ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: November 9, 2013

R. Alimi, Ed.
Google
R. Penno, Ed.
Cisco Systems
Y. Yang, Ed.
Yale University
May 8, 2013

ALTO Protocol
draft-ietf-alto-protocol-15.txt

Abstract

Applications using the Internet already have access to topology information of Internet Service Provider (ISP) networks. For example, views to Internet routing tables at looking glass servers are available and can be practically downloaded to many application clients. What is missing is knowledge of the underlying network topologies from the point of view of ISPs (henceforth referred as Providers). In other words, what a Provider prefers in terms of traffic optimization -- and a way to distribute it.

The Application-Layer Traffic Optimization (ALTO) Service provides network information (e.g., basic network location structure and preferences of network paths) with the goal of modifying network resource consumption patterns while maintaining or improving application performance. The basic information of ALTO is based on abstract maps of a network. These maps provide a simplified view, yet enough information about a network for applications to effectively utilize them. Additional services are built on top of the maps.

This document describes a protocol implementing the ALTO Service. Although the ALTO Service would primarily be provided by the network service providers (e.g., Internet service providers), content service providers and third parties could also operate an ALTO service. Applications that could use this service are those that have a choice to which end points to connect. Examples of such applications are peer-to-peer (P2P) and content delivery networks.

Requirements Language

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

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Background and Problem Statement

Today, network information available to applications is mostly from the view of endhosts. There is no clear mechanism for a network to convey to network applications its point of view on its network topological structures and path preferences, forcing applications to make approximations using data sources such as BGP Looking Glass and/or applications' own measurements, which can be misleading or inaccurate. On the other hand, modern network applications can be adaptive, and hence become more network-efficient (e.g., reduce network resource consumption) and achieve better application performance (e.g., accelerated download rate), by leveraging better network-provided information.

This document defines the ALTO protocol to implement the ALTO Service, which provides a simple mechanism to convey useful network information such as topological and path preference information to applications from the underlying network Providers' points of view. The ALTO protocol meets the ALTO requirements [I-D.ietf-alto-reqs], and unifies multiple protocols previously designed with similar intentions. See Appendix A for a list of people and Appendix B for a list of proposals that have made significant contributions to this effort.

The ALTO protocol uses a REST-ful design [Fielding-Thesis], and encodes its requests and responses using JSON [RFC4627]. These designs are chosen because of their flexibility and extensibility. In addition, these designs make it possible for ALTO to be deployed at scale by leveraging existing HTTP [RFC2616] implementations, infrastructures and deployment experience.

1.2. Solution Benefits

At a high level, the ALTO Service allows a Service Provider (e.g., an ISP) to publish network information such as network locations, costs between them at configurable granularities, and endhost properties.

A mechanism to publish such information can benefit both Service Providers (providers of the information) and Applications (consumers of the information). We enumerate some benefits below.

1.2.1. Service Providers

A Service Provider that provides an ALTO Service can achieve better utilization of its networking infrastructure. For example, by using ALTO as a tool to interact with applications, a Service Provider is

able to provide network information to applications so that the applications can better manage traffic on more expensive or difficult to provision links such as long distance, transit or backup links.

1.2.2. Applications

Applications that use an ALTO Service can benefit from better knowledge of the network to avoid network bottlenecks. For example, a peer-to-peer overlay application can use information provided by an ALTO Service to avoid selecting peers connected with low bandwidth links. By using ALTO information, applications can reduce the reliance on obtaining network information through third-party databases; applications relying on measuring path performance metrics themselves can reduce the measurement overhead by conducting only fine-tuning or fault-tolerant measurements on top of ALTO information.

2. Terminology

We use the following terms defined in [RFC5693]: Application, Overlay Network, Peer, Resource, Resource Identifier, Resource Provider, Resource Consumer, Resource Directory, Transport Address, Host Location Attribute, ALTO Service, ALTO Server, ALTO Client, ALTO Query, ALTO Reply, ALTO Transaction, Local Traffic, Peering Traffic, Transit Traffic.

We also use the following additional terms: Endpoint Address, Autonomous System Number (ASN), Network Location, ALTO Information, and ALTO Information Base.

2.1. Endpoint

An Endpoint is an application or host that is capable of communicating (sending and/or receiving messages) on a network.

An Endpoint is typically either a Resource Provider or Resource Consumer.

2.2. Endpoint Address

An Endpoint Address represents the communication address of an endpoint. Common forms of Endpoint Addresses include IP address, MAC address, overlay ID, and phone number. An Endpoint Address can be network-attachment based (e.g., IP address) or network-attachment agnostic (e.g., MAC address).

Each Endpoint Address has an associated Address Type, which indicates

both its syntax and semantics.

2.3. ASN

An Autonomous System Number.

2.4. Network Location

Network Location is a generic term denoting a single Endpoint or a group of Endpoints. For instance, it can be a single IPv4 or IPv6 address, an IPv4 or IPv6 prefix, or a set of prefixes.

2.5. ALTO Information

ALTO Information is a generic term referring to the network information sent by an ALTO Server.

2.6. ALTO Information Base

Internal representation of the ALTO Information maintained by the ALTO Server. Note that the structure of this internal representation is not defined by this document.

3. Architecture

We now define the ALTO architecture and the ALTO Protocol's place in the overall architecture.

3.1. ALTO Service and Protocol Scope

Each network region in the global Internet can provide its ALTO Service, which conveys network information from the perspective of that network region. A network region in this context can be an Autonomous System (AS), an ISP, a region smaller than an AS or ISP, or a set of ISPs. The specific network region that an ALTO Service represents will depend on the ALTO deployment scenario and ALTO service discovery mechanism.

Specifically, the ALTO Service of a network region defines network Endpoints (and aggregations thereof) and generic costs amongst them from the region's perspective. The network Endpoints may include all Endpoints in the global Internet. Hence, we say that the network information provided by the ALTO Service of a network region represents the "my-Internet View" of the network region.

To better understand the ALTO Service and the role of the ALTO Protocol, we show in Figure 1 the overall ALTO system architecture.

In this architecture, an ALTO Server prepares ALTO Information; an ALTO Client uses ALTO Service Discovery to identify an appropriate ALTO Server; and the ALTO Client requests available ALTO Information from the ALTO Server using the ALTO Protocol.

The ALTO Information provided by the ALTO Server can be updated dynamically based on network conditions, or can be seen as a policy which is updated at a larger time-scale.

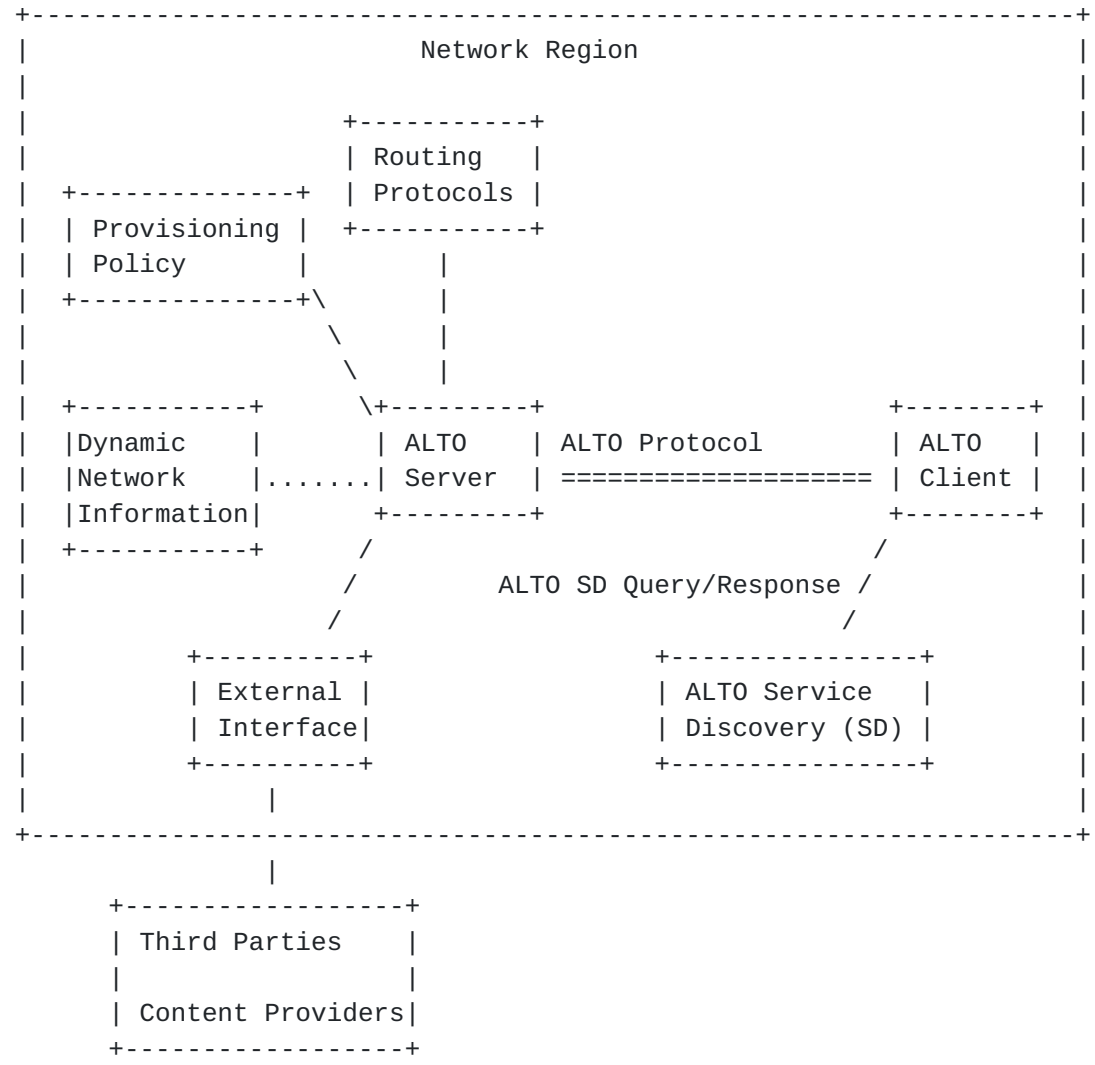

Figure 1: Basic ALTO Architecture.

Figure 1 illustrates that the ALTO Information provided by an ALTO Server may be influenced (at the service provider's discretion) by other systems. In particular, the ALTO Server can aggregate information from multiple systems to provide an abstract and unified view that can be more useful to applications. Examples of other systems include (but are not limited to) static network configuration

databases, dynamic network information, routing protocols, provisioning policies, and interfaces to outside parties. These components are shown in the figure for completeness but are outside the scope of this specification. Recall that while the ALTO Protocol may convey dynamic network information, it is not intended to replace near-real-time congestion control protocols.

It may also be possible for an ALTO Server to exchange network information with other ALTO Servers (either within the same administrative domain or another administrative domain with the consent of both parties) in order to adjust exported ALTO Information. Such a protocol is also outside the scope of this specification.

3.2. ALTO Information Reuse and Redistribution

ALTO Information may be useful to a large number of applications and users. At the same time, distributing ALTO Information must be efficient and not become a bottleneck.

The design of ALTO allows integration with the existing HTTP caching infrastructure to redistribute ALTO Information. If caching or redistribution is used, the response message to an ALTO Client may be returned from a third-party.

Application-dependent mechanisms, such as P2P DHTs or P2P file-sharing, may be used to cache and redistribute ALTO Information. This document does not define particular mechanisms for such redistribution.

Additional protocol mechanisms (e.g., expiration times and digital signatures for returned ALTO information) are left for future investigation.

4. ALTO Information Service Framework

The ALTO Protocol conveys network information through services, where each service defines a set of related functionalities. An ALTO Client can query each service individually. All of the services defined in ALTO are said to form the ALTO service framework and are provided through a common transport protocol, messaging structure and encoding, and transaction model. Functionalities offered in different services can overlap.

In this document, we focus on achieving the goal of conveying (1) Network Locations, which denote the locations of Endpoints at a network, (2) provider-defined costs for paths between pairs of

Network Locations, and (3) network related properties of endhosts. We achieve the goal by defining the Map Service, which provides the core ALTO information to clients, and three additional services: the Map Filtering Service, Endpoint Property Service, and Endpoint Cost Service. Additional services can be defined in companion documents. Below we give an overview of the services. Details of the services will be presented in the following sections.

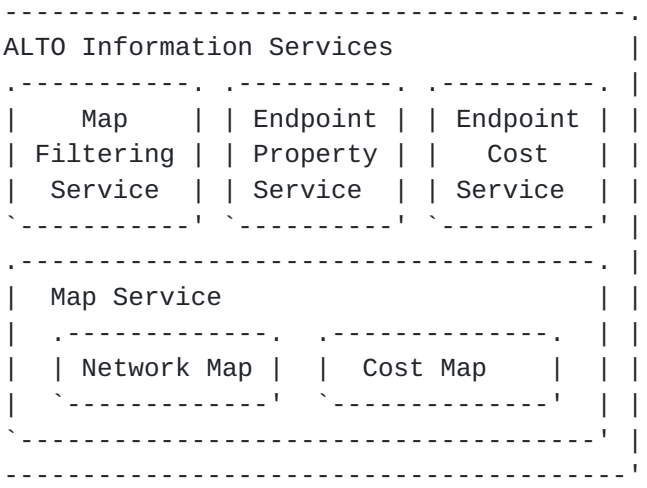

Figure 2: ALTO Service Framework.

4.1. ALTO Information Services

4.1.1. Map Service

The Map Service provides batch information to ALTO Clients in the form of Network Map and Cost Map. The Network Map (See Section 5) provides the full set of Network Location groupings defined by the ALTO Server and the Endpoints contained within each grouping. The Cost Map (see Section 6) provides costs between the defined groupings.

These two maps can be thought of (and implemented as) as simple files with appropriate encoding provided by the ALTO Server.

4.1.2. Map Filtering Service

Resource constrained ALTO Clients may benefit from filtering of query results at the ALTO Server. This avoids that an ALTO Client spends network bandwidth and CPU cycles to collect results and then perform client-side filtering. The Map Filtering Service allows ALTO Clients to query for the ALTO Server Network Map and Cost Map based on additional parameters.

