates property values with the PIDs in a network map. Accordingly, this domain always depends on a network map.

#### 3.2.1. Domain Name

pid

#### 3.2.2. Domain-Specific Entity Addresses

The entity addresses are the PID names of the associated network map.

#### 3.2.3. Hierarchy and Inheritance

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

#### 3.2.4. Relationship To Internet Addresses Domains

The PID domain and the Internet address domains are completely independent; the properties associated with a PID have no relation to the properties associated with the prefixes or endpoint addresses in that PID. An ALTO server MAY choose to assign some or all properties of a PID to the prefixes in that PID.

For example, suppose "PID1" consists of the prefix "ipv4:192.0.2.0/24", and has the property "P" with value "v1". The Internet address entities "ipv4:192.0.2.0" and "ipv4:192.0.2.0/24", in the IPv4 domain MAY have a value for the property "P", and if they do, it is not necessarily "v1".

### 3.3. Internet Address Properties vs. PID Properties

Because the Internet address and PID domains are completely separate, the question may arise as to which domain is best for a property. In general, the Internet address domain is RECOMMENDED for properties that are closely related to the Internet address, or are associated with, and inherited through, blocks of addresses.

The PID domain is RECOMMENDED for properties that arise from the definition of the PID, rather than from the Internet address prefixes in that PID.

For example, because Internet addresses are allocated to service providers by blocks of prefixes, an "ISP" property would be best associated with the Internet address domain. On the other hand, a

property that explains why a PID was formed, or how it relates the a provider's network, would best be associated with the PID domain.

### 3.4. ANE Domain

#### 3.4.1. Domain Name

ane

#### 3.4.2. Domain-Specific Entity Addresses

The entity address of ane domain 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.

#### 3.4.3. Hierarchy and Inheritance

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

#### 3.4.4. Relationship to Cost Map

TBA

## 4. Property Map Resource

A Property Map returns the properties defined for all entities in one or more domains. Note that Property Map Resource is not applicable to ANE domain.

Section 7.4 gives an example of a property map request and its response.

### 4.1. Media Type

The media type of an ALTO Property Map resource is "application/alto-propmap+json".

#### 4.2. HTTP Method

An ALTO Property Map resource is requested using the HTTP GET method.

#### 4.3. Accept Input Parameters

None.

#### 4.4. Capabilities

The capabilities are defined by an object of type `PropertyMapCapabilities`:

```
object {  
  DomainName domain-types<1..*>;  
  PropertyName prop-types<1..*>;  
} PropertyMapCapabilities;
```

where "domain-types" is an array with the domains of the entities in this property map, and "prop-types" is an array with the names of the properties returned for entities in those domains. TODO: discuss semantics and requirements of multiple domains.

#### 4.5. Uses

An array with the resource ID(s) of resource(s) with which the domains in this map are associated. In most cases, this array will have at most one ID, for example, for a network map resource. TODO: discuss semantics and requirements of multiple resources.

#### 4.6. Response

If the domains in this property map depend on other resources, the "dependent-vtags" field in the "meta" field of the response MUST be an array that includes the version tags of those resources. The data component of a Property Map response is named "property-map", which is a JSON object of type `PropertyMapData`, where:

```
object {  
  PropertyMapData property-map;  
} InfoResourceProperties : ResponseEntityBase;  
  
object-map {  
  EntityAddr -> EntityProps;  
} PropertyMapData;  
  
object {  
  PropertyName -> JSONValue;  
} EntityProps;
```

The ResponseEntityBase type is defined in Section 8.4 of [RFC7285].

Specifically, a PropertyMapData object has one member for each entity in the Property Map. The entity's properties are encoded in the corresponding EntityProps object. EntityProps encodes one name/value pair for each property, where the property names are encoded as strings of type PropertyName. A protocol implementation SHOULD assume that the property value is either a JSONString or a JSON "null" value, and fail to parse if it is not, unless the implementation is using an extension to this document that indicates when and how property values of other data types are signaled.

An ALTO Server MAY explicitly define a property as not having a value for a particular entity. That is, a server MAY say that a property is "defined to have no value", as opposed to the property being "undefined". If that entity would inherit a value for that property, then the ALTO server MUST return a "null" value for that property, and an ALTO client MUST recognize a "null" value means "do not apply the inheritance rules for this property." If the entity would not inherit a value, the ALTO server MAY return "null" or MAY just omit the property.

For each entity in the Property Map, the ALTO Server returns the value defined for each of the properties specified in this resource's "capabilities" list. For efficiency, the ALTO Server SHOULD omit property values that are inherited rather than explicitly defined; if a client needs inherited values, the client SHOULD use the domain's inheritance rules to deduce those values.

## 5. Filtered Property Map Resource

A Filtered Property Map returns the values of a set of properties for a set of entities selected by the client.

Section 7.5, Section 7.6 and Section 7.7 give examples of filtered property map requests and responses.

### 5.1. Media Type

The media type of an ALTO Property Map resource is "application/alto-propmap+json".

### 5.2. HTTP Method

An ALTO Filtered Property Map resource is requested using the HTTP POST method.

### 5.3. Accept Input Parameters

The input parameters for a Filtered Property Map request are supplied in the entity body of the POST request. This document specifies the input parameters with a data format indicated by the media type "application/alto-propmapparams+json", which is a JSON object of type ReqFilteredPropertyMap:

```
object {  
  EntityAddr      entities<1..*>;  
  PropertyName    properties<1..*>;  
} ReqFilteredPropertyMap;
```

with fields:

**entities:** List of entity addresses for which the specified properties are to be returned. The ALTO server MUST interpret entries appearing multiple times as if they appeared only once. The domain of each entity MUST be included in the list of domains in this resource's "capabilities" field (Section 5.4).

**properties:** List of properties to be returned for each entity. Each specified property MUST be included in the list of properties in this resource's "capabilities" field (Section 5.4). The ALTO server MUST interpret entries appearing multiple times as if they appeared only once.

Note that the "entities" and "properties" fields MUST have at least one entry each.

### 5.4. Capabilities

The capabilities are defined by an object of type PropertyMapCapabilities, as defined in Section 4.4.

### 5.5. Uses

An array with the resource ID(s) of resource(s) with which the domains in this map are associated. In most cases, this array will have at most one ID, and it will be for a network map resource.

### 5.6. Response

The response is the same as for the property map (Section 4.6), except that it only includes the entities and properties requested by the client.

Also, the Filtered Property Map response MUST include all inherited property values for the specified entities (unlike the Full Property Map, the Filtered Property Map response does not include enough information for the client to calculate the inherited values).

Discussion Needed: sometimes the client can compute some inherited property values. In this case, can the Filter Property Map response only contain the uncomputable inherited property values instead of all of them?

## 6. Impact on Legacy ALTO Servers and ALTO Clients

### 6.1. Impact on Endpoint Property Service

The Property Maps defined in this document provide the same functionality as the Endpoint Property Service (EPS) defined in Section 11.4 of [RFC7285]. Accordingly, it is RECOMMENDED that the EPS be deprecated in favor of Property Maps. However, ALTO servers MAY provide an EPS for the benefit of legacy clients.

### 6.2. Impact on Resource-Specific Properties

Section 10.8 of [RFC7285] defines two categories of endpoint properties: "resource-specific" and "global". Resource-specific property names are prefixed with the ID of the resource they depended upon, while global property names have no such prefix. The property map resources defined in this document do not distinguish between those two types of properties. Instead, if there is a dependency, it is indicated by the "uses" capability of a property map, and is shared by all properties and entity domains in that map. Accordingly, it is RECOMMENDED that resource-specific endpoint properties be deprecated, and no new resource-specific endpoint properties be defined.

### 6.3. Impact on the "pid" Property

Section 7.1.1 of [RFC7285] defines the resource-specific endpoint property "pid", whose value is the name of the PID containing that endpoint. For compatibility with legacy clients, an ALTO server which provides the "pid" property via the Endpoint Property Service MUST use that definition, and that syntax, in the EPS resource.

However, when used with Property Maps, this document amends the definition of the "pid" property as follows.

First, the name of the property is simply "pid"; the name is not prefixed with the resource ID of a network map. The "uses" capability of the property map resource indicates the associated network map. This implies that a property map can only return the "pid" property for one network map; if an ALTO server provides several network maps, it MUST provide a property map resource for each one.

Second, a client MAY request the "pid" property for a block of addresses. An ALTO server determines the value of "pid" for an address block C as follows. Let CS be the set of all address blocks in the network map. If C is in CS, then the value of "pid" is the name of the PID associated with C. Otherwise, find the longest block C' in CS such that C' prefix-matches C, but is shorter than C. If there is such a block C', the value of "pid" is the name of the PID associated with C'. If not, then "pid" has no value for block C.

Note that although an ALTO server MAY provide a GET-mode property map resource which returns the entire map for the "pid" property, there is no need to do so, because that map is simply the inverse of the network map.

### 6.4. Impact on Other Properties

In general, there should be little or no impact on other previously defined properties. The only consideration is that properties can now be defined on blocks of addresses, rather than just individual addresses, which might change the semantics of a property.

## 7. Examples

### 7.1. Network Map

The examples in this section use a very simple default network map:

```

defaultpid:  ipv4:0.0.0.0/0  ipv6:::0/0
pid1:         ipv4:192.0.2.0/25
pid2:         ipv4:192.0.2.0/28  ipv4:192.0.2.16/28

```

Figure 3: Example Network Map

## 7.2. Property Definitions

The examples in this section use four additional properties, "ISP", "ASN", "country" and "state", with the following values:

	ISP	ASN	country	state
ipv4:192.0.2.0/24:	BitsRus	-	us	-
ipv4:192.0.2.0/28:	-	12345	-	NJ
ipv4:192.0.2.16/28:	-	12345	-	CT
ipv4:192.0.2.0:	-	-	-	PA

Figure 4: Example Property Values

## 7.3. Information Resource Directory (IRD)

The following IRD defines the relevant resources of the ALTO server. It provides two Property Map resources, one for the "ISP" and "ASN" properties, and another for the "country" and "state" properties. The server could have provided a Property Map resource for all four properties, but did not, presumably because the organization that runs the ALTO server believes any given client is not interested in all four properties.

The server provides two Filtered Property Maps. The first returns all four properties, and the second just returns the "pid" property for the default network map.

The Filtered Property Maps for the "ISP", "ASN", "country" and "state" properties do not depend on the default network map (it does not have a "uses" capability), because the definitions of those properties do not depend on the default network map. The Filtered Property Map for the "pid" property does have a "uses" capability for the default network map, because that defines the values of the "pid" property.