4.1.3. Endpoint Property Service

This service allows ALTO Clients to look up properties for individual Endpoints. An example property of an Endpoint is its Network Location (i.e., its grouping defined by the ALTO Server). Another example property is its connectivity type such as ADSL (Asymmetric Digital Subscriber Line), Cable, or FTTH (Fiber To The Home).

4.1.4. Endpoint Cost Service

Some ALTO Clients may also benefit from querying for costs and rankings based on Endpoints. The Endpoint Cost Service allows an ALTO Server to return either numerical costs or ordinal costs (rankings) directly amongst Endpoints.

5. Network Map

An ALTO Network Map defines a grouping of network endpoints. In this document, we use Network Map to refer to the syntax and semantics of how an ALTO Server distributes the grouping. This document does not discuss the internal representation of this data structure within the ALTO Server.

The definition of Network Map is based on the observation that in reality, many endpoints are close by to one another in terms of network connectivity. By treating a group of close-by endpoints together as a single entity, an ALTO Server indicates aggregation of these endpoints due to their proximity. This aggregation can also lead to greater scalability without losing critical information when conveying other network information (e.g., when defining Cost Map).

5.1. Provider-defined Identifier (PID)

One issue is that proximity varies depending on the granularity of the ALTO information configured by the provider. In one deployment, endpoints on the same subnet may be considered close; while in another deployment, endpoints connected to the same Point of Presence (PoP) may be considered close.

ALTO introduces provider-defined Network Location identifiers called Provider-defined Identifiers (PIDs) to provide an indirect and network-agnostic way to specify an aggregation of network endpoints that may be treated similarly, based on network topology, type, or other properties. Specifically, a PID is a US-ASCII string of type PIDName (see Section 9.1) and its associated set of Endpoint Addresses. As we discussed above, there can be many different ways of grouping the endpoints and assigning PIDs. For example, a PID may

denote a subnet, a set of subnets, a metropolitan area, a PoP, an autonomous system, or a set of autonomous systems.

A key use case of PIDs is to specify network preferences (costs) between PIDs instead of individual endpoints. This allows cost information to be more compactly represented and updated at a faster time scale than the network aggregations themselves. For example, an ISP may prefer that endpoints associated with the same PoP (Point-of-Presence) in a P2P application communicate locally instead of communicating with endpoints in other PoPs. The ISP may aggregate endhosts within a PoP into a single PID in the Network Map. The cost may be encoded to indicate that Network Locations within the same PID are preferred; for example, cost(PID_i, PID_i) == c and cost(PID_i, PID_j) > c for i != j. Section 6 provides further details on using PIDs to represent costs in an ALTO Cost Map.

5.2. Endpoint Addresses

The endpoints aggregated into a PID are denoted by endpoint addresses. There are many types of addresses, such as IP addresses, MAC addresses, or overlay IDs. This specification only considers IP addresses.

5.2.1. IP Addresses

When either an ALTO Client or ALTO Server needs to determine which PID in a Network Map contains a particular IP address, longest-prefix matching MUST be used.

A Network Map MUST define a PID for each possible address in the IP address space for all of the address types contained in the map. A RECOMMENDED way to satisfy this property is to define a PID with the shortest enclosing prefix of the addresses provided in the map. For a map with full IPv4 reachability, this would mean including the 0.0.0.0/0 prefix in a PID; for full IPv6 reachability, this would be the ::/0 prefix.

Each endpoint MUST map into exactly one PID. Since longest-prefix matching is used to map an endpoint to a PID, this can be accomplished by ensuring that no two PIDs contain an identical IP prefix.

5.3. Example Network Map

Figure 3 illustrates an example Network Map. PIDs are used to identify network-agnostic aggregations.

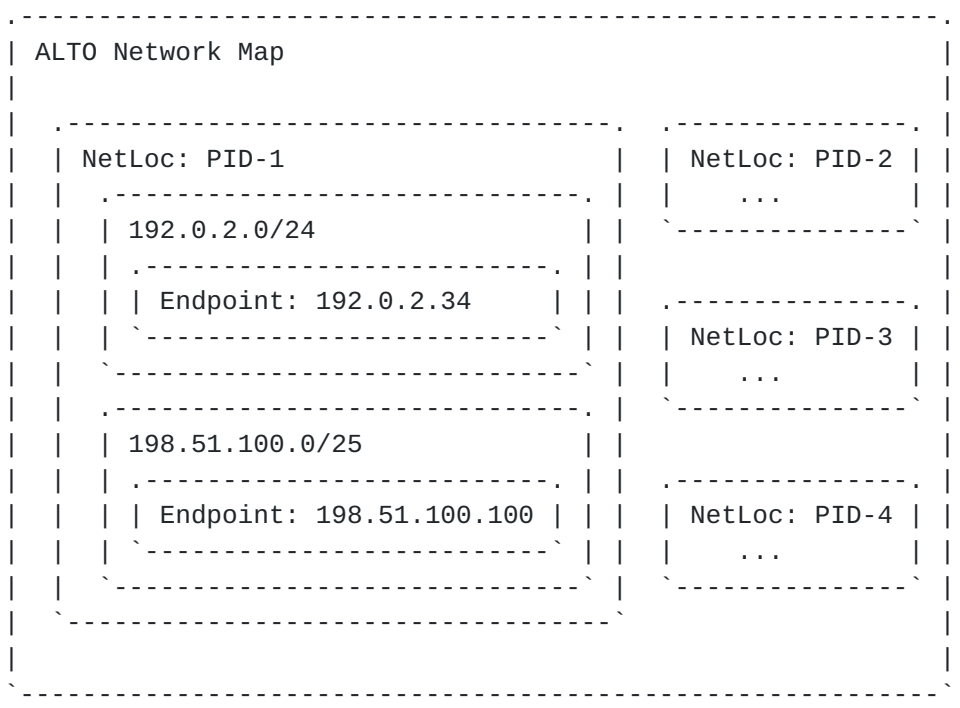

Figure 3: Example Network Map.

6. Cost Map

An ALTO Server indicates preferences amongst network locations in the form of Path Costs. Path Costs are generic costs and can be internally computed by a network provider according to its own needs.

An ALTO Cost Map defines Path Costs pairwise amongst sets of source and destination Network Locations defined by PIDs. Each Path Cost is the end-to-end cost when a unit of traffic goes from the source to the destination.

As cost is directional from the source to the destination, an application, when using ALTO Information, may independently determine how the Resource Consumer and Resource Provider are designated as the source or destination in an ALTO query, and hence how to utilize the Path Cost provided by ALTO Information. For example, if the cost is expected to be correlated with throughput, a typical application concerned with bulk data retrieval may use the Resource Provider as the source, and Resource Consumer as the destination.

One advantage of separating ALTO information into a Network Map and a Cost Map is that the two components can be updated at different time scales. For example, Network Maps may be stable for a longer time while Cost Maps may be updated to reflect dynamic network conditions.

As used in this document, the Cost Map refers to the syntax and semantics of the information distributed by the ALTO Server. This document does not discuss the internal representation of this data structure within the ALTO Server.

6.1. Cost Types

Path Costs have attributes:

- Metric: identifies what the costs represent;
- Mode: identifies how the costs should be interpreted.

The combination of a metric and a mode defines a Cost Type. Certain queries for Cost Maps allow the ALTO Client to indicate the desired Cost Type.

6.1.1. Cost Metric

The Metric attribute indicates what the cost represents. For example, an ALTO Server could define costs representing air-miles, hop-counts, or generic routing costs.

Cost metrics are indicated in protocol messages as strings.

6.1.1.1. Cost Metric: routingcost

An ALTO Server MUST offer the 'routingcost' Cost Metric.

This Cost Metric conveys a generic measure for the cost of routing traffic from a source to a destination. Lower values indicate a higher preference for traffic to be sent from a source to a destination.

Note that an ISP may internally compute routing cost using any method it chooses (e.g., air-miles or hop-count) as long as it conforms to these semantics.

6.1.2. Cost Mode

The Mode attribute indicates how costs should be interpreted. Specifically, the Mode attribute indicates whether returned costs should be interpreted as numerical values or ordinal rankings.

It is important to communicate such information to ALTO Clients, as certain operations may not be valid on certain costs returned by an ALTO Server. For example, it is possible for an ALTO Server to return a set of IP addresses with costs indicating a ranking of the

IP addresses. Arithmetic operations that would make sense for numerical values, do not make sense for ordinal rankings. ALTO Clients may handle such costs differently.

Cost Modes are indicated in protocol messages as strings.

An ALTO Server MUST support at least one of 'numerical' and 'ordinal' modes. An ALTO Client SHOULD be cognizant of operations when a desired Cost Mode is not supported. For example, an ALTO Client desiring numerical costs may adjust its behaviors if only the ordinal Cost Mode is available. Alternatively, an ALTO Client desiring ordinal costs may construct ordinal costs given numerical values if only the numerical Cost Mode is available.

6.1.2.1. Cost Mode: numerical

This Cost Mode is indicated by the string 'numerical'. This mode indicates that it is safe to perform numerical operations (e.g. normalization or computing ratios for weighted load-balancing) on the returned costs. The values are floating-point numbers.

6.1.2.2. Cost Mode: ordinal

This Cost Mode is indicated by the string 'ordinal'. This mode indicates that the costs values in a Cost Map are a ranking (relative to all other values in a Cost Map), with lower values indicating a higher preference. The values are non-negative integers. Ordinal cost values in a Cost Map need not be unique nor contiguous. In particular, it is possible that two entries in a map have an identical rank (ordinal cost value). This document does not specify any behavior by an ALTO Client in this case; an ALTO Client may decide to break ties by random selection, other application knowledge, or some other means.

It is important to note that the values in the Cost Map provided with the ordinal Cost Mode are not necessarily the actual cost known to the ALTO Server.

6.2. Cost Map Structure

A query for a Cost Map either explicitly or implicitly includes a list of Source Network Locations and a list of Destination Network Locations. (Recall that a Network Location can be an endpoint address or a PID.)

Specifically, assume that a query has a list of multiple Source Network Locations, say [Src_1, Src_2, ..., Src_m], and a list of multiple Destination Network Locations, say [Dst_1, Dst_2, ...,

Dst_n].

The ALTO Server will return the Path Cost for each of the m*n communicating pairs (i.e., Src_1 -> Dst_1, ..., Src_1 -> Dst_n, ..., Src_m -> Dst_1, ..., Src_m -> Dst_n). If the ALTO Server does not define a Path Cost for a particular pair, it may be omitted. We refer to this structure as a Cost Map.

If the Cost Mode is 'ordinal', the Path Cost of each communicating pair is relative to the m*n entries.

6.3. Network Map and Cost Map Dependency

If a Cost Map contains PIDs in the list of Source Network Locations or the list of Destination Network Locations, the Path Costs are generated based on a particular Network Map (which defines the PIDs). Version Tags are introduced to ensure that ALTO Clients are able to use consistent information even though the information is provided in two maps.

A Version Tag is an opaque string associated with a Network Map maintained by the ALTO Server. Two Version Tags match only if their strings are the same. Whenever the content of the Network Map maintained by the ALTO Server changes, the Version Tag MUST also be changed. Possibilities for generating a Version Tag include the last-modified timestamp for the Network Map, or a hash of its contents, where the collision probability is considered zero in practical deployment scenarios.

A Network Map distributed by the ALTO Server includes its Version Tag. A Cost Map referring to PIDs also includes the Version Tag of the Network Map on which it is based.

7. Endpoint Properties

An endpoint property defines a network-aware property of an endpoint.

7.1. Endpoint Property Type

For each endpoint and an endpoint property type, there can be a value for the property. The type of an Endpoint property is indicated in protocol messages as a string. The value depends on the specific property. For example, for a property such as whether an endpoint is metered, the value is a true or false value.

7.1.1. Endpoint Property Type: pid

An ALTO Server MUST define the 'pid' Endpoint Property Type, which provides the PID of an endpoint. Since the PID of an endpoint depends on the Network Map, Version Tag of the Network Map used to return the pid property MUST be included.

8. Protocol Specification: General Processing

This section first specifies general client and server processing. The details of specific services will be covered in the following sections.

8.1. Overall Design

The ALTO Protocol uses a REST-ful design. There are two primary components to this design:

- Information Resources: Each service provides network information as a set of information resources, which are distinguished by their media types [RFC2046]. An ALTO Client may construct an HTTP request for a particular information resource (including any parameters, if necessary), and an ALTO Server returns the requested information resource in an HTTP response.

- Information Resource Directory (IRD): An ALTO Server provides to ALTO Clients a list of available information resources and the URI at which each is provided. This document refers to this list as the Information Resource Directory. ALTO Clients consult the directory to determine the services provided by an ALTO Server.

8.2. Notation

This document uses 'JSONString', 'JSONNumber', 'JSONBool' to indicate the JSON string, number, and boolean types, respectively. The type 'JSONValue' indicates a JSON value, as specified in Section 2.1 of [RFC4627].

We use an adaptation of the C-style struct notation to define the members (names/values) of JSON objects. An optional member is enclosed by [ ], and an array is indicated by two numbers, m and n, where m indicates the minimal number of values, and n is the maximum. When we write .. for n, it means no upper bound. In the definitions, the JSON names of the members are case sensitive.

For example, the definition below defines a new type Type4, with three members named "name1", "name2", and "name3" respectively. The

member named "name3" is optional, and the member named "name2" is an array of at least one value.

```
object {
  Type1   name1;
  Type2   name2<1..*>;
  [Type3 name3;]
} Type4;
```

We also define dictionary maps (or maps for short) from strings to JSON values. For example, the definition below defines a Type3 object as a map. Type1 must be defined as string, and Type2 can be any type.

```
object-map {
  Type1   -> Type2;
} Type3;
```

Note that despite the notation, no standard, machine-readable interface definition or schema is provided. Extension documents may document these as necessary.