Note that for legacy clients, the ALTO server provides an Endpoint Property Service for the "pid" property for the default network map.

```

"meta": { ... },
"resources" : {
  "default-network-map" : {
    "uri" : "http://alto.example.com/networkmap",

```



```
    "media-type" : "application/alto-networkmap+json"
  },
  .... property map resources ....
  "country-state-property-map" : {
    "uri" : "http://alto.example.com/propmap/full/inet-cs",
    "media-type" : "application/alto-propmap+json",
    "capabilities" : {
      "domain-types": [ "ipv4", "ipv6" ],
      "prop-types" : [ "country", "state" ]
    }
  },
  "isp-asn-property-map" : {
    "uri" : "http://alto.example.com/propmap/full/inet-ia",
    "media-type" : "application/alto-propmap+json",
    "capabilities" : {
      "domain-types": [ "ipv4", "ipv6" ],
      "prop-types" : [ "ISP", "ASN" ]
    }
  },
  "iacs-property-map" : {
    "uri" : "http://alto.example.com/propmap/lookup/inet-iacs",
    "media-type" : "application/alto-propmap+json",
    "accepts" : "application/alto-propmapparams+json",
    "capabilities" : {
      "domain-types": [ "ipv4", "ipv6" ],
      "prop-types" : [ "ISP", "ASN", "country", "state" ]
    }
  },
  "pid-property-map" : {
    "uri" : "http://alto.example.com/propmap/lookup/pid",
    "media-type" : "application/alto-propmap+json",
    "accepts" : "application/alto-propmapparams+json",
    "uses" : [ "default-network-map" ]
    "capabilities" : {
      "domain-types" : [ "ipv4", "ipv6" ],
      "prop-types" : [ "pid" ]
    }
  },
  "availbw-property-map" : {
    "uri" : "http://alto.example.com/propmap/lookup/availbw",
    "media-type" : "application/alto-propmap+json",
    "accepts" : "application/alto-propmapparams+json",
    "capabilities" : {
      "domain-types" : [ "ane" ],
      "prop-types" : [ "availbw", "delay" ]
    }
  },
  "legacy-pid-property-map" : {
```

```
    "uri" : "http://alto.example.com/legacy/eps-pid",
    "media-type" : "application/alto-endpointprop+json",
    "accepts" : "application/alto-endpointpropparams+json",
    "capabilities" : {
      "prop-types" : [ "default-network-map.pid" ]
    }
  }
}
```

Figure 5: Example IRD

#### 7.4. Property Map Example

The following example uses the properties and IRD defined above to retrieve a property map for entities with the "ISP" and "ASN" properties. Note that the response does not include the entity "ipv4:192.0.2.0", because it does not have a value for either of those properties. Also note that the entities "ipv4:192.0.2.0/28" and "ipv4:192.0.2.16/28" are refinements of "ipv4:192.0.2.0/24", and hence inherit its value for "ISP" property. But because that value is inherited, it is not

```
GET /propmap/full/inet-ia HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
```

```
HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json
```

```
{
  "property-map": {
    "ipv4:192.0.2.0/24": { "ISP": "BitsRus" },
    "ipv4:192.0.2.0/28": { "ASN": "12345" },
    "ipv4:192.0.2.16/28": { "ASN": "12345" }
  }
}
```

#### 7.5. Filtered Property Map Example #1

The following example uses the Filtered Property Map resource to request the "ISP", "ASN" and "state" properties for several IPv4 addresses. Note that the value of "state" for "ipv4:192.0.2.0" is the only explicitly defined property; the other values are all derived by the inheritance rules for Internet address entities.

```
POST /propmap/lookup/inet-iacs HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: ###
Content-Type: application/alto-propmapparams+json
```

```
{
  "entities" : [ "ipv4:192.0.2.0",
                 "ipv4:192.0.2.1",
                 "ipv4:192.0.2.17" ],
  "properties" : [ "ISP", "ASN", "state" ]
}
```

```
HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json
```

```
{
  "property-map": {
    "ipv4:192.0.2.0": {
      "ISP": "BitsRus", "ASN": "12345", "state": "PA"},
    "ipv4:192.0.2.1": {
      "ISP": "BitsRus", "ASN": "12345", "state": "NJ"},
    "ipv4:192.0.2.17": {
      "ISP": "BitsRus", "ASN": "12345", "state": "CT"}
  }
}
```

#### 7.6. Filtered Property Map Example #2

The following example uses the Filtered Property Map resource to request the "ASN", "country" and "state" properties for several IPv4 prefixes. Note that none of the returned property values were explicitly defined; all values are derived by the inheritance rules for Internet address entities.

Also note the "ASN" property has the value "12345" for both the blocks "ipv4:192.0.2.0/28" and "ipv4:192.0.2.16/28", so every address in the block "ipv4:192.0.2.0/27" has that property value. However the block "ipv4:192.0.2.0/27" itself does not have a value for "ASN": address blocks cannot inherit properties from blocks with longer prefixes, even if every such block has the same value.

```
POST /propmap/lookup/inet-iacs HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: ###
Content-Type: application/alto-propmapparams+json
```

```
{
  "entities" : [ "ipv4:192.0.2.0/26",
                 "ipv4:192.0.2.0/27",
                 "ipv4:192.0.2.0/28" ],
  "properties" : [ "ASN", "country", "state" ]
}
```

```
HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json
```

```
{
  "property-map": {
    "ipv4:192.0.2.0/26": { "country": "us" },
    "ipv4:192.0.2.0/27": { "country": "us" },
    "ipv4:192.0.2.0/28": { "ASN": "12345",
                           "country": "us",
                           "state": "NJ" }
  }
}
```

### 7.7. Filtered Property Map Example #3

The following example uses the Filtered Property Map resource to request the "pid" property for several IPv4 addresses and prefixes.

Note that the value of "pid" for the prefix "ipv4:192.0.2.0/26" is "pid1", even though all addresses in that block are in "pid2", because "ipv4:192.0.2.0/25" is the longest prefix in the network map which prefix-matches "ipv4:192.0.2.0/26", and that prefix is in "pid1".

```
POST /propmap/lookup/pid HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: ###
Content-Type: application/alto-propmapparams+json
```

```
{
  "entities" : [
    "ipv4:192.0.2.0",
    "ipv4:192.0.2.16",
    "ipv4:192.0.2.64",
    "ipv4:192.0.2.128",
    "ipv4:192.0.2.0/26",
    "ipv4:192.0.2.0/30" ],
  "properties" : [ "pid" ]
}
```

```
HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json
```

```
{
  "meta" : {
    "dependent-vtags" : [
      { "resource-id": "default-network-map",
        "tag": "7915dc0290c2705481c491a2b4ffbec482b3cf62" }
    ]
  },
  "property-map": {
    "ipv4:192.0.2.0":      { "pid": "pid2" },
    "ipv4:192.0.2.16":    { "pid": "pid2" },
    "ipv4:192.0.2.64":    { "pid": "pid1" },
    "ipv4:192.0.2.128":   { "pid": "defaultpid" },
    "ipv4:192.0.2.0/26":  { "pid": "pid1" },
    "ipv4:192.0.2.0/30":  { "pid": "pid2" }
  }
}
```

#### 7.8. Filtered Property Map Example #4

The following example uses the Filtered Property Map resource to request the "availbw" property for several abstract network elements.

```
POST /propmap/lookup/availbw HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: ###
Content-Type: application/alto-propmapparams+json
```

```
{
  "entities" : [
    "ane:L001",
    "ane:Lae0",
    "ane:L3eb" ],
  "properties" : [ "availbw" ]
}
```

```
HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json
```

```
{
  "property-map": {
    "ane:L001": { "availbw": "55" },
    "ane:Lae0": { "availbw": "70" },
    "ane:L3eb": { "availbw": "40" }
  }
}
```

## 8. Security Considerations

As discussed in Section 15 of [RFC7285], properties MAY have sensitive customer-specific information. If this is the case, an ALTO Server MAY limit access to those properties by providing several different Property Maps. For non-sensitive properties, the ALTO Server would provide a URI which accepts requests from any client. Sensitive properties, on the other hand, would only be available via a secure URI which would require client authentication.

Also, while technically this document does not introduce any security risks not inherent in the Endpoint Property Service defined by [RFC7285], the GET-mode property map resource defined in this document does make it easier for a client to download large numbers of property values. Accordingly, an ALTO Server SHOULD limit GET-mode Property Maps to properties which do not contain sensitive data.

## 9. IANA Considerations

This document defines additional application/alto-\* media types, and extends the ALTO endpoint property registry.

### 9.1. application/alto-\* Media Types

This document registers two additional ALTO media types, listed in Table 1.

Type	Subtype	Specification
application	alto-propmap+json	Section 4.1
application	alto-propmapparams+json	Section 5.3

Table 1: Additional ALTO Media Types.

Type name: application

Subtype name: This document registers multiple subtypes, as listed in Table 1.

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type. See [RFC7159].

Security considerations: Security considerations related to the generation and consumption of ALTO Protocol messages are discussed in Section 15 of [RFC7285].

Interoperability considerations: This document specifies formats of conforming messages and the interpretation thereof.

Published specification: This document is the specification for these media types; see Table 1 for the section documenting each media type.

Applications that use this media type: ALTO servers and ALTO clients either stand alone or are embedded within other applications.

Additional information: ~ Magic number(s): ~ n/a File extension(s): ~ This document uses the mime type to refer to protocol messages and

thus does not require a file extension. Macintosh file type code(s):  
~ n/a

Person & email address to contact for further information: See  
Authors' Addresses section.

Intended usage: COMMON

Restrictions on usage: n/a

Author: See Authors' Addresses section.

Change controller: Internet Engineering Task Force  
(mailto:iesg@ietf.org).

## 9.2. ALTO Entity Domain Registry

This document requests IANA to create and maintain the "ALTO Entity Domain Registry", listed in Table 2.

Identifier	Entity Address Encoding	Hierarchy & Inheritance
ipv4 ipv6 pid ane	See Section 3.1.1 See Section 3.1.2 See Section 3.2 See Section 3.4	See Section 3.1.3 See Section 3.1.3 None None

Table 2: ALTO Entity Domain Names.

This registry serves two purposes. First, it ensures uniqueness of identifiers referring to ALTO entity domains. Second, it states the requirements for allocated domain names.

New ALTO entity domains are assigned after IETF Review [RFC5226] to ensure that proper documentation regarding the new ALTO entity domains and their security considerations has been provided. RFCs defining new entity domains SHOULD indicate how an entity in a registered domain is encoded as an EntityName, and, if applicable, the rules defining the entity hierarchy and property inheritance. Updates and deletions of ALTO entity domains follow the same procedure.

Registered ALTO entity domain identifiers MUST conform to the syntactical requirements specified in Section 2.4. Identifiers are to be recorded and displayed as strings.