8.3. Basic Operation

The ALTO Protocol employs standard HTTP [RFC2616]. It is used for discovering available Information Resources at an ALTO Server and retrieving Information Resources. ALTO Clients and ALTO Servers use HTTP requests and responses carrying ALTO-specific content with encoding as specified in this document, and MUST be compliant with [RFC2616].

8.3.1. Client Discovering Information Resources

To discover available Information Resources, an ALTO Client requests the Information Resource Directory, which an ALTO Server provides at the URI found by the ALTO Discovery protocol.

Informally, an Information Resource Directory enumerates URIs at which an ALTO Server offers Information Resources. Each entry in the directory indicates a URI at which an ALTO Server accepts requests, and returns either the requested Information Resource or an Information Resource Directory that references additional Information Resources. See Section 8.5 for a detailed specification.

8.3.2. Client Requesting Information Resources

Through the retrieved Information Resource Directories, an ALTO Client can determine whether an ALTO Server supports the desired Information Resource, and if it is supported, the URI at which it is available.

Where possible, the ALTO Protocol uses the HTTP GET method to request resources. However, some ALTO services provide Information Resources that are the function of one or more input parameters. Input parameters are encoded in the HTTP request's entity body, and the ALTO Client MUST use the HTTP POST method to send the parameters.

When requesting an ALTO Information Resource that requires input parameters specified in a HTTP POST request, an ALTO Client MUST set the Content-Type HTTP header to the media type corresponding to the format of the supplied input parameters.

8.3.3. Server Responding to IR Request

Upon receiving a request for an Information Resource that the ALTO Server can provide, the ALTO server MUST return the requested Information Resource. In other cases, to be more informative ([I-D.ietf-httpbis-p2-semantics]), the ALTO server MAY provide the ALTO Client with an Information Resource Directory indicating how to reach the desired information resource, or return an ALTO error object; see Section 8.7 for more details on ALTO error handling.

It is possible for an ALTO Server to leverage caching HTTP intermediaries to respond to both GET and POST requests by including explicit freshness information (see Section 14 of [RFC2616]). Caching of POST requests is not widely implemented by HTTP intermediaries, however an alternative approach is for an ALTO Server, in response to POST requests, to return an HTTP 303 status code ("See Other") indicating to the ALTO Client that the resulting Information Resource is available via a GET request to an alternate URL. HTTP intermediaries that do not support caching of POST requests could then cache the response to the GET request from the ALTO Client following the alternate URL in the 303 response if the response to the subsequent GET request contains explicit freshness information.

The ALTO server MUST indicate the type of its response using a media type (i.e., the Content-Type HTTP header of the response).

8.3.4. Client Handling Server Response

8.3.4.1. Using Information Resources

This specification does not indicate any required actions taken by ALTO Clients upon successfully receiving an Information Resource from an ALTO Server. Although ALTO Clients are suggested to interpret the received ALTO Information and adapt application behavior, ALTO Clients are not required to do so.

8.3.4.2. Handling IRD as a Response

After receiving an Information Resource Directory, the Client can consult it to determine if any of the offered URIs contain the desired Information Resource. Note that it is possible for an ALTO Client to receive an Information Resource Directory from an ALTO Server as a response to its request for a specific Information Resource. In this case, the ALTO Client may ignore the response or still parse the response. To indicate that an ALTO Client will always check if a response is an Information Resource Directory, the ALTO Client can indicate in the "Accept" header of a HTTP request that it can accept Information Resource Directory; see Section 8.5 for the media type.

8.3.4.3. Handling Error Conditions

If an ALTO Client does not successfully receive a desired Information Resource from a particular ALTO Server (i.e., server response indicates error or there is no response), the Client can either choose another server (if one is available) or fall back to a default behavior (e.g., perform peer selection without the use of ALTO information, when used in a peer-to-peer system).

8.3.5. Authentication and Encryption

When server and/or client authentication, encryption, and/or integrity protection are required, an ALTO Server MUST support SSL/TLS [RFC5246] as a mechanism. For cases such as a public ALTO service or deployment scenarios where there is an implicit trust relationship between the client and the server and the network infrastructure connecting them is secure, SSL/TLS may not be necessary. See [RFC6125] for considerations regarding verification of server identity.

8.3.6. HTTP Cookies

If cookies are included in an HTTP request received by an ALTO Server, they MUST be ignored.

8.3.7. Parsing

This document only details object members used by this specification. Extensions may include additional members within JSON objects defined in this document. ALTO implementations MUST ignore such unknown fields when processing ALTO messages.

8.4. Information Resource: Attributes

An Information Resource encodes the ALTO Information desired by an ALTO Client. This document specifies multiple Information Resources that can be provided by an ALTO Server.

Each Information Resource has certain attributes associated with it, including its data format, its capabilities, and its accepted input parameters. These attributes are published by an ALTO Server in its Information Resource Directory.

8.4.1. Media Type

A Media Type [RFC2046] uniquely indicates a data format used to encode the content of an Information Resource that an ALTO Server returns to an ALTO Client in the HTTP entity body.

8.4.2. Capabilities

The Capabilities associated with an Information Resource announced by an ALTO Server indicates specific capabilities that the server can provide. For example, if an ALTO Server allows an ALTO Client to specify cost constraints when the Client requests a Cost Map Information Resource, the Server advertises the cost-constraints capability for its Cost Map Information Resource.

8.4.3. Accepts Input Parameters

An ALTO Server may allow an ALTO Client to supply input parameters when requesting certain Information Resources. The associated accepts attribute of an Information Resource is a Media Type, which indicates how the Client specifies the input parameters as contained in the entity body of the HTTP POST request.

8.5. Information Resource Directory

An ALTO Server uses Information Resource Directory to publish available Information Resources and their aforementioned attributes. Since resource selection happens after consumption of the Information Resource Directory, the format of the Information Resource Directory is designed to be simple with the intention of future ALTO Protocol

versions maintaining backwards compatibility. Future extensions or versions of the ALTO Protocol SHOULD be accomplished by extending existing media types or adding new media types, but retaining the same format for the Information Resource Directory.

An ALTO Server MUST make an Information Resource Directory available via the HTTP GET method to a URI discoverable by an ALTO Client. Discovery of this URI is out of scope of this document, but could be accomplished by manual configuration or by returning the URI of an Information Resource Directory from the ALTO Discovery Protocol [I-D.ietf-alto-server-discovery]. For recommendations on how the URI may look like, see [I-D.ietf-alto-server-discovery].

8.5.1. Media Type

The media type to indicate an information directory is "application/alto-directory+json".

8.5.2. Encoding

An Information Resource Directory is a JSON object of type InfoResourceDirectory:

```
object {
  IRDMeta          meta;
  IRDResourceEntry resources<0..*>;
} InfoResourceDirectory;

object-map {
  JSONString -> JSONValue;
} IRDMeta;

object {
  JSONString      uri;
  JSONString      media-types<1..*>;
  [JSONString     accepts<0..*>;]
  [Capabilities   capabilities;]
} IRDResourceEntry;

object {
  ...
} Capabilities;
```

where the "meta" member provides definitions related with the IRD itself, or can be used when defining multiple individual Information resources;

the "resources" array indicates a list of Information Resources provided by an ALTO Server. Note that the list of available resources is enclosed in a JSON object for extensibility; future protocol versions may specify additional members in the InfoResourceDirectory object.

Each entry specifies:

uri A URI at which the ALTO Server provides one or more Information Resources, or an Information Resource Directory indicating additional Information Resources. URIs can be relative and MUST be resolved according to Section 5 of [RFC3986].

media-types The list of all media types of Information Resources (see Section 8.4.1) available via GET or POST requests to the corresponding URI or URIs discoverable via the URI.

accepts The list of all media types of input parameters (see Section 8.4.3) accepted by POST requests to the corresponding URI or URIs discoverable via the URI. If this member is not present, it MUST be assumed to be an empty array.

capabilities A JSON Object enumerating capabilities of an ALTO Server in providing the Information Resource at the corresponding URI and Information Resources discoverable via the URI. If this member is not present, it MUST be assumed to be an empty object. If a capability for one of the offered Information Resources is not explicitly listed here, an ALTO Client may either issue an OPTIONS HTTP request to the corresponding URI to determine if the capability is supported, or assume its default value documented in this specification or an extension document describing the capability.

If an entry has an empty list for "accepts", then the corresponding URI MUST support GET requests. If an entry has a non-empty list for "accepts", then the corresponding URI MUST support POST requests. If an ALTO Server wishes to support both GET and POST on a single URI, it MUST specify two entries in the Information Resource Directory.

8.5.3. Example

The following is an example Information Resource Directory returned by an ALTO Server.

```
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: TBA
   Content-Type: application/alto-directory+json

   {
     "meta" : {
       "cost-types": {
          "num-routing": {"cost-mode" :  "numerical",
                          "cost-metric": "routingcost",
                          "description": "My default"},
          "num-hop":     {"cost-mode" :  "numerical",
                          "cost-metric": "hopcount"}
          "ord-routing": {"cost-mode" :  "ordinal",
                          "cost-metric": "routingcost"},
          "ord-hop":     {"cost-mode" :  "ordinal",
                          "cost-metric": "hopcount"}
       }
     },
     "resources" : [
       {
         "uri" : "http://alto.example.com/networkmap",
         "media-types" : [ "application/alto-networkmap+json" ]
       }, {
         "uri" : "http://alto.example.com/costmap/num/routingcost",
         "media-types" : [ "application/alto-costmap+json" ],
         "capabilities" : {
           "cost-type-names" : [ "num-routing" ]
         }
       }, {
         "uri" : "http://alto.example.com/costmap/num/hopcount",
         "media-types" : [ "application/alto-costmap+json" ],
         "capabilities" : {
           "cost-type-names" : [ "num-hop" ]
         }
       }, {
         "uri" : "http://custom.alto.example.com/maps",
         "media-types" : [
           "application/alto-networkmap+json",
           "application/alto-costmap+json"
         ],
         "accepts" : [
           "application/alto-networkmapfilter+json",
           "application/alto-costmapfilter+json"
         ]
       }, {
         "uri" : "http://alto.example.com/endpointprop/lookup",
         "media-types" : [ "application/alto-endpointprop+json" ],
         "accepts" : [ "application/alto-endpointpropparams+json" ],
```

```
        "capabilities" : {
          "prop-types" : [ "pid" ]
        }
      }, {
        "uri" : "http://alto.example.com/endpointcost/lookup",
        "media-types" : [ "application/alto-endpointcost+json" ],
        "accepts" : [ "application/alto-endpointcostparams+json" ],
        "capabilities" : {
          "cost-constraints" : true,
          "cost-type-names" : [ "num-routing", "num-hop",
                                "ord-routing", "ord-hop"]
        }
      }
    ]
  }
```

Specifically, the "meta" member of the example IRD defines four cost types so that each Information Resource can use them.

The "resources" array of the example IRD defines six Information Resources. For example, the last entry is to provide the Endpoint Cost Service, which is indicated by the media-type "application/alto-endpointcost+json". An ALTO Client should use uri "http://alto.example.com/endpointcost/lookup" to access the service. The ALTO Client should format its request body to be the "application/alto-endpointcostparams+json" media type, as specified by the "accepts" attribute of the Information Resource. The "cost-type-names" member of the "capabilities" attribute of the Information Resource includes 4 defined cost types from the "meta" member of the IRD. Hence, one can verify that the Endpoint Cost Information Resource supports both Cost Metrics 'routingcost' and 'hopcount', each available for both 'numerical' and 'ordinal'. When requesting the Information Resource, an ALTO Client can specify cost constraints, as indicated by the "cost-constraints" member of the "capabilities" attribute.

8.5.4. Multiple Choices and OPTIONS

ALTO Information Resource Directory provides flexibility to an ALTO Server (e.g., delegation) so that it MAY indicate multiple Information Resources using one URI endpoint. In the example above, the ALTO Server provides additional Network and Cost Maps via a separate subdomain, "custom.alto.example.com". In particular, the maps available via this subdomain are Filtered Network and Cost Maps as well as pre-generated maps for the "hopcount" and "routingcost" Cost Metrics in the "ordinal" Cost Mode.

If an ALTO Client requests one URI that provides multiple Information Resources, the ALTO Server replies with an HTTP 300 status code ("Multiple Choices"). The ALTO Client can discover the specific Information Resources indicated by the URI using an OPTIONS request. The ALTO Server SHOULD use the Information Resource Directory format in its reply to an OPTIONS request.