Requests to add a new value to the registry MUST include the following information:

- o Identifier: The name of the desired ALTO entity domain.
- o Entity Address Encoding: The procedure for encoding the address of an entity of the registered type as an EntityAddr (see Section 2.3).
- o Hierarchy: If the entities form a hierarchy, the procedure for determining that hierarchy.
- o Inheritance: If entities can inherit property values from other entities, the procedure for determining that inheritance.
- o Security Considerations: In some usage scenarios, entity addresses carried in ALTO Protocol messages MAY reveal information about an ALTO client or an ALTO service provider. Applications and ALTO service providers using addresses of the registered type SHOULD be made aware of how (or if) the addressing scheme relates to private information and network proximity.

This specification requests registration of the identifiers "ipv4", "ipv6" and "pid", as shown in Table 2.

### 9.3. ALTO Endpoint Property Type Registry

The ALTO Endpoint Property Type Registry was created by [RFC7285]. If possible, the name of that registry SHOULD be changed to "ALTO Entity Property Type Registry", to indicate that it is not restricted to Endpoint Properties. If it is not feasible to change the name, the description MUST be amended to indicate that it registers properties in all domains, rather than just the Internet address domain.

## 10. References

- [I-D.ietf-alto-path-vector]  
Bernstein, G., Chen, S., Gao, K., Lee, Y., Roome, W., Scharf, M., Yang, Y., and J. Zhang, "ALTO Extension: Path Vector Cost Mode", draft-ietf-alto-path-vector-00 (work in progress), May 2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<http://www.rfc-editor.org/info/rfc3986>>.
- [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <<http://www.rfc-editor.org/info/rfc4632>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<http://www.rfc-editor.org/info/rfc5226>>.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <<http://www.rfc-editor.org/info/rfc5952>>.
- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <<http://www.rfc-editor.org/info/rfc7159>>.
- [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, <<http://www.rfc-editor.org/info/rfc7285>>.

#### Authors' Addresses

Wendy Roome  
Nokia Bell Labs  
600 Mountain Ave, Rm 3B-324  
Murray Hill, NJ 07974  
USA

Phone: +1-908-582-7974  
Email: [wendy@roome.com](mailto:wendy@roome.com)

Y. Yang  
Yale University  
51 Prospect Street  
New Haven, CT 06511  
USA

Phone: +1-203-432-6400  
Email: yry@cs.yale.edu

CDNI	J. Seedorf
Internet-Draft	HFT Stuttgart - Univ. of Applied Sciences
Intended status: Standards Track	Y. Yang
Expires: January 3, 2018	Tongji/Yale
	K. Ma
	Ericsson
	J. Peterson
	Neustar
	July 2, 2017

Content Delivery Network Interconnection (CDNI) Request Routing: CDNI  
Footprint and Capabilities Advertisement using ALTO  
draft-seedorf-cdni-request-routing-alto-10

#### Abstract

The Content Delivery Networks Interconnection (CDNI) WG is defining a set of protocols to inter-connect CDNs, to achieve multiple goals such as extending the reach of a given CDN to areas that are not covered by that particular CDN. One componet that is needed to achieve the goal of CDNI is the CDNI Request Routing Footprint & Capabilities Advertisement interface (FCI) [RFC7336]. [RFC8008] has defined precisely the semantics of FCI and provided guidelines on the FCI protocol, but the exact protocol is explicitly outside the scope of that document. In this document, we define an FCI protocol using the Application Layer Traffic Optimization (ALTO) protocol.

#### Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 3, 2018.

## Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Background . . . . .	4
2.1. Semantics of FCI Advertisement . . . . .	4
2.2. ALTO Background and Benefits . . . . .	5
3. CDNI FCI ALTO Service . . . . .	7
3.1. Server Response Encoding . . . . .	8
3.1.1. Media Type . . . . .	8
3.1.2. Meta Information . . . . .	8
3.1.3. Data Information . . . . .	8
3.2. Protocol Errors . . . . .	8
3.3. Examples . . . . .	8
3.3.1. Basic Example . . . . .	8
3.3.2. Incremental FCI Update Example . . . . .	9
3.3.3. FCI Using ALTO Network Map Example . . . . .	9
4. Security Considerations . . . . .	9
5. Acknowledgements . . . . .	10
6. References . . . . .	10
6.1. Normative References . . . . .	10
6.2. Informative References . . . . .	11
Authors' Addresses . . . . .	11

## 1. Introduction

Many Network Service Providers (NSPs) are currently considering or have already started to deploy Content Delivery Networks (CDNs) within their networks. As a consequence of this development, there is a need for interconnecting these local CDNs. Content Delivery Networks Interconnection (CDNI) has the goal of standardizing protocols to enable such interconnection of CDNs [RFC6707].

The CDNI problem statement [RFC6707] defines four interfaces to be standardized within the IETF for CDN interconnection:

- o CDNI Request Routing Interface
- o CDNI Metadata Interface
- o CDNI Logging Interface
- o CDNI Control Interface

The main purpose of the CDNI Request Routing Interface is described in [RFC6707] as follows: "The CDNI Request Routing interface enables a Request Routing function in an Upstream CDN to query a Request Routing function in a Downstream CDN to determine if the Downstream CDN is able (and willing) to accept the delegated Content Request. It also allows the Downstream CDN to control what should be returned to the User Agent in the redirection message by the upstream Request Routing function." On a high level, the scope of the CDNI Request Routing Interface therefore contains two main tasks:

- o determining if the downstream CDN is willing to accept a delegated content request;
- o redirecting the content request coming from an upstream CDN to the proper entry point or entity in the downstream CDN.

Correspondingly, the request routing interface is broadly divided into two functionalities:

- o CDNI FCI: the advertisement from a dCDN to a uCDN or a query from a uCDN to a dCDN for the uCDN to decide whether to redirect particular user requests to that dCDN;
- o CDNI RI: the synchronous operation of actually redirecting a user request.

This document focuses solely on CDNI FCI, with a goal to specify a new Application Layer Traffic Optimization (ALTO) [RFC7285] service called 'CDNI/FCI Service', to transport and update CDNI FCI JSON objects, which are defined in a separate document in [RFC8008].

Throughout this document, we use the terminology for CDNI defined in [RFC6707] and [RFC8008].

## 2. Background

The design of CDNI FCI transport using ALTO depends on understanding of both FCI semantics and ALTO. Hence, we start with a review of both.

### 2.1. Semantics of FCI Advertisement

The CDNI document on "Footprint and Capabilities Semantics" [RFC8008] defines the semantics for the CDNI FCI. It thus provides guidance on what Footprint and Capabilities mean in a CDNI context and how a protocol solution should in principle look like. The definitions in [RFC8008] depend on [RFC8006]. Here we briefly summarize key related points of [RFC8008] and [RFC8006]. For a detailed discussion, the reader is referred to the RFCs.

- o Footprint and capabilities are tied together and cannot be interpreted independently from each other. In such cases, i.e. where capabilities must be expressed on a per footprint basis, it may be beneficial to combine footprint and capabilities advertisement. [RFC8008] integrates footprint and capabilities with an approach of "capabilities with footprint restrictions".
- o Given that a large part of Footprint and Capabilities Advertisement will actually happen in contractual agreements, the semantics of CDNI Footprint and Capabilities advertisement refer to answering the following question: what exactly still needs to be advertised by the CDNI FCI? For instance, updates about temporal failures of part of a footprint can be useful information to convey via the CDNI request routing interface. Such information would provide updates on information previously agreed in contracts between the participating CDNs. In other words, the CDNI FCI is a means for a dCDN to provide changes/updates regarding a footprint and/or capabilities it has prior agreed to serve in a contract with a uCDN. Hence, server push and incremental encoding will be necessary techniques.
- o Multiple types of footprints are defined in [RFC8006]:
  - \* List of ISO Country Codes
  - \* List of AS numbers
  - \* Set of IP-prefixes

A 'set of IP-prefixes' must be able to contain full IP addresses, i.e., a /32 for IPv4 and a /128 for IPv6, and also IP prefixes with an arbitrary prefix length. There must also be support for

multiple IP address versions, i.e., IPv4 and IPv6, in such a footprint.

- o For all of these mandatory-to-implement footprint types, footprints can be viewed as constraints for delegating requests to a dCDN: A dCDN footprint advertisement tells the uCDN the limitations for delegating a request to the dCDN. For IP prefixes or ASN(s), the footprint signals to the uCDN that it should consider the dCDN a candidate only if the IP address of the request routing source falls within the prefix set (or ASN, respectively). The CDNI specifications do not define how a given uCDN determines what address ranges are in a particular ASN. Similarly, for country codes a uCDN should only consider the dCDN a candidate if it covers the country of the request routing source. The CDNI specifications do not define how a given uCDN determines the country of the request routing source. Multiple footprint constraints are additive, i.e. the advertisement of different types of footprint narrows the dCDN candidacy cumulatively.
- o The following capabilities are defined as 'base' capabilities, i.e. ones that are needed in any case and therefore constitute mandatory capabilities to be supported by the CDNI FCI:
  - \* Delivery Protocol (e.g., HTTP vs. RTMP)
  - \* Acquisition Protocol (for acquiring content from a uCDN)
  - \* Redirection Mode (e.g., DNS Redirection vs. HTTP Redirection as discussed in [RFC7336])
  - \* Capabilities related to CDNI Logging (e.g., supported logging mechanisms)
  - \* Capabilities related to CDNI Metadata (e.g., authorization algorithms or support for proprietary vendor metadata)

## 2.2. ALTO Background and Benefits

Application Layer Traffic Optimization (ALTO) [RFC7285] is an approach for guiding the resource provider selection process in distributed applications that can choose among several candidate resources providers to retrieve a given resource. By conveying network layer (topology) information, an ALTO server can provide important information to "guide" the resource provider selection process in distributed applications. Usually, it is assumed that an ALTO server conveys information these applications cannot measure themselves [RFC5693].



Originally, ALTO was motivated by the huge amount of cross-ISP traffic generated by P2P applications [RFC5693]. Recently, however, ALTO is also being considered for improving the request routing in CDNs [I-D.jenkins-alto-cdn-use-cases]. In this context, it has also been proposed to use ALTO for selecting an entry-point in a downstream NSP's network (see section 3.4 "CDN delivering Over-The-Top of a NSP's network" in [I-D.jenkins-alto-cdn-use-cases]). Also, the CDNI problem statement explicitly mentions ALTO as a candidate protocol for "algorithms for selection of CDN or Surrogate by Request-Routing systems" [RFC6707].

The following reasons make ALTO a suitable candidate protocol for downstream CDN selection as part of CDNI request routing and in particular for an FCI protocol:

- o CDN request routing is done at the application layer. ALTO is a protocol specifically designed to improve application layer traffic (and application layer connections among hosts on the Internet) by providing additional information to applications that these applications could not easily retrieve themselves. For CDNI, this is exactly the case: a uCDN wants to improve application layer CDN request routing by using dedicated information (provided by a dCDN) that the uCDN could not easily obtain otherwise.
- o The semantics of an ALTO network map are an exact match for the needed information to convey a footprint by a downstream CDN, in particular if such a footprint is being expressed by IP-prefix ranges.
- o Security: ALTO maps can be signed and hence provide inherent integrity protection (see Section 4)
- o RESTful-Design: The ALTO protocol has undergone extensive revisions in order to provide a RESTful design regarding the client-server interaction specified by the protocol. A CDNI FCI interface based on ALTO would inherit this RESTful design.
- o Error-handling: The ALTO protocol has undergone extensive revisions in order to provide sophisticated error-handling, in particular regarding unexpected cases. A CDNI FCI interface based on ALTO would inherit this thought-through and mature error-handling.
- o Filtered network map: The ALTO Map Filtering Service (see [RFC7285] for details) would allow a uCDN to query only for parts of an ALTO map.

- o Server-initiated Notifications and Incremental Updates: In case the footprint or the capabilities of a downstream CDN change abruptly (i.e. unexpectedly from the perspective of an upstream CDN), server initiated notifications would enable a dCDN to directly inform an upstream CDN about such changes. Consider the case where - due to failure - part of the footprint of the dCDN is not functioning, i.e. the CDN cannot serve content to such clients with reasonable QoS. Without server-initiated notifications, the uCDN might still use a very recent network and cost map from dCDN, and therefore redirect request to dCDN which it cannot serve. Similarly, the possibility for incremental updates would enable efficient conveyance of the aforementioned (or similar) status changes by the dCDN to the uCDN. The newest design of ALTO supports server pushed incremental updates [I-D.ietf-alto-incr-update-sse].
- o Content Availability on Hosts: A dCDN might want to express CDN capabilities in terms of certain content types (e.g. codecs/formats, or content from certain content providers). The new endpoint property for ALTO would enable a dCDN to make available such information to an upstream CDN. This would enable a uCDN to determine if a given dCDN actually has the capabilities for a given request with respect to the type of content requested.
- o Resource Availability on Hosts or Links: The capabilities on links (e.g. maximum bandwidth) or caches (e.g. average load) might be useful information for an upstream CDN for optimized downstream CDN selection. For instance, if a uCDN receives a streaming request for content with a certain bitrate, it needs to know if it is likely that a dCDN can fulfill such stringent application-level requirements (i.e. can be expected to have enough consistent bandwidth) before it redirects the request. In general, if ALTO could convey such information via new endpoint properties, it would enable more sophisticated means for downstream CDN selection with ALTO.

### 3. CDNI FCI ALTO Service

The ALTO protocol is based on an ALTO Information Service Framework which consists of several services, where all ALTO services are 'provided through a common transport protocol, messaging structure and encoding, and transaction model' [RFC7285]. The ALTO protocol specification [RFC7285] defines several such services, e.g. the ALTO map service.

This document defines a new ALTO Service called 'CDNI Footprint & Capabilities Advertisement Service' which conveys JSON objects of media type 'application/cdni'. This media type and JSON object

format is defined in [RFC8006] and [RFC8008]; this document specifies how to transport such JSON objects via the ALTO protocol with the ALTO 'CDNI Footprint & Capabilities Advertisement Service'.

### 3.1. Server Response Encoding

#### 3.1.1. Media Type

The media type of the CDNI FCI Map is 'application/cdni'.

#### 3.1.2. Meta Information

The 'meta' field of a FCI response MUST include 'vtag', which is an ALTO Version Tag of the retrieved FCIMapData according to [RFC7285] (Section 10.3.). It thus contains a 'resource-id' attribute, and a 'tag' is an identifier string.

#### 3.1.3. Data Information

The data component of a CDNI FCI resource is named 'cdni-fcimap' which is a JSON object defined by [RFC8008]. This JSON object is derived from ResponseEntityBase as specified in the ALTO protocol [RFC7285] (Section 8.4.).

### 3.2. Protocol Errors

Protocol errors are handled as specified in the ALTO protocol [RFC7285] (Section 8.5.).

### 3.3. Examples

#### 3.3.1. Basic Example

The following example shows an CDNI FCI response as in [RFC8008], however with meta-information as defined in Section 3.1.2 of this document.

```
GET /fcimap HTTP/1.1
Host: alto.example.com
Accept: application/cdni,application/alto-error+json

HTTP/1.1 200 OK
Content-Length: 439
Content-Type: application/cdni
{
  "meta" : {
    "vtag": {
      "resource-id": "my-default-fcimap",
      "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
    }
  },
  "cdni-fcimap": {
    "capabilities": [
      {
        "capability-type": "FCI.DeliveryProtocol",
        "capability-value": {
          "delivery-protocols": [
            "http/1.1",
          ]
        },
        "footprints": [
          <Footprint objects>
        ]
      }
    ]
  }
}
```

### 3.3.2. Incremental FCI Update Example

### 3.3.3. FCI Using ALTO Network Map Example

## 4. Security Considerations

One important security consideration is the proper authentication of advertisement information provided by a downstream CDN. The ALTO protocol provides a specification for a signature of ALTO information (see 8.2.2. of [RFC7285]). ALTO thus provides a proper means for protecting the integrity of FCI information.

More Security Considerations will be discussed in a future version of this document.

## 5. Acknowledgements

The authors would like to thank Kevin Ma, Daryl Malas, and Matt Caulfield for their timely reviews and invaluable comments.

Jan Seedorf is partially supported by the GreenICN project (GreenICN: Architecture and Applications of Green Information Centric Networking), a research project supported jointly by the European Commission under its 7th Framework Program (contract no. 608518) and the National Institute of Information and Communications Technology (NICT) in Japan (contract no. 167). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the GreenICN project, the European Commission, or NICT.

## 6. References

### 6.1. Normative References

- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, DOI 10.17487/RFC5693, October 2009, <<http://www.rfc-editor.org/info/rfc5693>>.
- [RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, DOI 10.17487/RFC6707, September 2012, <<http://www.rfc-editor.org/info/rfc6707>>.
- [RFC6770] Bertrand, G., Ed., Stephan, E., Burbidge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", RFC 6770, DOI 10.17487/RFC6770, November 2012, <<http://www.rfc-editor.org/info/rfc6770>>.
- [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, <<http://www.rfc-editor.org/info/rfc7285>>.
- [RFC7336] Peterson, L., Davie, B., and R. van Brandenburg, Ed., "Framework for Content Distribution Network Interconnection (CDNI)", RFC 7336, DOI 10.17487/RFC7336, August 2014, <<http://www.rfc-editor.org/info/rfc7336>>.

- [RFC7337] Leung, K., Ed. and Y. Lee, Ed., "Content Distribution Network Interconnection (CDNI) Requirements", RFC 7337, DOI 10.17487/RFC7337, August 2014, <<http://www.rfc-editor.org/info/rfc7337>>.
- [RFC8006] Niven-Jenkins, B., Murray, R., Caulfield, M., and K. Ma, "Content Delivery Network Interconnection (CDNI) Metadata", RFC 8006, DOI 10.17487/RFC8006, December 2016, <<http://www.rfc-editor.org/info/rfc8006>>.
- [RFC8008] Seedorf, J., Peterson, J., Previdi, S., van Brandenburg, R., and K. Ma, "Content Delivery Network Interconnection (CDNI) Request Routing: Footprint and Capabilities Semantics", RFC 8008, DOI 10.17487/RFC8008, December 2016, <<http://www.rfc-editor.org/info/rfc8008>>.

## 6.2. Informative References

- [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-07 (work in progress), July 2017.
- [I-D.jenkins-alto-cdn-use-cases]  
Niven-Jenkins, B., Watson, G., Bitar, N., Medved, J., and S. Previdi, "Use Cases for ALTO within CDNs", draft-jenkins-alto-cdn-use-cases-03 (work in progress), June 2012.
- [I-D.ma-cdni-capabilities]  
Ma, K. and J. Seedorf, "CDNI Footprint & Capabilities Advertisement Interface", draft-ma-cdni-capabilities-09 (work in progress), April 2016.

## Authors' Addresses

Jan Seedorf  
HFT Stuttgart - Univ. of Applied Sciences  
Schellingstrasse 24  
Stuttgart 70174  
Germany

Phone: +49-0711-8926-2801  
Email: [jan.seedorf@hft-stuttgart.de](mailto:jan.seedorf@hft-stuttgart.de)

Y.R. Yang  
Tongji/Yale University  
51 Prospect Street  
New Haven, CT 06511  
United States of America

Email: [yry@cs.yale.edu](mailto:yry@cs.yale.edu)  
URI: <http://www.cs.yale.edu/~yry/>

Kevin J. Ma  
Ericsson  
43 Nagog Park  
Acton, MA 01720  
United States of America

Phone: +1-978-844-5100  
Email: [kevin.j.ma@ericsson.com](mailto:kevin.j.ma@ericsson.com)

Jon Peterson  
NeuStar  
1800 Sutter St Suite 570  
Concord, CA 94520  
United States of America

Email: [jon.peterson@neustar.biz](mailto:jon.peterson@neustar.biz)

ALTO WG  
Internet-Draft  
Intended status: Informational  
Expires: January 4, 2018

Q. Xiang  
Tongji/Yale University  
H. Newman  
California Institute of Technology  
G. Bernstein  
Grotto Networking  
H. Du  
Tongji/Yale University  
K. Gao  
Tsinghua University  
A. Mughal  
J. Balcas  
California Institute of Technology  
J. Zhang  
Tongji University  
Y. Yang  
Tongji/Yale University  
July 3, 2017

Resource Orchestration for Multi-Domain Data Analytics  
draft-xiang-alto-exascale-network-optimization-03.txt

## Abstract

Data-intensive analytics is entering the era of multi-domain, geographically-distributed, collaborative computing, where different organizations contribute various resources to collaboratively collect, share and analyze extremely large amounts of data. Examples of this paradigm include the Compact Muon Solenoid (CMS) and A Toroidal LHC ApparatuS (ATLAS) experiments of the Large Hadron Collider (LHC) program. Massive datasets continue to be acquired, simulated, processed and analyzed by globally distributed science networks in these collaborations. Applications that manage and analyze such massive data volumes can benefit substantially from the information about networking, computing and storage resources from each member's site, and more directly from network-resident services that optimize and load balance resource usage among multiple data transfers and analytics requests, and achieve a better utilization of multiple resources in clusters.