Consider the preceding example. The ALTO Client can discover the maps available at "custom.alto.example.com" by successfully performing an OPTIONS request to "http://custom.alto.example.com/maps":

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

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

   {
     "meta" : {
       "cost-types": {
          "num-routing": {"cost-mode" :  "numerical",
                          "cost-metric": "routingcost",
                          "description": "My default"},
          "num-hop":     {"cost-mode" :  "numerical",
                          "cost-metric": "hopcount"}
          "ord-routing": {"cost-mode" :  "ordinal",
                          "cost-metric": "routingcost"},
          "ord-hop":     {"cost-mode" :  "ordinal",
                          "cost-metric": "hopcount"}
       }
     },
     "resources" : [
       {
         "uri" : "http://custom.alto.example.com/networkmap/filtered",
         "media-types" : [ "application/alto-networkmap+json" ],
         "accepts" : [ "application/alto-networkmapfilter+json" ]
       }, {
         "uri" : "http://custom.alto.example.com/costmap/filtered",
         "media-types" : [ "application/alto-costmap+json" ],
         "accepts" : [ "application/alto-costmapfilter+json" ],
         "capabilities" : {
           "cost-constraints" : true,
           "cost-type-names" : [ "num-routing", "num-hop",
                                 "ord-routing", "ord-hop" ]
         }
       }, {
         "uri" : "http://custom.alto.example.com/ord/routingcost",
         "media-types" : [ "application/alto-costmap+json" ],
         "capabilities" : {
           "cost-type-names" : [ "ord-routing" ]
         }
       }, {
         "uri" : "http://custom.alto.example.com/ord/hopcount",
         "media-types" : [ "application/alto-costmap+json" ],
         "capabilities" : {
           "cost-type-names" : [ "ord-hop" ],
         }
       }
     ]
   }
```

8.5.5. Usage Considerations

8.5.5.1. ALTO Client

This document specifies no requirements or constraints on ALTO Clients with regards to how they process an Information Resource Directory to identify the URI corresponding to a desired Information Resource. However, some advice is provided for implementors.

It is possible that multiple entries in the directory match a desired Information Resource. For instance, in the example in Section 8.5.3, a full Cost Map with "numerical" Cost Mode and "routingcost" Cost Metric could be retrieved via a GET request to "http://alto.example.com/costmap/num/routingcost", or via a POST request to "http://custom.alto.example.com/costmap/filtered".

In general, it is preferred for ALTO Clients to use GET requests where appropriate, since it is more likely for responses to be cachable.

8.5.5.2. ALTO Server

This document indicates that an ALTO Server may or may not provide the Information Resources specified in the Map Filtering Service. If these resources are not provided, it is indicated to an ALTO Client by the absence of a Network Map or Cost Map with any media types listed under "accepts".

8.6. Information Resource: Content Encoding

Though each Information Resource may have a distinct syntax and hence its unique Media Type, they are designed to have a common structure containing generic ALTO-layer metadata about the resource, as well as data itself.

Specifically, each Information Resource has a single top-level JSON object of type InfoResourceEntity:

```
object {
  InfoResourceMeta       meta;
  InfoResourceDataType   data;
} InfoResourceEntity;
```

with members:

meta meta-information pertaining to the Information Resource;

data the data contained in the Information Resource.

8.6.1. Meta Information

Meta information is encoded as a JSON object. This document does not specify any members, but it is defined here as a standard container for extensibility. Specifically, InfoResourceMetaData is defined as:

```
object-map {
  JSONString -> JSONValue
} InfoResourceMetaData;
```

8.6.2. Data Information

The "data" member of the InfoResourceEntity encodes the resource-specific data. In this document, we define four specific InfoResourceDataType: InfoResourceNetworkMap, InfoResourceCostMap, InfoResourceEndpointProperty, and InfoResourceEndpointCostMap, whose structures will be detailed below.

8.6.3. Example

The following is an example of the encoding for an Information Resource:

```
HTTP/1.1 200 OK
Content-Length: 40
Content-Type: application/alto-costmap+json

{
  "meta" : {},
  "data" : {
    ...
  }
}
```

8.7. Protocol Errors

If there is an error processing a request, an ALTO Server SHOULD return additional ALTO-layer information, if it is available, in the form of an ALTO Error Resource encoded in the HTTP response' entity body. If no ALTO-layer information is available, an ALTO Server may

omit an ALTO Error resource from the response.

With or without additional ALTO-layer error information, an ALTO Server MUST set an appropriate HTTP status code. It is important to note that the HTTP Status Code and ALTO Error Resource have distinct roles. An ALTO Error Resource provides detailed information about why a particular request for an ALTO Resource was not successful. The HTTP status code indicates to HTTP processing elements (e.g., intermediaries and clients) how the response should be treated.

8.7.1. Media Type

The media type for an ALTO Error Resource is "application/alto-error+json".

8.7.2. Resource Format and Error Codes

An ALTO Error Resource has the format:

```
object {
  JSONString code;
} ErrorResourceEntity;
```

where:

code An ALTO Error Code defined in Table 1. Note that the ALTO Error Codes defined in Table 1 are limited to support the error conditions needed for purposes of this document. Additional status codes may be defined in companion or extension documents.

ALTO Error Code	Description
E_SYNTAX	Parsing error in request (including identifiers)
E_JSON_FIELD_MISSING	Required field missing
E_JSON_VALUE_TYPE	JSON Value of unexpected type
E_INVALID_COST_MODE	Invalid cost mode
E_INVALID_COST_METRIC	Invalid cost metric
E_INVALID_PROPERTY_TYPE	Invalid property type

Table 1: Defined ALTO Error Codes.

If multiple errors are present in a single request (e.g., a request uses a JSONString when a JSONNumber is expected and a required field

is missing), then the ALTO Server MUST return exactly one of the detected errors. However, the reported error is implementation defined, since specifying a particular order for message processing encroaches needlessly on implementation technique.

8.7.3. Overload Conditions and Server Unavailability

If an ALTO Server detects that it cannot handle a request from an ALTO Client due to excessive load, technical problems, or system maintenance, it SHOULD do one of the following:

- Return an HTTP 503 ("Service Unavailable") status code to the ALTO Client. As indicated by [RFC2616], a the Retry-After HTTP header may be used to indicate when the ALTO Client should retry the request.

- Return an HTTP 307 ("Temporary Redirect") status code indicating an alternate ALTO Server that may be able to satisfy the request.

The ALTO Server MAY also terminate the connection with the ALTO Client.

The particular policy applied by an ALTO Server to determine that it cannot service a request is outside of the scope of this document.

9. Protocol Specification: Basic ALTO Data Types

This section details the format for particular data values used in the ALTO Protocol.

9.1. PID Name

A PID Name is encoded as a US-ASCII string. The string MUST be no more than 64 characters, and MUST NOT contain any ASCII character below 0x21 or above 0x7E or the '.' separator (0x2E). The '.' separator is reserved for future use and MUST NOT be used unless specifically indicated by a companion or extension document.

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

9.2. Version Tag

A Version Tag is encoded as a case-sensitive US-ASCII string. The string MUST be no more than 64 characters, and MUST NOT contain any ASCII character below 0x21 or above 0x7E.

The type 'VersionTag' is used in this document to indicate a string of this type. Two tags are the same if and only if they are byte for byte equal.

9.3. Endpoints

This section defines formats used to encode addresses for Endpoints. In a case that multiple textual representations encode the same Endpoint address or prefix (within the guidelines outlined in this document), the ALTO Protocol does not require ALTO Clients or ALTO Servers to use a particular textual representation, nor does it require that ALTO Servers reply to requests using the same textual representation used by requesting ALTO Clients. ALTO Clients must be cognizant of this.

9.3.1. Address Type

Address Types are encoded as US-ASCII strings consisting of only alphanumeric characters (code points 0x30-0x39, 0x41-0x5A, and 0x61-0x7A). This document defines the address type 'ipv4' to refer to IPv4 addresses, and 'ipv6' to refer to IPv6 addresses. All Address Type identifiers appearing in an HTTP request or response with an 'application/alto-*' media type MUST be registered in the ALTO Address Type registry (see Section 13.4).

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

9.3.2. Endpoint Address

Endpoint Addresses are encoded as US-ASCII strings. The exact characters and format depend on the type of endpoint address.

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

9.3.2.1. IPv4

IPv4 Endpoint Addresses are encoded as specified by the 'IPv4address' rule in Section 3.2.2 of [RFC3986].

9.3.2.2. IPv6

IPv6 Endpoint Addresses are encoded as specified in Section 4 of [RFC5952].

9.3.2.3. Typed Endpoint Addresses

When an Endpoint Address is used, an ALTO implementation must be able to determine its type. For this purpose, the ALTO Protocol allows endpoint addresses to also explicitly indicate their type.

Typed Endpoint Addresses are encoded as US-ASCII strings of the format 'AddressType:EndpointAddr' (with the ':' character as a separator). The type 'TypedEndpointAddr' is used to indicate a string of this format.

9.3.3. Endpoint Prefixes

For efficiency, it is useful to denote a set of Endpoint Addresses using a special notation (if one exists). This specification makes use of the prefix notations for both IPv4 and IPv6 for this purpose.

Endpoint Prefixes are encoded as US-ASCII strings. The exact characters and format depend on the type of endpoint address.

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

9.3.3.1. IPv4

IPv4 Endpoint Prefixes are encoded as specified in Section 3.1 of [RFC4632].

9.3.3.2. IPv6

IPv6 Endpoint Prefixes are encoded as specified in Section 7 of [RFC5952].

9.3.4. Endpoint Address Group

The ALTO Protocol includes messages that specify potentially large sets of endpoint addresses. Endpoint Address Groups provide a more efficient way to encode such sets, even when the set contains endpoint addresses of different types.

An Endpoint Address Group is defined as:

```
object-map {
  AddressType -> EndpointPrefix<0..*>;
} EndpointAddrGroup;
```

In particular, an Endpoint Address Group is a JSON object representing a map, where each key is the string corresponding to an address type, and the corresponding value is an array listing prefixes of addresses of that type.

The following is an example with both IPv4 and IPv6 endpoint addresses:

```
{
  "ipv4": [
    "192.0.2.0/24",
    "198.51.100.0/25"
  ],
  "ipv6": [
    "2001:db8:0:1::/64",
    "2001:db8:0:2::/64"
  ]
}
```

9.4. Cost Mode

A Cost Mode is encoded as a US-ASCII string. The string MUST either have the value 'numerical' or 'ordinal'.

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

9.5. Cost Metric

A Cost Metric is encoded as a US-ASCII string. The string MUST be no more than 32 characters, and MUST NOT contain characters other than alphanumeric characters (code points 0x30-0x39, 0x41-0x5A, and 0x61-0x7A), the hyphen ('-', code point 0x2D), or the colon (':', code point 0x3A).

Identifiers prefixed with 'priv:' are reserved for Private Use [RFC5226]. Identifiers prefixed with 'exp:' are reserved for Experimental use. For an identifier with the 'priv:' or 'exp:' prefix, an additional string (e.g., company identifier or random string) MUST follow to reduce potential collisions. For example, a short string after 'exp:' to indicate the starting time of a specific experiment is recommended. All other identifiers appearing in an HTTP request or response with an 'application/alto-*' media type MUST be registered in the ALTO Cost Metrics registry Section 13.2.

The type 'CostMetric' is used in this document to indicate a string

of this format.

9.6. Cost Type

The combinaton of CostType and a CostMode defines a CostType:

```
object {
  CostMetric cost-metric;
  CostModel  cost-mode;
  [JSONString description;]
} CostType;
```

'description', if present, MUST contain a US-ASCII string with a human-readable description of the cost-metric and cost-mode. The field SHOULD NOT be interpreted by an ALTO Client.

9.7. Endpoint Property

An Endpoint Property is encoded as a US-ASCII string. The string MUST be no more than 32 characters, and MUST NOT contain characters other than alphanumeric characters (code points 0x30-0x39, 0x41-0x5A, and 0x61-0x7A), the hyphen ('-', code point 0x2D), or the colon (':', code point 0x3A).

Identifiers prefixed with 'priv:' are reserved for Private Use [RFC5226]. Identifiers prefixed with 'exp:' are reserved for Experimental use. For an identifier with the 'priv:' or 'exp:' prefix, an additional string (e.g., company identifier or random string) MUST follow to reduce potential collisions. For example, a short string after 'exp:' to indicate the starting time of a specific experiment is recommended. All other identifiers appearing in an HTTP request or response with an 'application/alto-*' media type MUST be registered in the ALTO Endpoint Property registry Section 13.3.

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

10. Protocol Specification: Service Information Resources

This section documents the individual Information Resources defined to provide the services define in this document.

10.1. Map Service

The Map Service provides batch information to ALTO Clients in the form of two types of maps: a Network Map and Cost Map.

10.1.1. Network Map

The Network Map Information Resource lists for each PID, the network locations (endpoints) within the PID. It MUST be provided by an ALTO Server.

10.1.1.1. Media Type

The media type of Network Map is "application/alto-networkmap+json".

10.1.1.2. HTTP Method

The Network Map resource is requested using the HTTP GET method.

10.1.1.3. Accept Input Parameters

None.

10.1.1.4. Capabilities

None.

10.1.1.5. Response

The "data" member of the returned InfoResourceEntity for a Network Map is an object of type InfoResourceNetworkMap:

```
object {
  VersionTag     map-vtag;
  NetworkMapData map;
} InfoResourceNetworkMap;

object-map {
  PIDName -> EndpointAddrGroup;
} NetworkMapData;
```

with members:

map-vtag The Version Tag (Section 6.3) of the Network Map.

map The Network Map data itself.

NetworkMapData is a JSON object representing a dictionary map with each key representing a single PID, and the value the associated set of endpoint addresses.

The returned Network Map MUST include all PIDs known to the ALTO Server.

10.1.1.6. Example

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

```
HTTP/1.1 200 OK
Content-Length: 370
Content-Type: application/alto-networkmap+json

{
  "meta" : {},
  "data" : {
    "map-vtag" : "1266506139",
    "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"
        ]
      }
    }
  }
}
```

The encodings were chosen for readability and compactness. If lookup efficiency at runtime is crucial, then the returned Network Map and Cost Map can be transformed into data structures offering more efficient lookup, such as transforming the Network Map into a IP trie for longest-prefix matching and the Cost Map into a matrix.

10.1.2. Cost Map

The Cost Map resource lists the Path Cost for each pair of source/ destination PID defined by the ALTO Server for a given Cost Metric and Cost Mode. This resource MUST be provided for at least the 'routingcost' Cost Metric.

10.1.2.1. Media Type

The media type of Cost Map is "application/alto-costmap+json".

10.1.2.2. HTTP Method

The Cost Map resource is requested using the HTTP GET method.

10.1.2.3. Accept Input Parameters

None.

10.1.2.4. Capabilities

The capabilities of an ALTO Server URI providing an unfiltered cost map is a JSON Object of type CostMapCapabilities:

```
object {
  JSONString cost-type-names<1..*>;
} CostMapCapabilities;
```

with member:

cost-type-names A sequence of CostType names defined in "cost-types" of the "meta" member of an IRD. These represent the Cost Types that are supported via the corresponding URI in the IRD. If there is more than one Cost Type in this list, then the ALTO Server SHOULD return an IRD to the client to lead it towards the URIs for the corresponding Cost Maps. Since an unfiltered Cost Map is requested via an HTTP GET that accepts no input parameters, an ALTO Client MUST be led towards a resource that has a single element in the 'cost-type-names' list.

10.1.2.5. Response

The "data" member of the returned InfoResourceEntity for a Cost Map is an object of type InfoResourceCostMap:

```
object {
  CostType    cost-type;
  VersionTag  map-vtag;
  CostMapData map;
} InfoResourceCostMap;

object-map {
  PIDName -> DstCosts;
} CostMapData;

object-map {
  PIDName -> JSONValue;
} DstCosts;
```

with members:

cost-type Cost Type (Section 9.6) used in the Cost Map.

map-vtag The Version Tag (Section 6.3) of the Network Map used to generate the Cost Map.

map The Cost Map data itself.

CostMapData is a dictionary map object, with each key being the PIDName string identifying the corresponding Source PID, and value being a type of DstCosts, which denotes the associated costs from the Source PID to a set of destination PIDs (Section 6.2). An implementation of the protocol in this document SHOULD assume that the cost is a JSONNumber and fail to parse if it is not, unless the implementation is using an extension to this document that indicates when and how costs of other data types are signaled.

The returned Cost Map MUST include the Path Cost for each (Source PID, Destination PID) pair for which a Path Cost is defined. An ALTO Server MAY omit entries for which a Path Cost is not defined (e.g., both the Source and Destination PIDs contain addresses outside of the Network Provider's administrative domain).

10.1.2.6. Example

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

```
HTTP/1.1 200 OK
Content-Length: TBA
Content-Type: application/alto-costmap+json

{
  "meta" : {},
  "data" : {
    "cost-type" : {"cost-mode"  : "numerical",
                   "cost-metric": "routingcost"},
    "map-vtag"  : "1266506139",
    "map" : {
      "PID1": { "PID1": 1,  "PID2": 5,  "PID3": 10 },
      "PID2": { "PID1": 5,  "PID2": 1,  "PID3": 15 },
      "PID3": { "PID1": 20, "PID2": 15  }
    }
  }
}
```

Similar to the Network Map case, we considered array-based encoding for "map", but chose the current encoding for clarity.

10.2. Map Filtering Service

The Map Filtering Service allows ALTO Clients to specify filtering criteria to return a subset of the full maps available in the Map Service.

10.2.1. Filtered Network Map

A Filtered Network Map is a Network Map Information Resource (Section 10.1.1) for which an ALTO Client may supply a list of PIDs to be included. A Filtered Network Map MAY be provided by an ALTO Server.

10.2.1.1. Media Type

As a Filtered Network Map is a Network Map, it uses the media type defined for Network Map at Section 10.1.1.1.

10.2.1.2. HTTP Method

A Filtered Network Map is requested using the HTTP POST method.

10.2.1.3. Accept Input Parameters

An ALTO Client supplies filtering parameters by specifying media type "application/alto-networkmapfilter+json" with HTTP POST body

containing a JSON Object of type ReqFilteredNetworkMap, where:

```
object {
  PIDName pids<0..*>;
  AddressType address-types<0..*>;
} ReqFilteredNetworkMap;
```

with members:

pids Specifies list of PIDs to be included in the returned Filtered Network Map. If the list of PIDs is empty, the ALTO Server MUST interpret the list as if it contained a list of all currently-defined PIDs. The ALTO Server MUST interpret entries appearing multiple times as if they appeared only once.

address-types Specifies list of address types to be included in the returned Filtered Network Map. If the list of address types is empty, the ALTO Server MUST interpret the list as if it contained a list of all address types known to the ALTO Server. The ALTO Server MUST interpret entries appearing multiple times as if they appeared only once.

10.2.1.4. Capabilities

None.

10.2.1.5. Response

See Section 10.1.1.5 for the format.

The ALTO Server MUST only include PIDs in the response that were specified (implicitly or explicitly) in the request. If the input parameters contain a PID name that is not currently defined by the ALTO Server, the ALTO Server MUST behave as if the PID did not appear in the input parameters. Similarly, the ALTO Server MUST only enumerate addresses within each PID that have types which were specified (implicitly or explicitly) in the request. If the input parameters contain an address type that is not currently known to the ALTO Server, the ALTO Server MUST behave as if the address type did not appear in the input parameters.

10.2.1.6. Example

```
POST /networkmap/filtered HTTP/1.1
Host: custom.alto.example.com
Content-Length: 27
Content-Type: application/alto-networkmapfilter+json
Accept: application/alto-networkmap+json,application/alto-error+json

{
  "pids": [ "PID1", "PID2" ]
}
```

```
HTTP/1.1 200 OK
Content-Length: 255
Content-Type: application/alto-networkmap+json

{
  "meta" : {},
  "data" : {
    "map-vtag" : "1266506139",
    "map" : {
      "PID1" : {
        "ipv4" : [
          "192.0.2.0/24",
          "198.51.100.0/24"
        ]
      },
      "PID2" : {
        "ipv4": [
          "198.51.100.128/24"
        ]
      }
    }
  }
}
```

10.2.2. Filtered Cost Map

A Filtered Cost Map is a Cost Map Information Resource (Section 10.1.2) for which an ALTO Client may supply additional parameters limiting the scope of the resulting Cost Map. A Filtered Cost Map MAY be provided by an ALTO Server.

10.2.2.1. Media Type

As a Filtered Cost Map is a Cost Map, it uses the media type defined for Cost Map at Section 10.1.2.1.

10.2.2.2. HTTP Method

A Filtered Cost Map is requested using the HTTP POST method.

10.2.2.3. Accept Input Parameters

The input parameters for a Filtered Map 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-costmapfilter+json", which is a JSON Object of type ReqFilteredCostMap, where:

```
object {
  CostType   cost-type;
  [JSONString constraints<0..*>;]
  [PIDFilter  pids;]
} ReqFilteredCostMap;

object {
  PIDName srcs<0..*>;
  PIDName dsts<0..*>;
} PIDFilter;
```

with members:

cost-type The CostType (Section 9.6) for the returned costs. This MUST be one of the supported Cost Types indicated in this resource's capabilities (Section 10.2.2.4).

constraints Defines a list of additional constraints on which elements of the Cost Map are returned. This parameter MUST NOT be specified if this resource's capabilities (Section 10.2.2.4) indicate that constraint support is not available. A constraint contains two entities separated by whitespace: (1) an operator, 'gt' for greater than, 'lt' for less than, 'ge' for greater than or equal to, 'le' for less than or equal to, or 'eq' for equal to; (2) a target cost value. The cost value is a number that MUST be defined in the same units as the Cost Metric indicated by the cost-metric parameter. ALTO Servers SHOULD use at least IEEE 754 double-precision floating point [IEEE.754.2008] to store the cost value, and SHOULD perform internal computations using double-

precision floating-point arithmetic. If multiple 'constraint' parameters are specified, they are interpreted as being related to each other with a logical AND.

pids A list of Source PIDs and a list of Destination PIDs for which Path Costs are to be returned. If a list is empty, the ALTO Server MUST interpret it as the full set of currently-defined PIDs. The ALTO Server MUST interpret entries appearing in a list multiple times as if they appeared only once. If the "pids" member is not present, both lists MUST be interpreted by the ALTO Server as containing the full set of currently-defined PIDs.

10.2.2.4. Capabilities

The URI providing this resource supports all capabilities documented in Section 10.1.2.4 (with identical semantics), plus additional capabilities. In particular, the capabilities are defined by a JSON object of type FilteredCostMapCapabilities:

```
object {
  JSONString cost-type-names<1..*>;
  JSONBool cost-constraints;
} FilteredCostMapCapabilities;
```

with members:

cost-type-names See Section 10.1.2.4 and note that the array can have 1 to many cost types.

cost-constraints If true, then the ALTO Server allows cost constraints to be included in requests to the corresponding URI. If not present, this member MUST be interpreted as if it specified false. ALTO Clients should be aware that constraints may not have the intended effect for cost maps with the 'ordinal' Cost Mode since ordinal costs are not restricted to being sequential integers.

10.2.2.5. Response

See Section 10.1.2.5 for the format.

The returned Cost Map MUST contain only source/destination pairs that have been indicated (implicitly or explicitly) in the input parameters. If the input parameters contain a PID name that is not currently defined by the ALTO Server, the ALTO Server MUST behave as if the PID did not appear in the input parameters.

If any constraints are specified, Source/Destination pairs for which the Path Costs do not meet the constraints MUST NOT be included in the returned Cost Map. If no constraints were specified, then all Path Costs are assumed to meet the constraints.

Note that ALTO Clients should verify that the Version Tag included in the response is consistent with the Version Tag of the Network Map used to generate the request (if applicable). If it is not, the ALTO Client may wish to request an updated Network Map, identify changes, and consider requesting a new Filtered Cost Map.

10.2.2.6. Example

```
POST /costmap/filtered HTTP/1.1
Host: custom.alto.example.com
Content-Type: application/alto-costmapfilter+json
Accept: application/alto-costmap+json,application/alto-error+json

{
  "cost-type" : {"cost-mode": "numerical",
                 "cost-metric": "routingcost"},
  "pids" : {
    "srcs" : [ "PID1" ],
    "dsts" : [ "PID1", "PID2", "PID3" ]
  }
}
```

```
HTTP/1.1 200 OK
Content-Length: 177
Content-Type: application/alto-costmap+json

{
  "meta" : {},
  "data" : {
    "cost-type": {"cost-mode" : "numerical",
                  "cost-metric" : "routingcost"},
    "map-vtag" : "1266506139",
    "map" : {
      "PID1": { "PID1": 0,  "PID2": 1,  "PID3": 2 }
    }
  }
}
```

10.3. Endpoint Property Service

The Endpoint Property Service provides information about Endpoint properties to ALTO Clients.

10.3.1. Endpoint Property

The Endpoint Property resource provides information about properties for individual endpoints. It MAY be provided by an ALTO Server. If an ALTO Server provides one or more Endpoint Property resources, then at least one MUST provide the 'pid' property.

10.3.1.1. Media Type

The media type of Endpoint Property is "application/alto-endpointprop+json".

10.3.1.2. HTTP Method

The Endpoint Property resource is requested using the HTTP POST method.

10.3.1.3. Accept Input Parameters

An ALTO Client supplies the endpoint properties to be queried through a media type "application/alto-endpointpropparams+json", and specifies in the HTTP POST entity body a JSON Object of type ReqEndpointProp:

```
object {
  EndpointPropertyType  properties<1..*>;
  TypedEndpointAddr     endpoints<1..*>;
} ReqEndpointProp;
```

with members:

properties List of endpoint properties to be returned for each endpoint. Each specified property MUST be included in the list of supported properties indicated by this resource's capabilities (Section 10.3.1.4). The ALTO Server MUST interpret entries appearing multiple times as if they appeared only once.

endpoints List of endpoint 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.

10.3.1.4. Capabilities

This resource may be defined across multiple types of endpoint properties. The capabilities of an ALTO Server URI providing Endpoint Properties are defined by a JSON Object of type EndpointPropertyCapabilities:

```
object {
  EndpointPropertyType prop-types<0..*>;
} EndpointPropertyCapabilities;
```

with members:

prop-types The Endpoint Properties (see Section 9.7) supported by the corresponding URI. If not present, this member MUST be interpreted as an empty array.

10.3.1.5. Response

The returned InfoResourceEntity object has "data" member of type InfoResourceEndpointProperty, where:

```
object {
  VersionTag              map-vtag; [DEPEND ON PROPERTIES]
  EndpointPropertyMapData map;
} InfoResourceEndpointProperty;

object-map {
  TypedEndpointAddr -> EndpointProps;
} EndpointPropertyMapData;

object {
  EndpointPropertyType -> JSONValue;
} EndpointProps;
```

EndpointPropertyMapData has one member for each endpoint indicated in the input parameters (with the name being the endpoint encoded as a TypedEndpointAddr). The requested properties for each endpoint are encoded in a corresponding EndpointProps object, which encodes one name/value pair for each requested property, where the property names are encoded as strings of type EndpointPropertyType. An implementation of the protocol in this document SHOULD assume that the property value is a JSONString 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.

The ALTO Server returns the value for each of the requested endpoint properties for each of the endpoints listed in the input parameters.

If the ALTO Server does not define a requested property's value for a particular endpoint, then it MUST omit that property from the response for only that endpoint.

The ALTO Server MAY include the Version Tag (Section 6.3) of the Network Map used to generate the response (if desired and applicable) as the 'map-vtag' member in the response. If the 'pid' property is returned for any endpoints in the response, the 'map-vtag' member is REQUIRED. Otherwise, it is OPTIONAL.

10.3.1.6. Example

```
POST /endpointprop/lookup HTTP/1.1
Host: alto.example.com
Content-Length: 96
Content-Type: application/alto-endpointpropparams+json
Accept: application/alto-endpointprop+json,application/alto-error+json

{
  "properties" : [ "pid", "example-prop" ],
  "endpoints" : [ "ipv4:192.0.2.34", "ipv4:203.0.113.129" ]
}


HTTP/1.1 200 OK
Content-Length: 149
Content-Type: application/alto-endpointprop+json

{
  "meta" : {},
  "data": {
    "map-vtag" : "1266506139",
    "map" : {
      "ipv4:192.0.2.34"    : { "pid": "PID1", "example-prop": "1" },
      "ipv4:203.0.113.129" : { "pid": "PID3" }
    }
  }
}
```

10.4. Endpoint Cost Service

The Endpoint Cost Service provides information about costs between individual endpoints.

In particular, this service allows lists of Endpoint prefixes (and addresses, as a special case) to be ranked (ordered) by an ALTO Server.

10.4.1. Endpoint Cost

The Endpoint Cost resource provides information about costs between individual endpoints. It MAY be provided by an ALTO Server.

It is important to note that although this resource allows an ALTO Server to reveal costs between individual endpoints, an ALTO Server is not required to do so. A simple alternative would be to compute the cost between two endpoints as the cost between the PIDs corresponding to the endpoints. See Section 14.3 for additional details.

10.4.1.1. Media Type

The media type of Endpoint Cost is "application/alto-endpointcost+json".

10.4.1.2. HTTP Method

The Endpoint Cost resource is requested using the HTTP POST method.

10.4.1.3. Accept Input Parameters

An ALTO Client supplies the endpoint cost parameters through a media type "application/alto-endpointcostparams+json", with an HTTP POST entity body of a JSON Object of type ReqEndpointCostMap:

```
object {
  CostType          cost-type;
  [JSONString       constraints<0..*>;]
  EndpointFilter    endpoints;
} ReqEndpointCostMap;

object {
  [TypedEndpointAddr srcs<0..*>;]
  TypedEndpointAddr dsts<1..*>;
} EndpointFilter;
```

with members:

cost-type The Cost Type (Section 9.6) to use for returned costs. This MUST be one of the CostType indicated in this resource's capabilities (Section 10.4.1.4).

constraints Defined equivalently to the "constraints" input parameter of a Filtered Cost Map (see Section 10.2.2).

endpoints A list of Source Endpoints and Destination Endpoints for which Path Costs are to be returned. If the list of Source Endpoints is empty (or not included), the ALTO Server MUST interpret it as if it contained the Endpoint Address corresponding to the client IP address from the incoming connection (see Section 12.3 for discussion and considerations regarding this mode). The list of destination Endpoints MUST NOT be empty. The ALTO Server MUST interpret entries appearing multiple times in a list as if they appeared only once.

10.4.1.4. Capabilities

In this document, we define EndpointCostCapabilities the same as FilteredCostMapCapabilities. See Section 10.2.2.4.

10.4.1.5. Response

The returned InfoResourceEntity object has "data" member equal to InfoResourceEndpointCostMap, where:

```
object {
  CostType            cost-type;
  EndpointCostMapData map;
} InfoResourceEndpointCostMap;

object-map {
  TypedEndpointAddr -> EndpointDstCosts;
} EndpointCostMapData;

object-map {
  TypedEndpointAddr -> JSONValue;
} EndpointDstCosts;
```

InfoResourceEndpointCostMap has members:

cost-type The Cost Type used in the returned Cost Map.

map The Endpoint Cost Map data itself.

EndpointCostMapData is a dictionary map object with each key representing a TypedEndpointAddr string identifying the Source Endpoint specified in the input parameters; the name for a member is. For each Source Endpoint, a EndpointDstCosts dictionary map object denotes the associated cost to each Destination Endpoint specified in input parameters. An implementation of the protocol in this document SHOULD assume that the cost value is a JSONNumber and fail to parse if it is not, unless the implementation is using an extension to this document that indicates when and how costs of other data types are signaled. If the ALTO Server does not define a cost value from a Source Endpoint to a particular Destination Endpoint, it MAY be omitted from the response.

10.4.1.6. Example

```
POST /endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: 195
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

{
  "cost-type": {"cost-mode" : "ordinal",
                "cost-metric" : "routingcost"},
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv4:203.0.113.45"
    ]
  }
}
```

```
HTTP/1.1 200 OK
Content-Length: 231
Content-Type: application/alto-endpointcost+json