The Application-Layer Traffic Optimization (ALTO) protocol can provide via extensions the network information about different clusters/sites, to both users and proactive network management services where applicable, with the goal of improving both application performance and network resource utilization. In this document, we propose that it is feasible to use existing ALTO services to provides not only network information, but also



information about computation and storage resources in data analytics networks. We introduce a uniform resource orchestration framework (Unicorn), which achieves an efficient multi-resource allocation to support low-latency dataset transfer and data intensive analytics for collaborative computing. It collects cluster information from multiple ALTO services utilizing topology extensions and leverages emerging SDN control capabilities to orchestrate the resource allocation for dataset transfers and analytics tasks, leading to improved transfer and analytics latency as well as more efficient utilization of multi-resources in sites.

#### Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 4, 2018.

#### Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Requirements Language . . . . .	5
3. Changes Since Version -02 . . . . .	5
4. Problem Settings . . . . .	6
4.1. Motivation . . . . .	6
4.2. Challenges . . . . .	6
5. Basic Idea . . . . .	7
5.1. Use ALTO services to provide multi-resource information .	7
5.2. Example 1: network information impacts data analytics	
scheduling . . . . .	8
5.3. Example 2: encode multi-resources in abstract network	
elements . . . . .	9
6. Key Issues . . . . .	10
7. Unified Resource Orchestration Framework . . . . .	11
7.1. Architecture . . . . .	11
7.2. Workflow converter . . . . .	14
7.2.1. User API . . . . .	14
7.3. Resource Demand Estimator . . . . .	15
7.4. Entity Locator . . . . .	15
7.5. ALTO Client . . . . .	15
7.5.1. Query Mode . . . . .	16
7.6. ALTO Server . . . . .	16
7.7. Resource View Extractor . . . . .	16
7.8. Execution Agents . . . . .	18
7.9. Multi-Resource Orchestrator . . . . .	18
7.9.1. Orchestration Algorithms . . . . .	18
7.9.2. Online, Dynamic Orchestration . . . . .	18
7.9.3. Example: A Max-Min Fairness Resource Allocation	
Algorithm . . . . .	19
8. Discussion . . . . .	20
8.1. Deployment . . . . .	20
8.2. Benefiting From ALTO Extension Services . . . . .	21
8.3. Limitations of the MFRA Algorithm . . . . .	21
9. Security Considerations . . . . .	22
10. IANA Considerations . . . . .	22
11. Acknowledgments . . . . .	22
12. References . . . . .	22
12.1. Normative References . . . . .	22
12.2. Informative References . . . . .	22
Authors' Addresses . . . . .	24

## 1. Introduction

As the data volume increases exponentially over time, data intensive analytics is transiting from single-domain computing to multi-organizational, geographically-distributed, collaborative computing,

where different organizations contribute various resources, e.g., computation, storage and networking resources, to collaboratively collect, share and analyze extremely large amounts of data. One leading example is the Large Hadron Collider (LHC) high energy physics (HEP) program, which aims to find new particles and interactions in a previously inaccessible range of energies. The scientific collaborations that have built and operate large HEP experimental facilities at the LHC, such as the Compact Muon Solenoid (CMS) and A Toroidal LHC ApparatuS (ATLAS), currently have more than 300 petabytes of data under management at hundreds of sites around the world, and this volume is expected to grow to one exabyte by approximately 2018.

With such an increasing data volume, how to manage the storage and analytics of these data in a globally distributed infrastructure has become an increasingly challenging issue. Applications such as the Production AND Distributed Analysis system (PanDA) in ATLAS and the Physics Experiment Data Export system (PhEDEx) in CMS have been developed to manage the data transfers among different cluster sites. Given a data transfer request, these applications make data transfer decisions based on the availability of dataset replicas at different sites and initiate retransmission from a different replica if the original transmission fails or is excessively delayed. And HTCondor [HTCondor] is deployed to achieve coarse-grained data analytics parallelization across these sites. When a data analytics task is submitted, HTCondor adopts a match-making process to assign the task to a certain set of servers in one site, based on the coarse-grained description of resource availability, such as the number of cores, the size of memory, the size of hard disk, etc. However, neither dataset transfers nor data analytics task parallelization takes fine-grained information of cluster resources, such as data locality, memory speed, network delay, network bandwidth, etc., into account, leading to high data transfer and analytics latency and underutilization of cluster resources.

The Application-Layer Traffic Optimization (ALTO) services defined in [RFC7285] provide network information with the goal of improving the network resource utilization while maintaining or improving application performance. Though ALTO is not designed to provide information about other resources, such as computing and storage resources in cluster networks, in this document we propose that exascale science networks can leverage existing ALTO services defined in [RFC7285] and ALTO topology extension services defined in network graph [DRAFT-NETGRAPH], path vector [DRAFT-PV], routing state abstraction [DRAFT-RSA], multi-cost [DRAFT-MC] and cost-calendar [DRAFT-CC] and etc. to encode information about multiple types of resources in science networks, such as memory I/O speed, CPU utilization, network bandwidth, and provide such information to

orchestration applications to improve the performance of dataset transfer and data analytics tasks, including throughput, latency, etc.

This document introduces a unified resource orchestration framework (Unicorn), which provides efficient multi-resource allocation to support low-latency, multi-domain, geo-distributed data analytics. Unicorn provides a set of simple APIs for authorized users to submit, update and delete dataset transfer requests and data intensive analytics requests. One important proposal we make in this document is that it is feasible to use ALTO services to provide not only network information, but also information on other resources in multi-domain, geo-distributed analytics networks including computing and storage.

A prototype of Unicorn with the dataset transfer scheduling component has been implemented on a single-domain Caltech SDN development testbed, where the ALTO OpenDaylight controller is used to collect topology information. We are currently designing the resource orchestration components to achieve low-latency data-intensive analytics.

This document is organized as follows: Section 3 summarizes the change of this document since version -01. Section 4 elaborates on the motivation and challenges for coordinating storage, computing and network resources in a globally distributed science network infrastructure. Section 5 discusses the basic idea of encoding multi-resource information into ALTO path vector and abstraction services and gives an example. Section 6 lists several key issues to address in order to realize the proposal of providing multi-resource information by ALTO topology services. Section 7 gives the details of Unicorn architecture for multi-domain, geo-distributed data analytics. Section 8 discusses current development progress of Unicorn and next steps.

## 2. Requirements Language

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

## 3. Changes Since Version -02

- o Add an example in Section 7 to show the importance of network information in resource allocation for data analytics.

- o Update the architecture of Unicorn in Section 7, i.e., add the entity locator and rename ANE aggregator to resource view extractor.
- o Add detailed description of how the entity locator and the resource view extractor work.
- o Minor changes in abstract and discussion sections.

## 4. Problem Settings

### 4.1. Motivation

Multi-domain, geo-distributed data analytics usually involves the participation of countries and sites all over the world. Science programs such as the CMS experiment and the ATLAS experiment at CERN are typical examples. The site located at the LHC laboratory is called a Tier-0 site, which processes the data selected and stored locally by the online systems that select and record the data in real-time as it comes off the particle detector, archives it and transfers it to over 10 Tier-1 sites around the globe. Raw datasets and processed datasets from Tier-1 sites are then transferred to over 160 Tier-2 sites around the world based on users' requests. Different sites have different resources and belong to different administration domains. With the exponentially increasing data volume in the CMS experiment, the management of large data transfers and data intensive analytics in such a global multi-domain science network has become an increasingly challenging issue. Allocating resources in different clusters to fulfill different users' dataset transfer requests and data analytics requests require careful orchestrating as different requests are competing for limited storage, computation and network resources.

### 4.2. Challenges

Orchestrating exascale dataset transfers and analytics in a globally distributed science network is non-trivial as it needs to cope with two challenges.

- o Different sites in this network belong to different administration domains. Sharing raw site/cluster information would violate sites' privacy constraints. Orchestrating data transfers and analytics requests based on highly abstracted, non-real-time network information may lead to suboptimal scheduling decisions. Hence the orchestrating framework must be able to collect sufficient resource information about different clusters/sites in real-time as well as over the longer term, to allow reasonably

optimized network resource utilization without violating sites' privacy requirements.

- o Different science programs tend to adopt different software infrastructures for managing dataset transfers and analytics, and may place different requirements. Hence the orchestrating framework must be modular so that it can support different dataset management systems and different orchestrating algorithms.

The orchestrating framework must support the interaction between the multi-resource orchestration module, the dataset transfer module, and the data analytics execution module. The key information to be exchanged between modules includes dataset information, the resource state of different clusters and sites, the transfer and analytic requests in progress, as well as trends and network-segment and site performance from the network point of view. Such interaction ensures that (1) the various programs can adopt their own data transfer and analytics systems to be multi-resource-aware, and more efficient in achieving their goals; and (2) the various orchestrating algorithms can achieve a reasonably optimized utilization on not only the network resource but also the computing and storage resources.

## 5. Basic Idea

### 5.1. Use ALTO services to provide multi-resource information

The ALTO protocol is designed to provide network information to applications so that applications can achieve better performance and the network more efficient use of resources. Different ALTO topology services including path vector, routing state abstraction, multi-cost, cost calendar, etc., have been proposed to provide fine-grained network information to applications. In this document, we propose that not only can ALTO provide network information of different cluster sites, it can also provide information of multiple resources, including computing and storage resources. To this end, the basic "one-big-switch" abstraction provided by the base ALTO protocol is not sufficient. Several examples have already been given in [DRAFT-PV] and [DRAFT-RSA] to demonstrate that. There has been a similar proposal before about using ALTO to provide resource information for data centers [DRAFT-DC]. However, that proposal requires a new information model for clusters or data centers, which may affect the compatibility of ALTO. The solution of this proposal is simpler. Its basic idea is that each computer node and storage node can be seen as an "abstract network element" defined in ALTO-path-vector [DRAFT-PV]. In this way, Unicorn can fully reuse all existing ALTO services by introducing only one cost-mode (pv) and two cost-metrics (ne and ane), instead of introducing a new information model.

## 5.2. Example 1: network information impacts data analytics scheduling

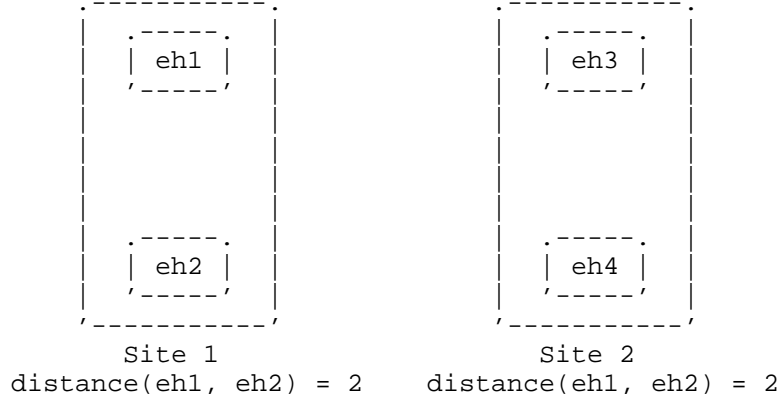


Figure 1: An Example for Data Locality Information.

We first use the example in Figure 1 to show that only network information itself has a huge impact on resource allocation for data analytics. In this scenario, a MapReduce task needs to be executed. The input data has two copies at end hosts eh1 and eh3, respectively. And the end hosts eh2 and eh4 will be the computation nodes with the same computation power, correspondingly. Using the common data analytics resource management framework such as Hadoop it can be revealed that both allocation options, i.e., eh1->eh2 and eh3->eh4, have a storage-computation distance of 2, i.e., they have the same data locality from Hadoop's view. As a result, it appears that both options would provide the same performance for this task.

However, assume that the bandwidth between eh1 and eh2 is 100Mb/s while that between eh3 and eh4 is 1Gb/s. These significant different data accessing speeds decide that these options have very different performances for the same task and the only optimal allocation option is to allocate this task to eh3->eh4. This example demonstrates the imperativeness of network information in making efficient resource allocation decisions. Such information is not provided in Hadoop or other similar or related projects such as Mesos. On the contrary, if ALTO servers are deployed at these sites, applications can retrieve such information via ALTO queries.

### 5.3. Example 2: encode multi-resources in abstract network elements

We use the same dumbbell topology in [DRAFT-RSA] as an example to show the feasibility of using ALTO topology service to provide multi-resource information. In this topology, we assume the bandwidth of each network cable is 1Gbps, including the cables connecting end hosts to switches. Consider a dataset transfer request which needs to schedule the traffic among a set of end host source-destination pairs, say eh1  $\rightarrow$  eh2, and eh3  $\rightarrow$  eh4. Assume that the transfer application receives information from the ALTO Cost Map service that both eh1  $\rightarrow$  eh2 and eh3  $\rightarrow$  eh4 have bandwidth 1Gbps. In [DRAFT-RSA], it is shown that whether each of the two traffic flows can receive 1Gbps bandwidth depends on whether the routes of two flows share a bottleneck link. Path vector and routing state abstraction services provide additional information about network state encoded in abstract network elements. If the returned state is  $ane1 + ane2 \leq 1\text{Gbps}$ , it means two flows cannot each get 1Gbps bandwidth at the same time. If the returned state is  $ane1 \leq 1\text{Gbps}$  and  $ane2 \leq 1\text{Gbps}$ , it means two flows each can get 1Gbps bandwidth.

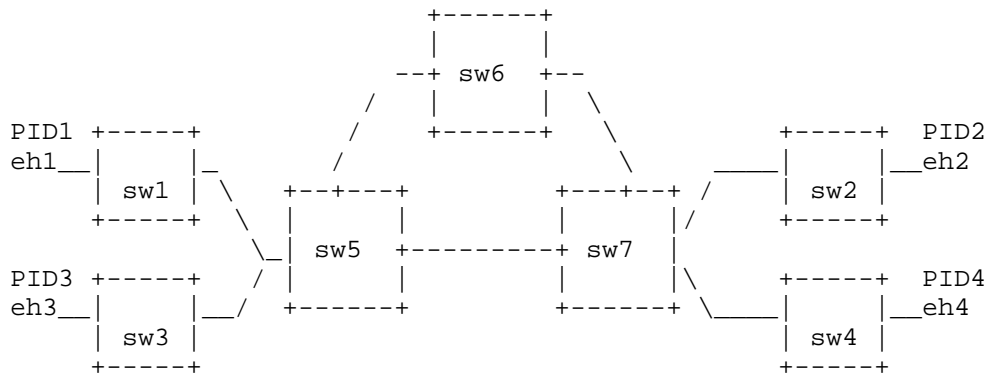


Figure 2: A Dumbbell Network Topology

Other than network resource, assume in this topology eh1 and eh3 are equipped with commodity hard disk drives (HDD) while eh2 and eh4 are equipped with SSDs. Because the bandwidth of an HDD is typically 0.8Gbps and that of SSD is typically 3Gbps. Even if the returned routing state is  $ane1 \leq 1\text{Gbps}$  and  $ane2 \leq 1\text{Gbps}$ , the actual bottleneck of each traffic flow is the storage I/O bandwidth at source host. As a result, the total bandwidth of both traffic flows can only reach 1.6Gbps.



It has been verified in the CMS experiment, and also several studies on commercial data centers that network resource are not always the bottleneck of large dataset transfers and data analytics. Many have reported that storage resources and computing resources become the bottleneck in a fairly large percentage of dataset transfers and data analytics tasks in science networks and commercial data centers.

In this example, if we look at the end hosts as abstract network elements, the storage I/O bandwidth of each host can also be encoded as an abstract element into the path-vector. And under the storage and route settings above, the returned cluster state would be  $ane1 \leq 0.8\text{Gbps}$  and  $ane2 \leq 0.8\text{Gbps}$ , which provides a more accurate capacity region for the requested traffic flows.

## 6. Key Issues

The last section described the basic idea of using ALTO topology services to provide multi-resource information and gives an example to demonstrate its feasibility. Next we list and discuss several key issues to address in this proposal.

- o Can ALTO topology services provide data locality information? Existing ALTO topology services do not provide such information. Many studies have pointed out that such information plays a vital role in reducing the latency of data-intensive analytics. If ALTO topology services can encode such information together with information of other resources together, data-intensive applications can benefit a great deal in terms of information aggregation and communication overhead.
- o How to quickly map applications' resource allocation decision on abstract multi-resource view back to the physical multi-resource view of clusters/sites? Fine-grained resource information can be encoded into abstract network elements to reduce overhead and provide certain privacy protection of clusters. Such information can be highly compressed (see the dumbbell example used in this document as well as in [DRAFT-PV] and [DRAFT-RSA]). In preliminary evaluations on RSA, the network element compression ratio can be as high as 80 percent. This ratio is expected to be even higher in large-scale data center or cluster setting, e.g. a fat-tree topology with  $k=48$ . Therefore a fast mapping from the resource orchestration decisions based on the abstract view back to the physical view is needed to satisfy the stringent latency requirement of large dataset transfers and data-intensive analytics.
- o How much privacy, including key resource configurations, raw topology, intra-cluster scheduling policy, etc., will be exposed?

Compared with the "one-big-switch" abstraction, other ALTO topology services such as path vector [DRAFT-PV] and routing state abstraction [DRAFT-RSA] provide fine-grained resource information to applications. Even if such information can be encoded into abstract network elements, it still risks exposing private information of different clusters/sites. Current internet drafts of these services did not provide any formal privacy analysis or performance measurement. This would be one of the key issues this document plans to investigate in the future.

- o How does current ALTO services such as path-vector and RSA scale when they are used to provide abstract information concerning multiple resources in clusters? Another issue along this line is how to balance the liveness of fine-grained resource information and the corresponding information delivery overhead? Although encoding information of network elements into abstract network elements can achieve a very competitive information compression ratio, large dataset transfers and analytics applications always involve many network elements in multiple clusters/sites and the absolute number of involved network elements keep increasing as the scale of clusters increase. In addition, when resource information in a cluster changes, the ALTO services need to inform all related applications. In either cases, delivering fine-grained resource information would cause high communication overhead. There still lacks of an analytics or experimental understanding on the scalability of path-vector and RSA services.

## 7. Unified Resource Orchestration Framework

### 7.1. Architecture

This section describes the design details of key components of the Unicorn framework: the workflow converter, the resource demand estimator, the ALTO clients, the ALTO servers, the resource view extractor, the multi-resource orchestrator and the execution agents. Figure 3 shows the architecture of Unicorn. The overall process is as follows.

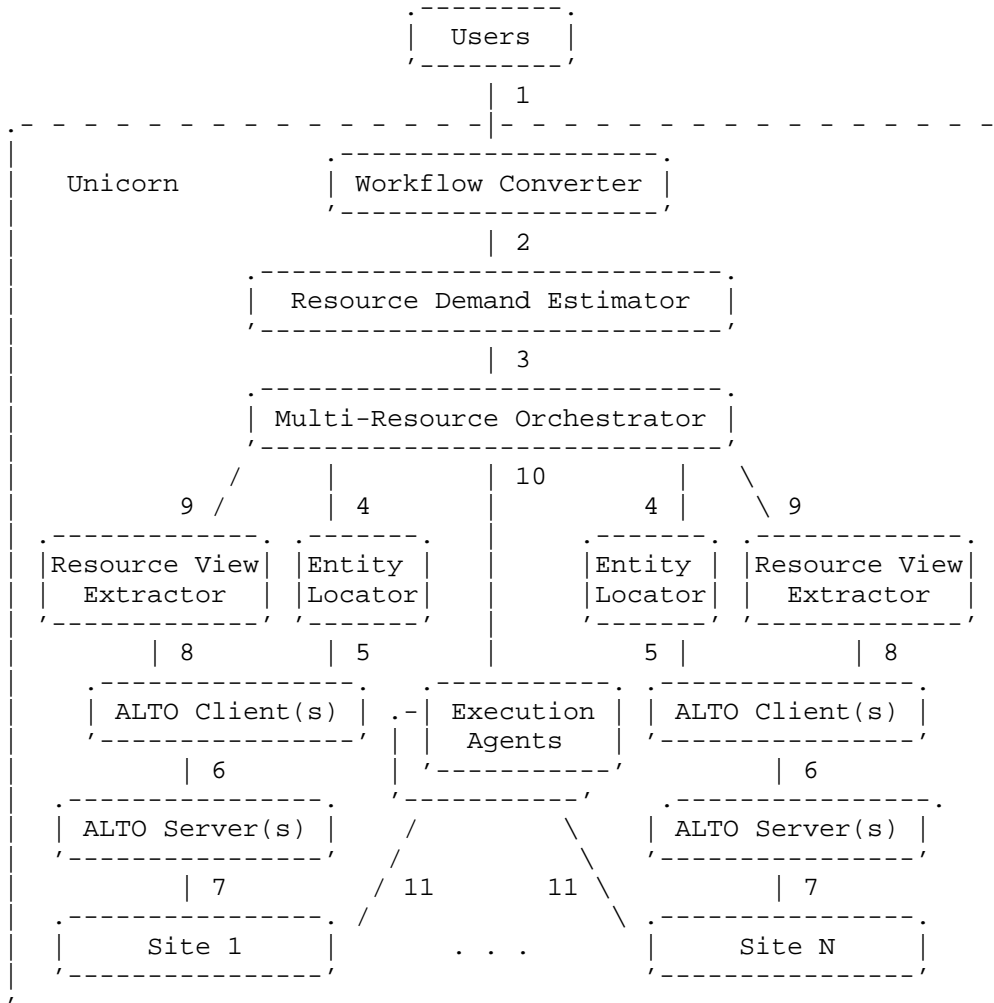


Figure 3: Architecture of Unicorn.