{
  "meta" : {},
  "data" : {
    "cost-type": {"cost-mode" : "ordinal",
                  "cost-metric" : "routingcost"},
    "map" : {
      "ipv4:192.0.2.2": {
        "ipv4:192.0.2.89"    : 1,
        "ipv4:198.51.100.34" : 2,
        "ipv4:203.0.113.45"  : 3
      }
    }
  }
}
```

11. Use Cases

The sections below depict typical use cases. While these use cases focus on peer-to-peer applications, ALTO can be applied to other

environments such as CDNs [I-D.jenkins-alto-cdn-use-cases].

11.1. ALTO Client Embedded in P2P Tracker

Many currently-deployed P2P systems use a Tracker to manage swarms and perform peer selection. Such a P2P Tracker can already use a variety of information to perform peer selection to meet application-specific goals. By acting as an ALTO Client, the P2P Tracker can use ALTO information as an additional information source to enable more network-efficient traffic patterns and improve application performance.

A particular requirement of many P2P trackers is that they must handle a large number of P2P clients. A P2P tracker can obtain and locally store ALTO information (the Network Map and Cost Map) from the ISPs containing the P2P clients, and benefit from the same aggregation of network locations done by ALTO Servers.

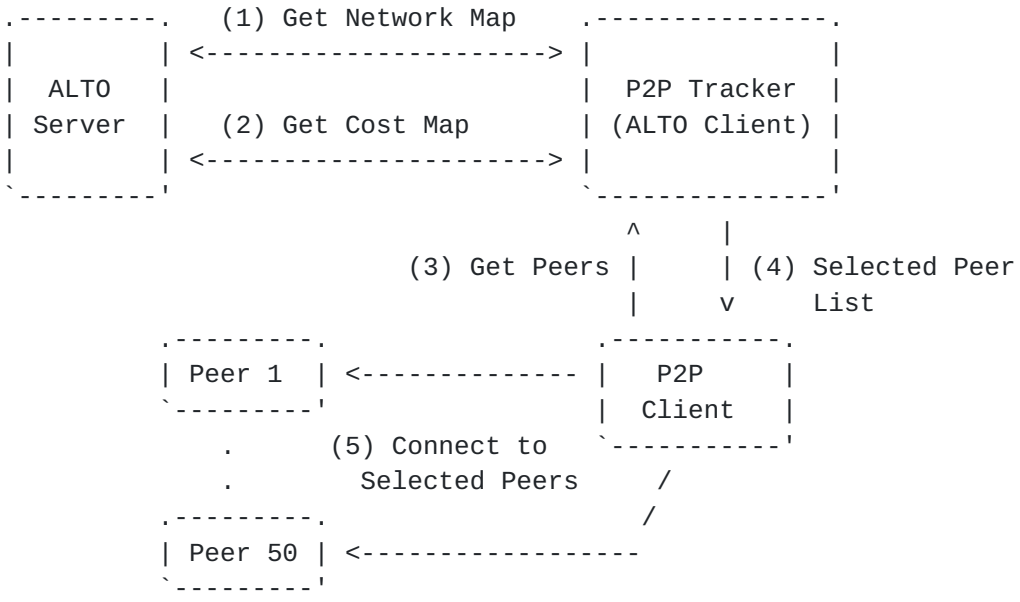

Figure 4: ALTO Client Embedded in P2P Tracker

Figure 4 shows an example use case where a P2P tracker is an ALTO Client and applies ALTO information when selecting peers for its P2P clients. The example proceeds as follows:

1. The P2P Tracker requests from the ALTO Server using the Network Map query the Network Map covering all PIDs. The Network Map includes the IP prefixes contained in each PID, allowing the P2P tracker to locally map P2P clients into PIDs.

2. The P2P Tracker requests from the ALTO Server the Cost Map amongst all PIDs identified in the preceding step.

3. A P2P Client joins the swarm, and requests a peer list from the P2P Tracker.

4. The P2P Tracker returns a peer list to the P2P client. The returned peer list is computed based on the Network Map and Cost Map returned by the ALTO Server, and possibly other information sources. Note that it is possible that a tracker may use only the Network Map to implement hierarchical peer selection by preferring peers within the same PID and ISP.

5. The P2P Client connects to the selected peers.

Note that the P2P tracker may provide peer lists to P2P clients distributed across multiple ISPs. In such a case, the P2P tracker may communicate with multiple ALTO Servers.

11.2. ALTO Client Embedded in P2P Client: Numerical Costs

P2P clients may also utilize ALTO information themselves when selecting from available peers. It is important to note that not all P2P systems use a P2P tracker for peer discovery and selection. Furthermore, even when a P2P tracker is used, the P2P clients may rely on other sources, such as peer exchange and DHTs, to discover peers.

When an P2P Client uses ALTO information, it typically queries only the ALTO Server servicing its own ISP. The my-Internet view provided by its ISP's ALTO Server can include preferences to all potential peers.

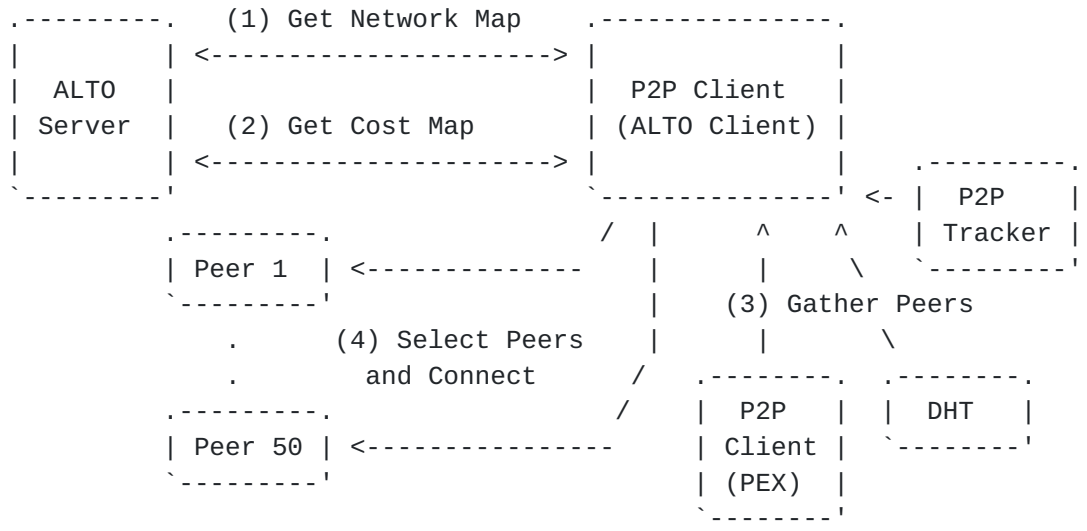

Figure 5: ALTO Client Embedded in P2P Client

Figure 5 shows an example use case where a P2P Client locally applies ALTO information to select peers. The use case proceeds as follows:

1. The P2P Client requests the Network Map covering all PIDs from the ALTO Server servicing its own ISP.

2. The P2P Client requests the Cost Map amongst all PIDs from the ALTO Server. The Cost Map by default specifies numerical costs.

3. The P2P Client discovers peers from sources such as Peer Exchange (PEX) from other P2P Clients, Distributed Hash Tables (DHT), and P2P Trackers.

4. The P2P Client uses ALTO information as part of the algorithm for selecting new peers, and connects to the selected peers.

11.3. ALTO Client Embedded in P2P Client: Ranking

It is also possible for a P2P Client to offload the selection and ranking process to an ALTO Server. In this use case, the ALTO Client gathers a list of known peers in the swarm, and asks the ALTO Server to rank them.

As in the use case using numerical costs, the P2P Client typically only queries the ALTO Server servicing its own ISP.

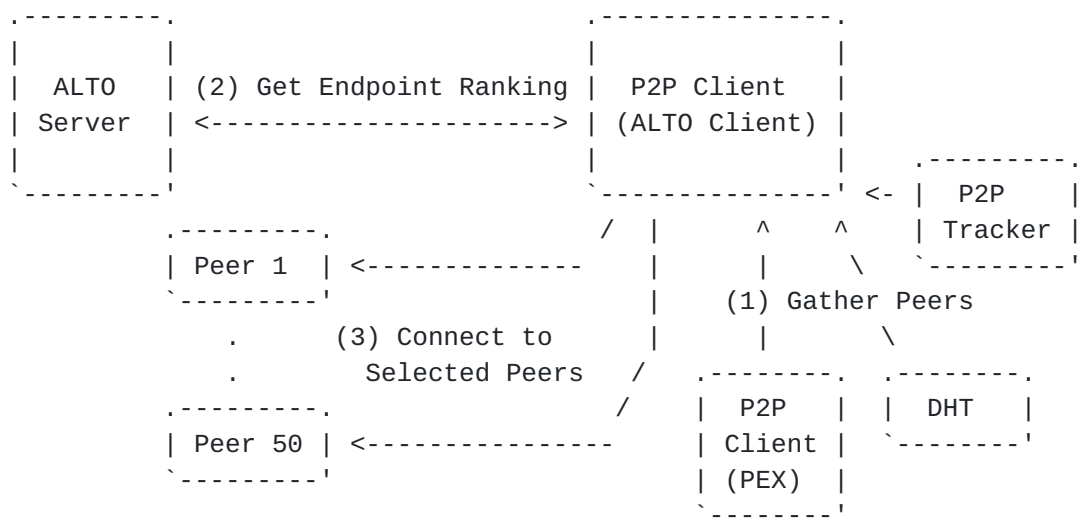

Figure 6: ALTO Client Embedded in P2P Client: Ranking

Figure 6 shows an example of this scenario. The use case proceeds as follows:

1. The P2P Client discovers peers from sources such as Peer Exchange (PEX) from other P2P Clients, Distributed Hash Tables (DHT), and P2P Trackers.

2. The P2P Client queries the ALTO Server's Ranking Service, including discovered peers as the set of Destination Endpoints, and indicates the 'ordinal' Cost Mode. The response indicates the ranking of the candidate peers.

3. The P2P Client connects to the peers in the order specified in the ranking.

12. Discussions

12.1. Discovery

The discovery mechanism by which an ALTO Client locates an appropriate ALTO Server is out of scope for this document. This document assumes that an ALTO Client can discover an appropriate ALTO Server. Once it has done so, the ALTO Client may use the Information Resource Directory (see Section 8.5) to locate an Information Resource with the desired ALTO Information.

12.2. Hosts with Multiple Endpoint Addresses

In practical deployments, a particular host can be reachable using multiple addresses (e.g., a wireless IPv4 connection, a wireline IPv4 connection, and a wireline IPv6 connection). In general, the particular network path followed when sending packets to the host will depend on the address that is used. Network providers may prefer one path over another. An additional consideration may be how to handle private address spaces (e.g., behind carrier-grade NATs).

To support such behavior, this document allows multiple endpoint addresses and address types. With this support, the ALTO Protocol allows an ALTO Service Provider the flexibility to indicate preferences for paths from an endpoint address of one type to an endpoint address of a different type.

12.3. Network Address Translation Considerations

At this day and age of NAT v4<->v4, v4<->v6 [RFC6144], and possibly v6<->v6[I-D.mrw-nat66], a protocol should strive to be NAT friendly and minimize carrying IP addresses in the payload, or provide a mode of operation where the source IP address provide the information necessary to the server.

The protocol specified in this document provides a mode of operation where the source network location is computed by the ALTO Server (i.e., the the Endpoint Cost Service) from the source IP address found in the ALTO Client query packets. This is similar to how some P2P Trackers (e.g., BitTorrent Trackers - see "Tracker HTTP/HTTPS Protocol" in [BitTorrent]) operate.

There may be cases where an ALTO Client needs to determine its own IP address, such as when specifying a source Endpoint Address in the Endpoint Cost Service. It is possible that an ALTO Client has multiple network interface addresses, and that some or all of them may require NAT for connectivity to the public Internet.

If a public IP address is required for a network interface, the ALTO client SHOULD use the Session Traversal Utilities for NAT (STUN) [RFC5389]. If using this method, the host MUST use the "Binding Request" message and the resulting "XOR-MAPPED-ADDRESS" parameter that is returned in the response. Using STUN requires cooperation from a publicly accessible STUN server. Thus, the ALTO client also requires configuration information that identifies the STUN server, or a domain name that can be used for STUN server discovery. To be selected for this purpose, the STUN server needs to provide the public reflexive transport address of the host.

ALTO Clients should be cognizant that the network path between Endpoints can depend on multiple factors, e.g., source address, and destination address used for communication. An ALTO Server provides information based on Endpoint Addresses (more generally, Network Locations), but the mechanisms used for determining existence of connectivity or usage of NAT between Endpoints are out of scope of this document.

12.4. Endpoint and Path Properties

An ALTO Server could make available many properties about Endpoints beyond their network location or grouping. For example, connection type, geographical location, and others may be useful to applications. This specification focuses on network location and grouping, but the protocol may be extended to handle other Endpoint properties.

13. IANA Considerations

13.1. application/alto-* Media Types

This document requests the registration of multiple media types, listed in Table 2.

Type	Subtype	Specification
application	alto-directory+json	Section 8.5
application	alto-networkmap+json	Section 10.1.1
application	alto-networkmapfilter+json	Section 10.2.1
application	alto-costmap+json	Section 10.1.2
application	alto-costmapfilter+json	Section 10.2.2
application	alto-endpointprop+json	Section 10.3.1
application	alto-endpointpropparams+json	Section 10.3.1
application	alto-endpointcost+json	Section 10.4.1
application	alto-endpointcostparams+json	Section 10.4.1
application	alto-error+json	Section 8.7

Table 2: ALTO Protocol Media Types.

Type name: application

Subtype name: This documents requests the registration of multiple subtypes, as listed in Table 2.

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: Encoding considerations are identical to those specified for the 'application/json' media type. See [RFC4627].

Security considerations: Security considerations relating to the generation and consumption of ALTO protocol messages are discussed in Section 14.

Interoperability considerations: This document specifies format of conforming messages and the interpretation thereof.

Published specification: This document is the specification for these media types; see Table 2 for the section documenting each media type.

Applications that use this media type: ALTO Servers and ALTO Clients either standalone or 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).

13.2. ALTO Cost Metric Registry

This document requests the creation of an ALTO Cost Metric registry, listed in Table 3, to be maintained by IANA.

Identifier	Intended Semantics
routingcost	See Section 6.1.1.1
priv:	Private use
exp:	Experimental use

Table 3: ALTO Cost Metrics.

This registry serves two purposes. First, it ensures uniqueness of identifiers referring to ALTO Cost Metrics. Second, it provides references to particular semantics of allocated Cost Metrics to be applied by both ALTO Servers and applications utilizing ALTO Clients.

New ALTO Cost Metrics are assigned after Expert Review [RFC5226]. The Expert Reviewer will generally consult the ALTO Working Group or its successor. Expert Review is used to ensure that proper documentation regarding ALTO Cost Metric semantics and security considerations has been provided. The provided documentation should be detailed enough to provide guidance to both ALTO Service Providers and applications utilizing ALTO Clients as to how values of the registered ALTO Cost Metric should be interpreted. Updates and deletions of ALTO Cost Metrics follow the same procedure.

Registered ALTO Cost Metric identifiers MUST conform to the syntactical requirements specified in Section 9.5. Identifiers are to be recorded and displayed as ASCII strings.

Identifiers prefixed with 'priv:' are reserved for Private Use. Identifiers prefixed with 'exp:' are reserved for Experimental use.

Requests to add a new value to the registry MUST include the following information:

- Identifier: The name of the desired ALTO Cost Metric.

- Intended Semantics: ALTO Costs carry with them semantics to guide their usage by ALTO Clients. For example, if a value refers to a measurement, the measurement units must be documented. For proper implementation of the ordinal Cost Mode (e.g., by a third-party service), it should be documented whether higher or lower values of the cost are more preferred.

- Security Considerations: ALTO Costs expose information to ALTO Clients. As such, proper usage of a particular Cost Metric may require certain information to be exposed by an ALTO Service Provider. Since network information is frequently regarded as

proprietary or confidential, ALTO Service Providers should be made aware of the security ramifications related to usage of a Cost Metric.

This specification requests registration of the identifier 'routingcost'. Semantics for the this Cost Metric are documented in Section 6.1.1.1, and security considerations are documented in Section 14.3.

13.3. ALTO Endpoint Property Type Registry

This document requests the creation of an ALTO Endpoint Property Types registry, listed in Table 4, to be maintained by IANA.

Identifier	Intended Semantics
pid	See Section 7.1.1
priv:	Private use
exp:	Experimental use

Table 4: ALTO Endpoint Property Types.

The maintenance of this registry is similar to that of the preceding ALTO Cost Metrics.

13.4. ALTO Address Type Registry

This document requests the creation of an ALTO Address Type registry, listed in Table 5, to be maintained by IANA.

Identifier	Address Encoding	Prefix Encoding	Mapping to/from IPv4/v6
ipv4	See Section 9.3.2	See Section 9.3.3	Direct mapping to IPv4
ipv6	See Section 9.3.2	See Section 9.3.3	Direct mapping to IPv6

Table 5: ALTO Address Types.

This registry serves two purposes. First, it ensures uniqueness of identifiers referring to ALTO Address Types. Second, it states the requirements for allocated Address Type identifiers.

New ALTO Address Types are assigned after Expert Review [RFC5226]. The Expert Reviewer will generally consult the ALTO Working Group or its successor. Expert Review is used to ensure that proper documentation regarding the new ALTO Address Types and their security considerations has been provided. The provided documentation should indicate how an address of a registered type is encoded as an EndpointAddr and, if possible, a compact method (e.g., IPv4 and IPv6 prefixes) for encoding a set of addresses as an EndpointPrefix. Updates and deletions of ALTO Address Types follow the same procedure.

Registered ALTO Address Type identifiers MUST conform to the syntactical requirements specified in Section 9.3.1. Identifiers are to be recorded and displayed as ASCII strings.

Requests to add a new value to the registry MUST include the following information:

- Identifier: The name of the desired ALTO Address Type.
- Endpoint Address Encoding: The procedure for encoding an address of the registered type as an EndpointAddr (see Section 9.3.2).
- Endpoint Prefix Encoding: The procedure for encoding a set of addresses of the registered type as an EndpointPrefix (see Section 9.3.3). If no such compact encoding is available, the same encoding used for a singular address may be used. In such a case, it must be documented that sets of addresses of this type always have exactly one element.
- Mapping to/from IPv4/IPv6 Addresses: If possible, a mechanism to map addresses of the registered type to and from IPv4 or IPv6 addresses should be specified.
- Security Considerations: In some usage scenarios, Endpoint 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' and 'ipv6', as shown in Table 5.

13.5. ALTO Error Code Registry

This document requests the creation of an ALTO Error Code registry, listed in Table 1, to be maintained by IANA.

14. Security Considerations

Some environments and use cases of ALTO require consideration of security attacks on ALTO Servers and Clients. In order to support those environments interoperably, the ALTO requirements document <xref target="RFC6708"/> outlines minimum-to-implement authentication and other security requirements. Below we consider the threats and protection strategies.

14.1. Authenticity and Integrity of ALTO Information

14.1.1. Risk Scenarios

An attacker may want to provide false or modified ALTO Information Resources or Information Resource Directory to ALTO Clients to achieve certain malicious goals. As an example, an attacker may provide false endpoint properties. For example, suppose that a network supports an endpoint property named hasQuota which reports if the endpoint has usage quota. An attacker may want to generate a false reply to lead to unexpected charges to the endpoint. An attack may also want to provide false Cost Map. For example, by faking a Cost Map that highly prefers a small address range or a single address, the attacker may be able to turn a distributed application into a Distributed Denial of Service (DDoS) tool.

Depending on the network scenario, an attacker can attack authenticity and integrity of ALTO Information Resources using various techniques, including, but not limited to, sending forged DHCP replies in an Ethernet, DNS poisoning, and installing a transparent HTTP proxy that does some modifications.

14.1.2. Protection Strategies

ALTO protects the authenticity and integrity of ALTO Information (both Information Directory and individual Information Resources) by leveraging the authenticity and integrity mechanisms in TLS. In particular, the ALTO Protocol requires that HTTP over TLS <xref target="RFC2818"/> MUST be supported, when protecting the authenticity and integrity of ALTO Information is required. The rules in <xref target="RFC2818"/> for a client to verify server identity using server certificates MUST be supported. ALTO Providers who request server certificates and certification authorities who issue ALTO-specific certificates SHOULD consider the recommendations and guidelines defined in <xref target="RFC6125"/>.

Software engineers developing and service providers deploying ALTO with should make themselves familar with up-to-date Best Current Practices on configuring HTTP over TLS.

14.1.3. Limitations

The protection of HTTP over TLS for ALTO depends on that the domain name in the URI for the Information Resources is not comprised. This will depend on the protection implemented by service discovery.

A deployment scenario may require redistribution of ALTO information to improve scalability. When authenticity and integrity of ALTO information are still required, then ALTO Clients obtaining ALTO information through redistribution must be able to validate the received ALTO information. Support for this validation is not provided in this document, but may be provided by extension documents.

14.2. Potential Undesirable Guidance from Authenticated ALTO Information

14.2.1. Risk Scenarios

The ALTO Service makes it possible for an ALTO Provider to influence the behavior of network applications. An ALTO Provider may be hostile to some applications and hence try to use ALTO Information Resources to achieve certain goals <xref target="RFC5693"/>: "redirecting applications to corrupted mediators providing malicious content, or applying policies in computing Cost Map based on criteria other than network efficiency." See <xref target="I-D.ietf-alto-deployments"/> for additional discussions on faked ALTO Guidance.

A related scenario is that an ALTO Server could unintentionally give "bad" guidance. For example, if many ALTO Clients follow the Cost Map or Endpoint Cost guidance without doing additional sanity checks or adaptation, more preferable hosts and/or links could get overloaded while less preferable ones remain idle; see AR-14 of [RFC6708] for related application considerations.

14.2.2. Protection Strategies