- o STEP 1 Authorized users submit high-level data analytics workflows to Unicorn through a set of simple APIs.
- o STEP 2 The workflow converter transforms the high-level data analytics workflows into low-level task workflows, i.e., a set of analytics tasks with precedence encoded in a directed acyclic graph (DAG).

- o STEP 3 The resource demand estimator automatically finds the optimal configuration (resource demand) of each task, i.e., the number of CPUs, the size of memory and disk, I/O bandwidth, etc.
- o STEP 4 The multi-resource orchestrator receives the resource demand of a set of tasks and sends them to the entity locator at each site.
- o STEP 5 The entity locator at each site retrieves the entity addresses of the end hosts that would be allocated for the tasks to be scheduled, and passes these addresses to the ALTO clients, and asks the ALTO clients to collect information about these end hosts, i.e., properties of corresponding computing, storage and networking resources.
- o STEP 6 The ALTO clients issue ALTO queries defined in the base ALTO protocol [RFC7285], e.g., EPS, ECS, Network Map, etc. and ALTO extension services, e.g., routing state abstraction (RSA) [DRAFT-RSA], path vector [DRAFT-PV], network graph [DRAFT-NETGRAPH], multi-cost [DRAFT-MC], cost-calendar [DRAFT-CC] and property map [DRAFT-PM], to collect resource information.
- o STEP 7 The ALTO servers at each site accept the queries from the ALTO client, collect resource information from the residing site and send back to the ALTO clients.
- o STEP 8 The ALTO clients send the response from ALTO servers to the resource view extractor.
- o STEP 9 The extractor uses a lightweight, optimal algorithm to compress the raw resource information provided by ALTO servers into a minimal, equivalent view of resources and sends back to the multi-resource orchestrator.
- o STEP 10 The orchestrator makes resource allocation decisions, e.g., dataset transfer scheduling and analytics task placement, based on the resource demand of analytics tasks and the resource supply sent back from the resource view extractor. Decisions are then sent to the execution agents deployed in corresponding sites.
- o STEP 11 The execution agents receive and execute instructions from the multi-resource orchestrator. They also monitor the status of different tasks and send the updated status to the multi-resource orchestrator.

Unicorn provides a unified, automatic solution for multi-domain, geo-distributed data analytics. In particular, its benefits include:

- o On the resource demand side, it provides a set of simple APIs for authorized users to submit and manage data analytics requests and enables real-time requests' status monitoring. And it automatically converts high-level analytics workflow into low-level task workflows and finds the optimal configuration for each task.
- o On the resource supply side, it collects the resource information of different sites through a common, REST based interface specified by the ALTO protocol, encodes such information into different entity abstractions and computes a minimal, yet accurate view on resource supply dynamic.
- o It provides a scalable multi-resource orchestrator that makes efficient resource allocation decisions to achieve high resource utilization and low-latency data analytics.
- o The architecture of Unicorn is modular to support different resource orchestration algorithms and the deployment of different ALTO servers.

## 7.2. Workflow converter

The converter is the front end of Unicorn. It is responsible for collecting high-level data analytics workflows from users and transforming them into low-level task workflows, e.g., HTCondor ClassAds. It provides a set of simple APIs for users to submit and manage requests, and to track the status of requests in real-time.

### 7.2.1. User API

- o `submitReq(request, [options])`

This API allows users to submit a request and specify corresponding options. The request can be a data transfer request or a data analytics request. Request options include priority, delay, etc. It returns a request identifier `reqID` that allows users to update, delete this request or track its status. The additional options may or may not be approved, and the relative priorities may be modified by the resource orchestrator depending on the role of users (regular users or administrators at different levels), the resource availability and the status of other ongoing requests.

- o `updateReq(requestID, [options])`

This API allows users to update the options of requests. It will return a SUCCESS if the new options are received by the request

parser. But these new options may or may not be approved, and may be modified by the resource orchestrator depending on the role of users (regular users or administrators), the resource availability and the status of other ongoing requests.

- o deleteReq(requestID)

This API allows users to delete a request by passing the corresponding requestID. A completed request cannot be deleted. An ongoing request will be stopped and the output data will be deleted.

- o getReqStatus(requestID)

This API allows users to query the status of a request by specifying the corresponding requestID. The returned status information includes whether the request has started, the assigned priority, the percentage of finished sub-requests, transmission statistics, the expected remaining time to finish, etc.

### 7.3. Resource Demand Estimator

The estimator leverages the fact that low-level tasks are typically repetitive or have very high similarities. It uses reinforcement learning to predict the optimal configuration for each task and passes the resource demand to the multi-resource orchestrator for further processing.

### 7.4. Entity Locator

The task configurations computed by the demand estimator has no knowledge on the underlying structure of topology of each site, i.e., the addresses of end hosts and network devices. Hence they cannot be directly used by the ALTO clients for querying resource information. The entity locator retrieves the entity addresses of the end hosts that would be allocated for the tasks to be scheduled, and passes these addresses to the ALTO clients, and asks the ALTO clients to collect information about these end hosts, i.e., properties of corresponding computing, storage and networking resources.

### 7.5. ALTO Client

The ALTO client is in the back end of Unicorn and is responsible for retrieving resource information through querying ALTO servers deployed at different sites. The resource information needed in Unicorn includes the topology, link bandwidth, computing node memory I/O speed, computing node CPU utilization, etc. The base ALTO protocol [RFC7285] provides an extreme single-node abstraction for

this information, which only allows the multi-resource orchestrator to make coarse-grained resource allocation decisions. To enable fine-grained multi-resource orchestration for dataset transfer and data analytics in cluster networks, ALTO topology extension services such as routing state abstraction (RSA) [DRAFT-RSA], path vector [DRAFT-PV], network graph [DRAFT-NETGRAPH], multi-cost [DRAFT-MC] and cost-calendar [DRAFT-CC] are needed to provide fine-grained information about different types of resources in clusters.

#### 7.5.1. Query Mode

The ALTO client should operate in different query modes depending on the implementation of ALTO servers. If an ALTO server does not support incremental updates using server-sent events (SSE) [DRAFT-SSE], the ALTO client sends queries to this server periodically to get the latest resource information. If the cluster state changes after one query, the ALTO client will not be aware of the change until next query. If an ALTO server supports SSE, the ALTO client only sends one query to the ALTO server to get the initial cluster information. When the resource state changes, the ALTO client will be notified by the ALTO server through SSE.

#### 7.6. ALTO Server

ALTO servers are deployed at different sites around the world, and at strategic locations in the network itself, to provide information about different types of resources in the cluster networks in response to queries from the ALTO client. Such information include topology, link bandwidth, memory I/O speed and CPU utilization at computing nodes, storage constraints in storage nodes and etc. Each ALTO server must provide basic information services as specified in [RFC7285] such as network map, cost map, endpoint cost service (ECS), etc. To support the fine-grained multi-resource allocation in Unicorn, each ALTO server should also provide more fine-grained information about different resources in clusters through ALTO extension services such as the routing state abstraction [DRAFT-RSA], path vector [DRAFT-PV], network graph [DRAFT-NETGRAPH], multi-cost [DRAFT-MC] and cost-calendar [DRAFT-CC] services.

#### 7.7. Resource View Extractor

In each site, the resource information collected by the ALTO clients is not directly sent back to the orchestrator. Instead, we design a resource view extractor to compress the resource information provided by the ALTO servers into a minimal, equivalent view of all the resources, i.e., computing, storage and networking resources, that would be allocated to a set of tasks. The extractor works in the following steps.

- o Resource-Task Matrix

Depending on specific services provided by the ALTO servers, the responses collected by ALTO clients may include information of different entities, e.g., endpoints, PIDs, etc. Each entity possesses a set of resources available for tasks to be scheduled. For each entity property  $p$  in the responses, such as bandwidth, delay, etc., the extractor composes an entity-task matrix  $M(p)$ . Each row of this  $M(p)$  represents an entity in the responses provides information about property  $p$  and each column represents a task to be scheduled. The element  $m(i, j)$  of  $M(p)$  is a variable representing the amount of property  $p$  of entity  $i$  that task  $j$  can get.

- o Resource-Task Constraints

For each property  $p$ , the extractor composes a series of resource-task constraints. The first set of constraints is  $\sum m(i, j) = r(p, j)$  for each task  $j$ . These constraints calculate  $r(p, j)$ , the total amount of property  $p$  provided to  $j$  by all the entities. The exact rule of summation depends on the property  $p$ . For instance, if  $p$  is latency, the summation is through common addition operations, but if  $p$  is bandwidth, the summation is a minimum function.

The second set of constraints is  $\sum m(i, j) \leq r(p, i)$  for each entity  $i$ . These constraints represent the usage of resource  $i$  on property  $p$  for all the tasks cannot exceed the capacity of resource  $i$  on this property. The exact rule of summation also depends on the property  $p$ . For instance, if  $p$  is bandwidth, the summation is through common addition operations, but if  $p$  is latency, the constraints become  $m(i, j) = r(p, i)$  for each entity  $i$  and each task  $j$ . This means that entity  $i$  provides the same delay property for each task.

- o Resource View Compression

The whole set of resource-task constraints are linear, and express the view of resources that are available for the tasks to be scheduled. The extractor uses a lightweight, optimal algorithm to compress them into a minimal, equivalent view of resources, i.e., a minimal set of linear constraints that represent the same feasible region as the original constraints. The basic idea of this algorithm is described in [DRAFT-RSA].



## 7.8. Execution Agents

Execution agents are deployed at each site and are responsible for the following functions:

- o Receive and process instructions from the multi-resource orchestrator, e.g. dataset transfer scheduling, data analytics task placement and execution, task update and abortion, etc.
- o Monitor the status of data analytics tasks and send the updated status to the multi-resource orchestrator.

Depending on the supporting data analytics frameworks, different request execution agents may be deployed in each site. For instance, in the CMS experiment at CERN, both MPI and Hadoop execution agents are deployed.

## 7.9. Multi-Resource Orchestrator

The multi-resource orchestrator receives the resource demand information, i.e., a set of low-level task workflows and their configurations, from the resource demand estimator. It then asks the entity locator in each site to get the addresses of end hosts that would be allocated for the tasks to be scheduled and ALTO clients to query properties related to these end hosts. When the resource view extractor sends the response back, the orchestrator makes resource allocation decisions, e.g., dataset transfer scheduling and analytics task execution, based on both resource demand dynamic and resource supply dynamic. The dataset transfer scheduling decisions include dataset replica selection, path selection, and bandwidth allocation, etc. The analytics task execution decisions include which cluster should allocate how much resources to execute which tasks. These decisions are sent to the execution agents at different sites for execution.

### 7.9.1. Orchestration Algorithms

The modular design of Unicorn allows the adoption of different orchestration algorithms and methodologies, depending on the specific performance requirements. In Section 7.8.3, a max-min fairness resource allocation algorithm for dataset transfer is described as an example.

### 7.9.2. Online, Dynamic Orchestration

The multi-resource orchestrator should adjust the resource allocation decisions based on the progress of ongoing requests, the utilization and dynamics of cluster resources. In normal cases, the multi-

resource orchestrator periodically collects such information and executes the orchestration algorithm. When it is notified of events such as request status update, cluster state update and etc., the orchestrator will also execute the orchestration algorithm to adjust resource allocations.

### 7.9.3. Example: A Max-Min Fairness Resource Allocation Algorithm

In this section, we describe a max-min fair resource allocation (MFRA) scheduling algorithm which aims to minimize the maximal time to complete a dataset transfer subject to a set of constraints. To make resource allocation decisions, MFRA requires sufficient network information including topology, link bandwidth and recent historical information in some cases. In a small-scale single-domain network, an SDN controller can provide the raw complete topology information for the MFRA algorithm. However, in a large-scale multi-domain science network such as CMS, providing the raw network topology is infeasible because (1) it would incur significant communication overhead; and (2) it would violate the privacy constraints of some sites. Several ALTO extension topology services including Abstract Path Vector [DRAFT-PV], Network Graphs [DRAFT-NETGRAPH] and RSA [DRAFT-RSA] can provide the fine-grained yet aggregated/abstract topology information for MFRA to efficiently utilize bandwidth resources in the network.

Ongoing pre-production deployment efforts of Unicorn in the CMS network involve the implementation of the RSA service. Other than topology information, the additional input of the MFRA algorithm is the priority of each class of flows, expressed in terms of upper and lower limits on the allocated bandwidth between the source and the destination for each data transfer requests.

The basic idea of the MFRA algorithm is to iteratively maximize the volume of data that can be transferred subject to the constraints. It works in quantized time intervals such that it schedules network paths and data volumes to be transferred in each time slot. When the DTR scheduler is notified of events such as the cancellation of a DTR, the completion of a DTR or network state changes, the MFRA algorithm will also be invoked to make updated network path and bandwidth allocation decisions.

In each execution cycle, MFRA first marks all transfers as unsaturated. Then it solves a linear programming model to find the common minimum transfer satisfaction rate (i.e., the ratio of transferred data volume in a time interval over the whole data volume of this request) that is satisfied by all transfer requests. With this common rate found, MFRA then randomly selects an unsaturated request in each iteration, increases its transfer rate as much as

possible by finding residual paths available in the network, or by increasing the allocated bandwidth along an existing path, until it reaches its upper limit or can otherwise not be increased further, so it is saturated. At each iteration, newly saturated requests are removed from the subsequent process by fixing their corresponding rate value, and completed transfers are removed from further consideration. After all the data transfer rates are saturated in the given time slot, then a feasible set of data transfer volumes scheduled to be transferred in the slot across each link in the network can be derived.

The MFRA algorithm yields a full utilization of limited network resources such as bandwidth so that all DTR can be completed in a timely manner. It allocates network resources fairly so that no DTR suffers starvation. It also achieves load balance among the sites and the network paths crossing a complex network topology so that no site and no network link is oversubscribed. Moreover, MFRA can handle the case where particular routing constraints are specified, e.g., where all routes are fixed ahead of time, or where each transfer request only uses one single path in each time slot, by introducing an additional set of linear constraints.

## 8. Discussion

### 8.1. Deployment

The Unicorn framework is the first step towards a new class of intelligent, SDN-driven global systems for multi-domain, geo-distributed data analytics involving a worldwide ensemble of sites and networks, such as CMS and ATLAS. Unicorn relies heavily on the ALTO services for collecting and expressing abstract, real-time resource information from different sites, and the SDN centralized control capability to orchestrate data analytics workflows. It aims to provide a new operational paradigm in which science programs can use complex network and computing infrastructures with high throughput, while allowing for coexistence with other network traffic.

A prototype case study implementation of Unicorn has been demonstrated on the Caltech/StarLight/Michigan/Fermilab SDN development testbed. Because this testbed is a single-domain network, the current Unicorn prototype leverages the ALTO OpenDaylight controller, to collect topology information. The CMS experiment is currently exploring pre-production deployments of Unicorn, looking towards future widespread production use. To achieve this goal, it is imperative to collect sufficient resource information from the various sites in the multi-domain CMS network, without causing any privacy leak. To this end, the ALTO RSA service

[DRAFT-RSA] is under development. Furthermore, as will be discussed next, other ALTO topology extension services can also substantially improve the performance of Unicorn.

## 8.2. Benefiting From ALTO Extension Services

The current ALTO base protocol [RFC7285] exposes network topology and endpoint properties using the extreme "my-Internet-view" representation, which abstracts a whole network as a single node that has a set of access ports, with each port connects to a set of end hosts called endpoints. Such an extreme abstraction leads to significant information loss on network topology [DRAFT-PV], which is the key information for Unicorn to make dynamic scheduling and resource allocation decisions. Though Unicorn can still allocate resource for data transfer and analytics requests on this abstract view, the resource allocation decisions are suboptimal. Alternatively, feeding the raw, complete network topology of each site to Unicorn is not desirable, either. First, this would violate privacy constraints of different sites. Secondly, a raw network topology would significantly increase the problem space and the solution space of the orchestrating algorithm, leading to a long computation time. Hence, Unicorn desires an ALTO topology service that is able to provide only enough fine-grained topology information.

Several ALTO topology extension services including Path Vector [DRAFT-PV], Network Graphs [DRAFT-NETGRAPH] and RSA [DRAFT-RSA], [DRAFT-PM] are potential candidates for providing fine-grained abstract network formation to Unicorn. In addition, we propose that these services can also be used to provide information about computing and storage resources of different cluster/sites by viewing each computing node and storage node as a network element or abstract network element. For instance, the path vector service supports the capacity region query, which accepts multiple concurrent data flows as the input and returns the information of bottleneck resources, which could be a set of links, computing devices or storage devices, for the given set of concurrent flows. This information can be interpreted as a set of linear constraints for the multi-resource orchestrator, which can help data transfer and analytics requests better utilize multiple types of resources in different clusters.

## 8.3. Limitations of the MFRA Algorithm

The first limitation of the MFRA algorithm is computation overhead. The execution of MFRA involves solving linear programming problems repeatedly at every time slot. The overhead of computation time is acceptable for small sets of dataset transfer requests, but may increase significantly when handling large sets of requests, e.g.,

hundreds of transfer requests. Current efforts towards addressing this issue include exploring the feasibility of incremental computation of scheduling policies, and reducing the problem scale by finding the minimal equivalent set of constraints of the linear programming model. The latter approach can benefit substantially from the ALTO RSA service [DRAFT-RSA].

The second limitation is that the current version of MFRA does not involve dataset replica selection. Simply denoting the replica selection as a set of binary constraint will significantly increase the computation complexity of the scheduling process. Current efforts focus on finding efficient algorithms to make dataset replica selection.

## 9. Security Considerations

This document does not introduce any privacy or security issue not already present in the ALTO protocol.

## 10. IANA Considerations

This document does not define any new media type or introduce any new IANA consideration.

## 11. Acknowledgments

The authors thank discussions with Shenshen Chen, Linghe Kong, Xiao Lin and Xin Wang.

## 12. References

### 12.1. 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, <<http://www.rfc-editor.org/info/rfc2119>>.

### 12.2. Informative References

[DRAFT-CC] Randriamasy, S., Yang, R., Wu, Q., Deng, L., and N. Schwan, "ALTO Cost Calendar", 2017, <<https://datatracker.ietf.org/doc/draft-ietf-alto-cost-calendar>>.

## [DRAFT-DC]

Lee, Y., Bernstein, G., Dhody, D., and T. Choi, "ALTO Extensions for Collecting Data Center Resource Information", 2014, <<https://datatracker.ietf.org/doc/draft-lee-alto-ext-dc-resource/>>.

## [DRAFT-MC]

Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost ALTO", 2017, <<https://datatracker.ietf.org/doc/draft-ietf-alto-multi-cost/>>.

## [DRAFT-NETGRAPH]

Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Topology Extensions: Node-Link Graphs", 2015, <<https://tools.ietf.org/html/draft-yang-alto-topology-06>>.

## [DRAFT-PM]

Roome, W. and Y. Yang, "Extensible Property Maps for the ALTO Protocol", 2015, <<https://datatracker.ietf.org/doc/draft-roome-alto-unified-props-new/>>.

## [DRAFT-PV]

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

## [DRAFT-RSA]

Gao, K., Wang, X., Yang, Y., and G. Chen, "ALTO Extension: A Routing State Abstraction Service Using Declarative Equivalence", 2015, <<https://datatracker.ietf.org/doc/draft-gao-alto-routing-state-abstraction/>>.

## [DRAFT-SSE]

Roome, W. and Y. Yang, "ALTO Incremental Updates Using Server-Sent Events (SSE)", 2015, <<https://datatracker.ietf.org/doc/draft-ietf-alto-incr-update-sse/>>.

## [HTCondor]

Thain, D., Tannenbaum, T., and M. Livny, "Distributed computing in practice: the Condor experience", 2005, <<http://dl.acm.org/citation.cfm?id=1064336>>.

[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, <<http://www.rfc-editor.org/info/rfc7285>>.

#### Authors' Addresses

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

Email: [qiao.xiang@cs.yale.edu](mailto:qiao.xiang@cs.yale.edu)

Harvey Newman  
California Institute of Technology  
1200 California Blvd.  
Pasadena, CA  
USA

Email: [newman@hep.caltech.edu](mailto:newman@hep.caltech.edu)

Greg Bernstein  
Grotto Networking  
Fremont, CA  
USA

Email: [gregb@grotto-networking.com](mailto:gregb@grotto-networking.com)

Haizhou Du  
Tongji/Yale University  
51 Prospect Street  
New Haven, CT  
USA

Email: [duhaizhou@gmail.com](mailto:duhaizhou@gmail.com)

Kai Gao  
Tsinghua University  
30 Shuangqinglu Street  
Beijing  
China

Email: gaok12@mails.tsinghua.edu.cn

Azher Mughal  
California Institute of Technology  
1200 California Blvd.  
Pasadena, CA  
USA

Email: azher@hep.caltech.edu

Justas Balcas  
California Institute of Technology  
1200 California Blvd.  
Pasadena, CA  
USA

Email: justas.balcas@cern.ch

Jingxuan Jensen Zhang  
Tongji University  
4800 Cao'an Hwy  
Shanghai 201804  
China

Email: jingxuan.n.zhang@gmail.com

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

Email: yry@cs.yale.edu