To protect applications from undesirable ALTO Information Resources, it is important to note that there is no protocol mechanism to require conforming behaviors on how applications use ALTO Information Resources. An application using ALTO may consider including a mechanism to detect misleading or undesirable results from using ALTO Information Resources. For example, if throughput measurements do not show "better-than-random" results when using the Cost Map to select resource providers, the application may want to disable ALTO usage or switch to an external ALTO Server provided by an "independent organization" (see AR-20 and AR-21 in [RFC 6708]). If the first ALTO server is provided by the access network service

provider and the access network service provider tries to redirect access to the external ALTO Server back to the provider's ALTO Server or try to tamper with the responses, the preceding authentication and integrity protection can detect such a behavior.

14.3. Confidentiality of ALTO Information

14.3.1. Risk Scenarios

Although in many cases ALTO Information Resources may be regarded as non-confidential information, there are deployment cases where ALTO Information Resources can be sensitive information that can pose risks if exposed to unauthorized parties. We discuss the risks and protection strategies for such deployment scenarios.

For example, an attacker may infer details regarding the topology, status, and operational policies of a network through the Network and Cost Maps. As a result, a sophisticated attacker may be able to infer more fine-graind topology information than an ISP hosting an ALTO server intends to disclose. The attacker can leverage the information to mount effective attacks such as focusing on high-cost links.

Revealing some endpoint properties may also reveal additional information than the Provider intended. For example, when adding the line bitrate as one endpoint property, such information may be potentially linked to the income of the habitants at the network location of an endpoint.

In <xref target="RFC6708"/> Section 5.2.1, three types of risks associated with the confidentiality of ALTO Information Resources are identified: risk type (1) Excess disclosure of the ALTO service provider's data to an authorized ALTO client; risk type (2) Disclosure of the ALTO service provider's data (e.g., network topology information) to an unauthorized third party; and risk type (3) Excess retrieval of the ALTO service provider's data by collaborating ALTO clients. Section 10 of <xref target="I-D.ietf-alto-deployments"/>also discusses information leakage from ALTO.

14.3.2. Protection Strategies

To address risk type (1), the Provider of an ALTO Server must be cognizant that the network topology and provisioning information provided through ALTO may lead to attacks. ALTO does not require any particular level of details of information disclosure, and hence the Provider should evaluate how much information is revealed and the associated risks.

To address risk type (2), the ALTO Protocol need confidentiality. Since ALTO requires that HTTP over TLS MUST be supported, the confidentiality mechanism is provided by HTTP over TLS.

For deployment scenarios where client authentication is desired to address risk type (2), ALTO requires that HTTP Digestion Authentication MUST be supported to achieve ALTO Client Authentication to limit the parties with whom ALTO information is directly shared. Depending on the use-case and scenario, an ALTO server may apply other access control techniques to restrict access to its services. Access control can also help to prevent Denial-of-Service attacks by arbitrary hosts from the Internet. See <xref target="I-D.ietf-alto-deployments"/> for a more detailed discussion on this issue.

14.3.3. Limitations

ALTO Information Providers should be cognizant that encryption only protects ALTO information until it is decrypted by the intended ALTO Client. Digital Rights Management (DRM) techniques and legal agreements protecting ALTO information are outside of the scope of this document.

14.4. Privacy for ALTO Users

14.4.1. Risk Scenarios

The ALTO Protocol provides mechanisms in which the ALTO Client serving a user can send messages containing Network Location Identifiers (IP addresses or fine-grained PIDs) to the ALTO Server. This is particularly true for the Endpoint Property, Endpoint Cost, and fine-grained Filtered Map services. The ALTO Server or a third-party who is able to intercept such messages can store and process obtained information in order to analyze user behaviors and communication patterns. The analysis may correlate information collected from multiple clients to deduce additional application/content information. Such analysis can lead to privacy risks. For a more comprehensive classification of related risk scenarios, see cases 4, 5, and 6 in [RFC 6708], Section 5.2.

14.4.2. Protection Strategies

To protect user privacy, an ALTO Client should be cognizant about potential ALTO Server tracking through client queries. An ALTO Client may consider the possibility of relying only on Network Map for PIDs and Cost Map amongst PIDs to avoid passing IP addresses of other endpoints (e.g., peers) to the ALTO Server. When specific IP addresses are needed (e.g., when using the Endpoint Cost Service), an

ALTO Client may consider obfuscation techniques such as specifying a broader address range (i.e., a shorter prefix length) or by zeroing out or randomizing the last few bits of IP addresses. Note that obfuscation may yield less accurate results.

14.5. Availability of ALTO Service

14.5.1. Risk Scenarios

An attacker may want to disable ALTO Service as a way to disable network guidance to large scale applications.In particular, queries which can be generated with low effort but result in expensive workloads at the ALTO Server could be exploited for Denial-of-Service attacks. For instance, a simple ALTO query with n Source Network Locations and m Destination Network Locations can be generated fairly easily but results in the computation of n*m Path Costs between pairs by the ALTO Server (see Section 5.2).

14.5.2. Protection Strategies

ALTO Provider should be cognizant of the workload at the ALTO Server generated by certain ALTO Queries, such as certain queries to the Map Service, the Map Filtering Service and the Endpoint Cost (Ranking) Service. One way to limit Denial-of-Service attacks is to employ access control to the ALTO Server. The ALTO Server can also indicate overload and reject repeated requests that can cause availability problems. More advanced protection schemes such as computational puzzles [I-D.jennings-sip-hashcash] may be considered in an extension document.

An ALTO Provider should also leverage the fact that the Map Service allows ALTO Servers to pre-generate maps that can be distributed to many ALTO Clients.

15. Manageability Considerations

This section details operations and management considerations based on existing deployments and discussions during protocol development. It also indicates where extension documents are expected to provide appropriate functionality discussed in [RFC5706] as additional deployment experience becomes available.

15.1. Operations

15.1.1. Installation and Initial Setup

The ALTO Protocol is based on HTTP. Thus, configuring an ALTO Server may require configuring the underlying HTTP server implementation to define appropriate security policies, caching policies, performance settings, etc.

Additionally, an ALTO Service Provider will need to configure the ALTO information to be provided by the ALTO Server. The granularity of the topological map and the cost map is left to the specific policies of the ALTO Service Provider. However, a reasonable default may include two PIDs, one to hold the endpoints in the provider's network and the second PID to represent full IPv4 and IPv6 reachability (see Section 5.2.1), with the cost between each source/destination PID set to 1. Another operational issue that the ALTO Service Provider needs to consider is that the filtering service can degenerate into a full map service when the filtering input is empty. Although this choice as the degeneration behavior provides continuity, the operational impact should be considered.

Implementers employing an ALTO Client should attempt to automatically discover an appropriate ALTO Server. Manual configuration of the ALTO Server location may be used where automatic discovery is not appropriate. Methods for automatic discovery and manual configuration are discussed in [I-D.ietf-alto-server-discovery].

Specifications for underlying protocols (e.g., TCP, HTTP, SSL/TLS) should be consulted for their available settings and proposed default configurations.

15.1.2. Migration Path

This document does not detail a migration path for ALTO Servers since there is no previous standard protocol providing the similar functionality.

There are existing applications making use of network information discovered from other entities such as whois, geo-location databases, or round-trip time measurements, etc. Such applications should consider using ALTO as an additional source of information; ALTO need not be the sole source of network information.

15.1.3. Requirements on Other Protocols and Functional Components

The ALTO Protocol assumes that HTTP client and server implementations exist. It also assumes that JSON encoder and decoder implementations exist.

An ALTO Server assumes that it can gather sufficient information to populate Network and Cost maps. "Sufficient information" is dependent on the information being exposed, but likely includes information gathered from protocols such as IGP and EGP Routing Information Bases (see Figure 1). Specific mechanisms have been proposed (e.g., [I-D.medved-alto-svr-apis]) and are expected to be provided in extension documents.

15.1.4. Impact and Observation on Network Operation

ALTO presents a new opportunity for managing network traffic by providing additional information to clients. The potential impact to network operation is large.

Deployment of an ALTO Server may shift network traffic patterns. Thus, an ALTO Service Provider should consider impacts on (or integration with) traffic engineering and the deployment of a monitoring service to observe the effects of ALTO operations. Note that ALTO-specific monitoring and metrics are discussed in 6.3 of [I-D.ietf-alto-deployments] and future versions of that document. In particular, an ALTO Service Provider may observe that ALTO Clients are not bound to ALTO Server guidance as ALTO is only one source of information.

An ALTO Service Provider should ensure that appropriate information is being exposed. Privacy implications for ISPs are discussed in Section 14.3. Both ALTO Service Providers and those using ALTO Clients should be aware of the impact of incorrect or faked guidance (see Section 10.3 of [I-D.ietf-alto-deployments] and future versions of that document).

15.2. Management

15.2.1. Management Interoperability

A common management API would be desirable given that ALTO Servers may typically be configured with dynamic data from various sources, and ALTO Servers are intended to scale horizontally for fault-tolerance and reliability. A specific API or protocol is outside the scope of this document, but may be provided by an extension document.

Logging is an important functionality for ALTO Servers and, depending on the deployment, ALTO Clients. Logging should be done via syslog [RFC5424].

15.2.2. Management Information

A Management Information Model (see Section 3.2 of [RFC5706] is not provided by this document, but should be included or referenced by any extension documenting an ALTO-related management API or protocol.

15.2.3. Fault Management

Monitoring ALTO Servers and Clients is described in Section 6.3 of [I-D.ietf-alto-deployments] and future versions of that document.

15.2.4. Configuration Management

Standardized approaches and protocols to configuration management for ALTO are outside the scope of this document, but this document does outline high-level principles suggested for future standardization efforts.

An ALTO Server requires at least the following logical inputs:

- Data sources from which ALTO Information is derived. This can either be raw network information (e.g., from routing elements) or pre-processed ALTO-level information in the form of a Network Map, Cost Map, etc.

- Algorithms for computing the ALTO information returned to clients. These could either return information from a database, or information customized for each client.

- Security policies mapping potential clients to the information that they have privilege to access.

Multiple ALTO Servers can be deployed for scalability. A centralized configuration database may be used to ensure they are providing the desired ALTO information with appropriate security controls. The ALTO information (e.g., Network Maps and Cost Maps) being served by each ALTO Server, as well as security policies (HTTP authentication, SSL/TLS client and server authentication, SSL/TLS encryption parameters) intended to serve the same information should be monitored for consistency.

15.2.5. Performance Management

An exhaustive list of desirable performance information from a ALTO Servers and ALTO Clients are outside of the scope of this document. The following is a list of suggested ALTO-specific to be monitored based on the existing deployment and protocol development experience:

- Requests and responses for each service listed in a Information Directory (total counts and size in bytes).
- CPU and memory utilization
- ALTO map updates
- Number of PIDs
- ALTO map sizes (in-memory size, encoded size, number of entries)

15.2.6. Security Management

Section 14 documents ALTO-specific security considerations. Operators should configure security policies with those in mind. Readers should refer to HTTP [RFC2616] and SSL/TLS [RFC5246] and related documents for mechanisms available for configuring security policies. Other appropriate security mechanisms (e.g., physical security, firewalls, etc) should also be considered.

16. References

16.1. Normative References

[IEEE.754.2008] Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE Standard 754, August 2008.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, August 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, March 2009.

[RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.

[RFC6708] Kiesel, S., Previdi, S., Stiemerling, M., Woundy, R., and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Requirements", RFC 6708, September 2012.

16.2. Informative References

[BitTorrent]
"Bittorrent Protocol Specification v1.0", <http://wiki.theory.org/BitTorrentSpecification>.

[Fielding-Thesis]
Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", University of California, Irvine, Dissertation 2000, 2000.

[I-D.akonjang-alto-proxidor]
Akonjang, O., Feldmann, A., Previdi, S., Davie, B., and D. Saucez, "The PROXIDOR Service",

draft-akonjang-alto-proxidor-00 (work in progress), March 2009.

[I-D.ietf-alto-deployments]
Stiemerling, M., Kiesel, S., and S. Previdi, "ALTO Deployment Considerations", draft-ietf-alto-deployments-06 (work in progress), February 2013.

[I-D.ietf-alto-reqs]
Previdi, S., Stiemerling, M., Woundy, R., and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Requirements", draft-ietf-alto-reqs-08 (work in progress), March 2011.

[I-D.ietf-alto-server-discovery]
Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and S. Yongchao, "ALTO Server Discovery", draft-ietf-alto-server-discovery-08 (work in progress), March 2013.

[I-D.ietf-httpbis-p2-semantics]
Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", draft-ietf-httpbis-p2-semantics-22 (work in progress), February 2013.

[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.medved-alto-svr-apis]
Medved, J., Ward, D., Peterson, J., Woundy, R., and D. McDysan, "ALTO Network-Server and Server-Server APIs", draft-medved-alto-svr-apis-00 (work in progress), March 2011.

[I-D.mrw-nat66]
Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", draft-mrw-nat66-16 (work in progress), April 2011.

[I-D.p4p-framework]
Alimi, R., Pasko, D., Popkin, L., Wang, Y., and Y. Yang, "P4P: Provider Portal for P2P Applications", draft-p4p-framework-00 (work in progress), November 2008.

[I-D.saumitra-alto-multi-ps]
Das, S., Narayanan, V., and L. Dondeti, "ALTO: A Multi Dimensional Peer Selection Problem", draft-saumitra-alto-multi-ps-00 (work in progress), October 2008.

[I-D.saumitra-alto-queryresponse]
Das, S. and V. Narayanan, "A Client to Service Query Response Protocol for ALTO", draft-saumitra-alto-queryresponse-00 (work in progress), March 2009.

[I-D.shalunov-alto-infoexport]
Shalunov, S., Penno, R., and R. Woundy, "ALTO Information Export Service", draft-shalunov-alto-infoexport-00 (work in progress), October 2008.

[I-D.wang-alto-p4p-specification]
Wang, Y., Alimi, R., Pasko, D., Popkin, L., and Y. Yang, "P4P Protocol Specification", draft-wang-alto-p4p-specification-00 (work in progress), March 2009.

[P4P-SIGCOMM08]
Xie, H., Yang, Y., Krishnamurthy, A., Liu, Y., and A. Silberschatz, "P4P: Provider Portal for (P2P) Applications", SIGCOMM 2008, August 2008.

[RFC5706] Harrington, D., "Guidelines for Considering Operations and Management of New Protocols and Protocol Extensions", RFC 5706, November 2009.

[RFC6144] Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", RFC 6144, April 2011.

Appendix A. Acknowledgments

Thank you to Jan Seedorf and Sebastian Kiesel for contributions to the Security Considerations section. Ben Niven-Jenkins, Wendy Roome and Michael Scharf gave substantial feedback and suggestions on the protocol design.

We would like to thank the following people whose input and involvement was indispensable in achieving this merged proposal:

Obi Akonjang (DT Labs/TU Berlin),

Saumitra M. Das (Qualcomm Inc.),

Syon Ding (China Telecom),

Doug Pasko (Verizon),

Laird Popkin (Pando Networks),

Satish Raghunath (Juniper Networks),

Albert Tian (Ericsson/Redback),

Yu-Shun Wang (Microsoft),

David Zhang (PPLive),

Yunfei Zhang (China Mobile).

We would also like to thank the following additional people who were involved in the projects that contributed to this merged document: Alex Gerber (ATT), Chris Griffiths (Comcast), Ramit Hora (Pando Networks), Arvind Krishnamurthy (University of Washington), Marty Lafferty (DCIA), Erran Li (Bell Labs), Jin Li (Microsoft), Y. Grace Liu (IBM Watson), Jason Livingood (Comcast), Michael Merritt (ATT), Ingmar Poese (DT Labs/TU Berlin), James Royalty (Pando Networks), Damien Saucez (UCL) Thomas Scholl (ATT), Emilio Sepulveda (Telefonica), Avi Silberschatz (Yale University), Hassan Sipra (Bell Canada), Georgios Smaragdakis (DT Labs/TU Berlin), Haibin Song (Huawei), Oliver Spatscheck (ATT), See-Mong Tang (Microsoft), Jia Wang (ATT), Hao Wang (Yale University), Ye Wang (Yale University), Haiyong Xie (Yale University).

Appendix B. Design History and Merged Proposals

The ALTO Protocol specified in this document consists of contributions from

- P4P [I-D.p4p-framework], [P4P-SIGCOMM08], [I-D.wang-alto-p4p-specification];
- ALTO Info-Export [I-D.shalunov-alto-infoexport];
- Query/Response [I-D.saumitra-alto-queryresponse], [I-D.saumitra-alto-multi-ps];
- ATTP [ATTP]; and

- Proxidor [I-D.akonjang-alto-proxidor].

Appendix C. Authors

[[CmtAuthors: RFC Editor: Please move information in this section to the Authors' Addresses section at publication time.]]

Stefano Previdi
Cisco

Email: sprevidi@cisco.com

Stanislav Shalunov
BitTorrent

Email: shalunov@bittorrent.com

Richard Woundy
Comcast

Richard_Woundy@cable.comcast.com

Authors' Addresses

Richard Alimi (editor)
Google
1600 Amphitheatre Parkway
Mountain View CA
USA

Email: ralimi@google.com

Reinaldo Penno (editor)
Cisco Systems
170 West Tasman Dr
San Jose CA
USA

Email: repenno@cisco.com

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

Email: yry@cs.yale.edu